

VIPR WG
Internet-Draft
Intended status: Standards Track
Expires: January 12, 2012

C. Jennings
Cisco
J. Rosenberg
jdrosen.net
M. Petit-Huguenin
Stonyfish
July 11, 2011

Verification Involving PSTN Reachability: Requirements and Architecture
Overview
draft-jennings-vipr-overview-01

Abstract

The Session Initiation Protocol (SIP) has seen widespread deployment within individual domains, typically supporting voice and video communications. Though it was designed from the outset to support inter-domain federation over the public Internet, such federation has not materialized. The primary reasons for this are the complexities of inter-domain phone number routing and concerns over security. This document reviews this problem space, outlines requirements, and then describes a new model and technique for inter-domain federation with SIP, called Verification Involving PSTN Reachability (ViPR). ViPR addresses the problems that have prevented inter-domain federation over the Internet. It provides fully distributed inter-domain routing for phone numbers, authorized mappings from phone numbers to domains, a new technique for automated VoIP anti-spam, and privacy of number ownership, all while preserving the trapezoidal model of SIP.

Legal

This documents and the information contained therein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION THEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering

Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on January 12, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	5
2. Problem Statement	5
2.1. The Phone Number Routing Problem	6
2.2. The Open Pinhole Problem	7
2.3. Quality of Service Problem	7
2.4. Troubleshooting Problem	8
3. Summary of Existing Solutions	8
3.1. Domain Routing	8
3.2. Public ENUM	9
3.3. Private Federations	9
4. Key Requirements	10
5. Executive Overview	11
5.1. Key Properties	11
5.2. Challenging Past Assumptions	13
5.3. Technical Overview	14
5.3.1. Storage of Phone Numbers	16
5.3.2. PSTN First Call	17
5.3.3. Validation and Caching	18
5.3.4. SIP Call	19
6. Architecture Components and Functions	24
6.1. ViPR Server	25
6.2. Call Agent	26
6.3. Border Element	27
6.4. Enrollment Server	28
6.5. P2P Network	28
7. Protocols	28
7.1. P2P: RELOAD	28
7.1.1. ViPR Usage	29
7.1.2. Certificate Usage	30
7.2. ViPR Access Protocol (VAP)	30
7.3. Validation Protocol	31
7.4. SIP Extensions	32
8. Example Call Flows	32
8.1. PSTN Call and VCR Upload	32
8.2. DHT Query and Validation	34
8.3. DHT Query and No Match	35
8.4. SIP Call	35
9. Security Considerations	36
9.1. Attacks on the DHT	37
9.2. Theft of Phone Numbers	37
9.3. Spam	38
9.4. Eavesdropping	38
10. IANA Considerations	39
11. Acknowledgements	39
12. References	39
12.1. Normative References	39

12.2. Informative References	40
Appendix A. Release notes	41
A.1. Modifications between rosenberg-04 and rosenberg-03	41
Authors' Addresses	41

1. Introduction

The Session Initiation Protocol (SIP) was originally published as RFC 2543 [RFC2543] in May of 1999. This was followed by subsequent publication of RFC 3261 [RFC3261], which brought the protocol to sufficient maturity to enable large scale market adoption.

And indeed, it has seen large scale market adoption. SIP has seen hundreds of implementations, spanning consumer products, enterprise servers, and large scale carrier equipment. It carries billions and billions of minutes of calls, and has become the lingua franca of interconnection between products from different vendors. If one measures success in deployment, then clearly SIP is a success.

Though SIP is used between domains, it is typically through private federation agreements. While such agreements are positive, they cause a "least common denominator" problem, which has limited the growth of advanced SIP features, and prevented the innovation that we expected SIP to drive. SIP was designed from the ground up to enable communications between users in different domains, all over the public Internet. The intention was that real-time communications should be no different than email or the web, with the same any-to-any connectivity that has fueled the successes of those technologies. Though SIP is used between domains, it is typically through private federation agreements. The any-to-any Internet federation model envisioned by SIP has not materialized at scale.

This document introduces a new technology, called Verification Involving PSTN Reachability (ViPR), that enables us to break down the barriers that have prevented inter-domain VoIP. By stepping back and changing some of the most fundamental assumptions about federation, ViPR is able to address the key problems preventing its deployment. ViPR focuses on incremental deployability over the unrealizable nirvana. At the same time, ViPR ensures that SIP's trapezoidal model - direct federation between domains without any intermediate processing beyond IP transport - is realized. That model is required in order to allow innovative new services to be deployed.

2. Problem Statement

The first question that must be asked is this - why haven't we seen widespread adoption of inter-domain SIP federation?

There are many reasons for it. They are - in order of importance - the phone number routing problem, the open pinhole problem, the quality of service problem, and the troubleshooting problem. The two former ones are the most significant.

2.1. The Phone Number Routing Problem

Inter-domain federation requires that the sending domain determine the address of the receiving domain, in the form of a DNS name (example.com) or one or more IP addresses that can be used to reach the domain. In email and in the web, this is easy. The identifiers used by those services - the email address and web URL respectively - embed the address of the receiving domain. A simple DNS lookup is all that is required to route the connection. SIP was designed to use the same email-style identifiers.

However, most SIP deployments utilize phone numbers, and not email-style SIP URIs. This is due to the huge installed base of users that continue to exist solely on the public switched telephone network (PSTN). In order to be reached by users on the PSTN, and in order to reach them, users in SIP deployments need to be assigned a regular PSTN number. Users in SIP deployments need to place that PSTN number on business cards, use it in their email signatures, and in general, give it out to their friends and colleagues, in order to be reached. While those users could additionally have an email style SIP URI, the PSTN number serves as a single, global identifier that works for receiving calls from users on the PSTN as well as users within the same SIP domain. Why have two identifiers when one will suffice? The universality of PSTN numbers is the reason why most SIP deployments continue to use them - often exclusively.

Another reason is that many SIP deployments utilize hardphones or telephony adaptors, and the user interfaces on these devices - patterned after existing phones - only allow phone-number based dialing. Consequently, these users are only allocated PSTN numbers, and not email-style SIP URI.

Finally, a large number of SIP deployments are in domains where the endpoints are not IP. Rather, they are circuit based devices, connected to a SIP network through a gateway. SIP is used within the core of the network, providing lower cost transit, or providing add-on services. Clearly, in these deployments, only phone numbers are used.

Consequently, to make inter-domain federation incrementally deployable and widely applicable, it needs to work with PSTN numbers rather than email-style SIP URI. Telephone numbers, unlike email addresses, do not provide any indication of the address of the domain which "owns" the phone number. Indeed, the notion of phone number ownership is somewhat cloudy. Numbers can be ported between carriers. They can be assigned to a user or enterprise, and then later re-assigned to someone else. Numbers are granted to users and enterprises through a complex delegation process involving the ITU,

governments, and telecommunications carriers, often involving local regulations that vary from country to country.

Therefore, in order to deploy inter-domain federation, domains are required to utilize some kind of mechanism to map phone numbers to the address of the domain to which calls should be routed. Though several techniques have been developed to address this issue, none have achieved large-scale Internet deployments.

2.2. The Open Pinhole Problem

The inter-domain federation mechanism built into SIP borrows heavily from email. Each domain runs a SIP server on an open port. When one domain wishes to contact another, it looks up the domain name in the DNS, and connects to the that server on the open port. Here, "open" means that the server is reachable from anywhere on the public Internet, and is not blocked by firewalls.

This simple design worked well in the early days of email. However, the email system has now become plagued with spam, to the point of becoming useless. Administrators of SIP domains fear - rightfully so - that if they make a SIP server available for anyone on the Internet to contact, it will open the floodgates for VoIP spam, which is far more disruptive than email-based spam [RFC5039]. Administrators also worry - rightfully so - that an open server will create a back-door for denial-of-service and other attacks that can potentially disrupt their voice service. Administrators are simply not willing to take that risk; rightly or wrongly, voice deployments demand higher uptimes and better levels of reliability than email, especially for enterprises.

Fears around spam and denial-of-service attacks, when put together, form the "open pinhole problem" - that domains are not willing to enable SIP on an open port facing the Internet.

To fix this, a new model for federation is needed - a model where these problems are addressed as part of the fundamental design, and not as an after-thought.

2.3. Quality of Service Problem

The Internet does not provide any QoS guarantees. All traffic is best effort. This is not an issue for data transaction services, like web and email. It is, however, a concern when using real-time services, such as voice and video.

That said, there are a large number of existing VoIP deployments that run over the Internet. Though the lack of QoS is a concern, it has

not proven a barrier to deployment. We believe that, if the more fundamental issues - the phone number routing and open pinhole problems - can be addressed, the QoS problem will sort itself out. As such, we do not discuss this issue further here.

2.4. Troubleshooting Problem

The final problem that is stopping large scale inter-domain federation is the troubleshooting problem. When connecting calls between domains, problems will happen. Calls will get blocked. Calls will get misdelivered. Features won't work. There will be one-way media or no media at all. The video won't start. Call quality will be poor.

These problems are common in VoIP deployments, and they are tough to troubleshoot even within a single administrative domain. When real-time services extend inter-domain, the problem becomes worse. A new angle is introduced: the first step is identifying who is at fault.

Fortunately, work is underway to improve the ability for network administrators to diagnose VoIP problems. Common log formats [CLF-SYNTAX] and consistent session IDs [SESSION-ID], for example, can help troubleshoot interdomain calls.

In addition to these, any new technology that facilitates inter-domain federation needs to have troubleshooting built-in, so that it is not a barrier to deployment.

3. Summary of Existing Solutions

Given the value that inter-domain SIP federation brings, it is no surprise that many attempts have been made at solving it. Indeed, these have all been deployed to varying degrees. However, all of them have fundamental limitations that have inhibited widespread deployment.

3.1. Domain Routing

The first solution that has been proposed for SIP inter-domain federation is built into SIP itself - domain routing. In this technique, users utilize email-style SIP URI as identifiers. By utilizing the DNS lookup mechanism defined in [RFC3263], SIP enables calls to be routed between domains in much the same way email is routed between domains.

This technique works well in theory, but it has two limitations which have limited its deployment:

1. The majority of SIP deployments utilize phone numbers, often exclusively. In such a case, domain routing cannot be used.
2. Domain federation brings with it the possibility (and strong likelihood) of the same levels of spam and DoS attacks that have plagued the email system.

These issues have already been discussed above.

3.2. Public ENUM

Public ENUM, defined in [RFC3761], tries to address the phone number routing problem by cleverly placing phone numbers into the public DNS. Clients can then perform a simple DNS lookup on a phone number, and retrieve a SIP URI which can be used to route to that phone number.

Unfortunately, public ENUM requires that the entries placed into the DNS be populated following a chain of responsibility that mirrors the ownership of the numbers themselves. This means that, in order for a number to be placed into the DNS, authorization to do so must start with the ITU, and from there, move to the country, telecom regulator, and ultimately the end user. The number of layers of bureaucracy required to accomplish this is non-trivial. In addition, the telecom operators - which would be partly responsible for populating the numbers into the DNS - have little incentive to do so. As a consequence, public ENUM is largely empty, and is likely to remain so for the foreseeable future.

Instead, ENUM has morphed into a technique for federation amongst closed peering partners, called private ENUM or infrastructure ENUM [RFC5067]. While there is value in this technology, it does not enable the open federation that public ENUM was designed to solve.

It is clear from the legacy of ENUM deployments, that any kind of phone number routing solution should not rely on government or telecom processes for population of the databases.

3.3. Private Federations

Private federations are a cooperative formed amongst a small number of participating domains. The cooperative agrees to use a common technique for federation, and through it, is able to connect to each other. There are many such federations in use today.

Some of these federations rely on a central database, typically run by the federation provider, that can be queried by participating domains. The database contains mappings from phone numbers to domains, and is populated by each of the participating domains, often

manually. Each domain implements an agreed-upon query interface that can be used to access the database when a number is called. Sometimes ENUM is used for this interface (called private ENUM), other times, a SIP redirection is used. Some federations also utilize private IP networks in order to address QoS problems. "SIP trunking" - a service being offered by many telecom operators as a SIP-based PRI replacement - is a form of private federation.

Private federations work, but they have one major limitation: scale. As the number of participating domains grows, several problems arise. Firstly, the size of the databases become unruly. Secondly, the correctness of the database becomes an issue, since the odds of misconfigured numbers (either intentionally or accidentally) increases. As the membership grows further, the odds increase that "bad" domains will be let in, introducing a source of spam and further problems. The owner of the federation can - and often does - assume responsibility for this, and can attempt to identify and shut down misbehaving participants. Indeed, as the size of the federations grow, the owner of the federation needs to spend increasing levels of capital on maintaining it. This, in turn, requires them to charge money for membership, and this can be a barrier to entry.

4. Key Requirements

From the discussion on the problems of inter-domain federation and the solutions that have been attempted so far, several key requirements emerge:

- REQ-1: The solution should allow for federation between any number of domains.
- REQ-2: The solution must enable users in one domain to identify users in another domain through the use of their existing E.164 based phone numbers.
- REQ-3: The solution must work with deployments that utilize any kind of endpoint, including non-IP phones connected through gateways, IP softphones and hardphones.
- REQ-4: The solution should not require any change in user behavior. The devices and techniques that users have been using previously to make inter-domain calls should continue to work, but now result in inter-domain IP federation.
- REQ-5: The solution should work worldwide, for any domain anywhere.
- REQ-6: The solution should not require any new services from any kind of centralized provider. A domain should be able, of its own free-will and accord, to deploy equipment and connect to the federation.

- REQ-7: The solution should not require any prior arrangement between domains in order to facilitate federation between those domains. Federation must occur opportunistically - connections established when they can be.
- REQ-8: The solution must work for domains of any size - starting at a single phone to the largest telecom operator with tens of millions of numbers.
- REQ-9: The solution must have built-in mechanisms for preventing spam and DoS attacks. These mechanisms must be fully automated.
- REQ-10: The solution must not require any processing whatsoever by SIP or RTP intermediaries. It must be possible for a direct SIP connection to be established between participating domains.

These requirements, when put together, appear to be mutually unsolvable. And indeed, they have been - until now.

5. Executive Overview

Verification Involving PSTN Reachability (ViPR) is a new technology that is aimed at solving the problems that have prevented large-scale Internet-based SIP federation of voice and video. ViPR solves these problems by creating a hybrid of three technologies - the PSTN itself, a Peer to Peer (P2P) network, and SIP. By combining all three, ViPR enables an incrementally deployable solution to federation.

5.1. Key Properties

ViPR has several important properties that enable it to solve the federation problem:

Works With Numbers: ViPR enables federation for existing PSTN numbers. It does not require users or administrators to know or configure email-style identifiers. It does not require the allocation of new numbers. It does not require a change in user behaviors. Whatever way users were dialing numbers yesterday, works with ViPR tomorrow.

Works with Existing Endpoints: ViPR does not require any changes to endpoints. Consequently, it works with existing SIP endpoints, or with non-IP endpoints connected through gateways.

Fully Distributed: ViPR does not require any kind of central authority or provider. A domain wishing to utilize ViPR just deploys it on their own. ViPR utilizes the existing PSTN and existing Internet connectivity the domain already has, and by combining them, achieves inter-domain federation. Domains do not need to wait for their service providers to roll out any kind of new features, databases, or functionality.

Verified Mappings: The biggest issue in mapping from a phone number to a domain or IP address, is determining whether the mapping is correct. Does that domain really own the given phone number?

While solutions like ENUM have solved this problem by relying on centralized delegations of authorization, ViPR provides a secure mapping in a fully distributed way. ViPR guarantees that phone calls cannot be misrouted or numbers stolen.

Worldwide: ViPR works worldwide. Any domain that is connected to both the PSTN and the Internet can participate. It doesn't matter whether the domain is in Africa, the Americas, or Australia.

Since ViPR does not depend on availability of any regional services beyond IP and PSTN access - both of which are already available globally - ViPR itself is globally available.

Unlimited Scale: ViPR has nearly infinite scale. Any number of domains can participate.

Self-Scale: ViPR self-scales. This means that the amount of computation, memory, and bandwidth that a domain must deploy scales in direct proportion to the size of their own user base.

Self-Learning: ViPR is completely automated. A domain never, ever has to configure any information about another domain. It never has to provision IP addresses, domain names, certificates, phone number prefixes or routing rules. Without any prior coordination, ViPR enables one domain to connect to a different domain.

Automated Anti-Spam ViPR comes with a built-in mechanism for preventing VoIP spam. This mechanism is new, and specific to VoIP. In this way, it is fundamentally different from existing VoIP anti-spam techniques which borrow from email [RFC5039]. This new technique is fully automated, and requires no configuration by administrators and no participation from end users. Though it is not a 100% solution to the problem, it brings substantial economic and legal ammunition to the table to act as a good deterrent for a long while.

Feature Velocity: ViPR enables direct SIP connections between two domains seeking to federate. There are no SIP intermediaries of any sort between the two. This means that domains have no dependencies on intermediaries for deployment of new features.

Designed for the Modern Internet: ViPR is built to run on the modern Internet. It assumes the worst from everyone. It assumes limited connectivity. It assumes network failures. It assumes there are attackers seeking to eavesdrop calls. Security is built-in and cannot be disabled.

Reliable: ViPR is reliable. Through its hybridization of the PSTN and the Internet, it makes sure that calls always go through. Indeed, to route a call between domains A and B, ViPR never depends on a server or service anywhere outside of domains A and B (besides vanilla PSTN and IP access) being operational.

At first glance, these properties seem impossible to realize. And

indeed, given the assumptions that have traditionally been made about how federation has to work, these properties are impossible to realize. It is only by stepping back, and rethinking these fundamental assumptions, that a solution can be found.

5.2. Challenging Past Assumptions

Two unstated assumptions of SIP federation are challenged by ViPR.

The first assumption that federation solutions have made is this:

The purpose of SIP federation is to eliminate the PSTN, and consequently, we cannot assume the PSTN itself as part of the solution.

Though unstated, this assumption has clearly been part of the design of existing solutions. SIP federation based on email-style URIs, as defined in RFC 3261, doesn't utilize or make mention of the PSTN. Solutions like ENUM, or private registries, do not utilize or make mention of the PSTN. In one sense, it's obvious that they shouldn't - after all, the purpose is to replace the PSTN. However, such an approach ignores an incremental solution - a solution which utilizes the PSTN itself to solve the hard problems in SIP federation.

After all, the PSTN has accomplished a great deal. It reaches worldwide. It provides a global numbering translation service that maps phone numbers to circuits. It is highly reliable, and provides QoS. It has been built up over decades to achieve these goals. This begs the question - can we build upon the capabilities already provided by the PSTN, and use them to solve the problems that plague SIP federation?

Indeed, the answer is yes once another assumption is challenged. This second assumption is:

A federation solution must be the same as the final target federation architecture, and not just a step towards it.

Though unstated, this assumption has also been true. SIP's email-style federation was a pure 'target architecture' - the place we want to get to. ENUM was the same - a worldwide global DNS database with everyone's phone numbers - an unrealizable nirvana of open connectivity.

Historically, technologies are more successful when they are incrementally deployable. Indeed, in many cases, the target architecture is unrealizable because there is no obvious way to get there. As such, the focus needs to be on the next incremental step that we can take, and that step in turn creates the technological and market pressures that will drive the next step. In the end, the target may not be the perfect nirvana we all imagined, but we've at least arrived.

As such, ViPR is very much focused on incremental deployability. It is not the end of the federation story, it is the beginning. It discards the nirvana of perfect IP federation for a solution that federates most, but not all calls, by relying on the PSTN to fill in the gaps. ViPR's philosophy is not to let the perfect be the enemy of the good.

5.3. Technical Overview

A high level view of the architecture is shown in Figure 1. The figure shows four different domains, a.com, b.com, c.com and d.com, federating using ViPR technology. Each domain is connected to both the public Internet and to the traditional PSTN.

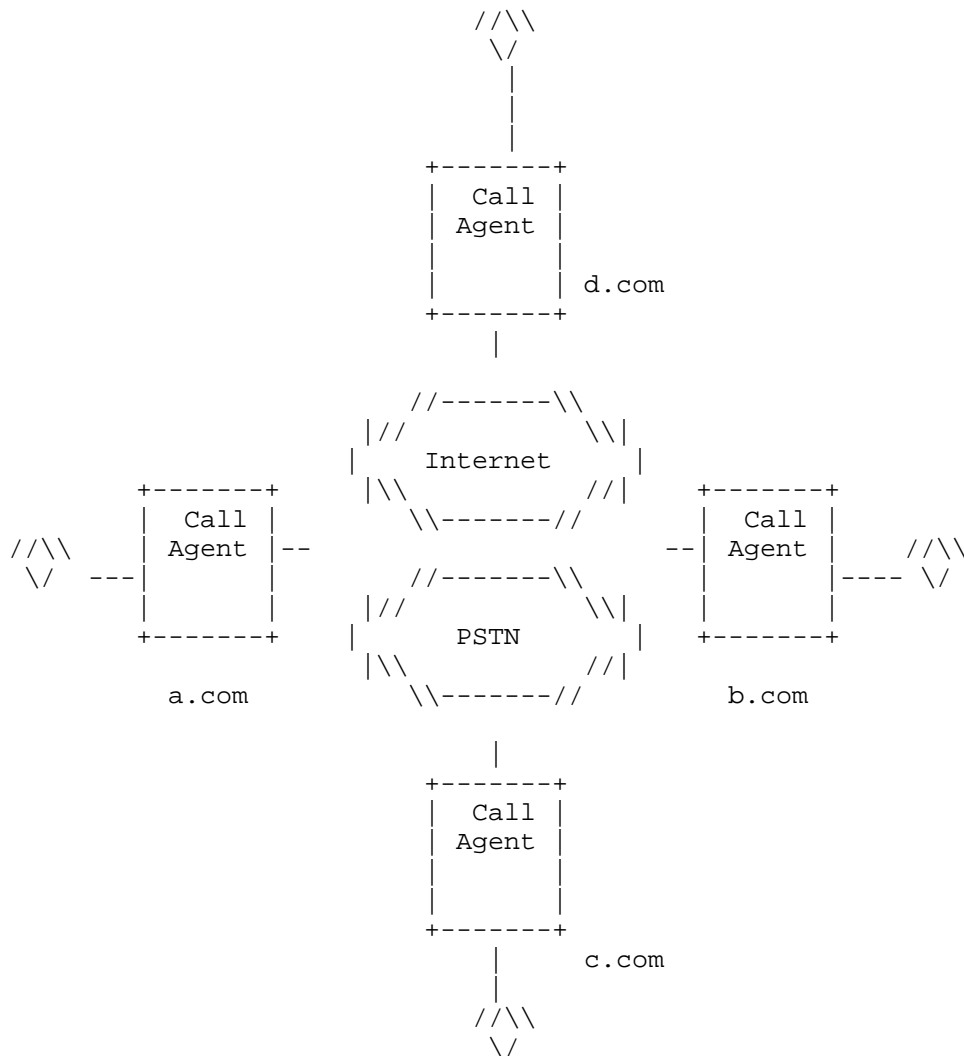


Figure 1: High Level Architecture

For purposes of explanation, it is easiest to think of each domain as having a single call agent which participates in the federation solution. In actuality, the functionality is decomposed into several sub-components, and this is discussed in more detail below. The call agent is connected to one or more phones in the domain, and is responsible for routing calls, handling features, and processing call state. The call agent is stateful, and is aware of when calls start and stop.

Assume that all four domains have a 'fresh' installation of ViPR, and that domain b.com 'owns' +1 408 555 5..., a block of 1000 numbers allocated by its PSTN provider.

The ViPR mechanism can be broken into four basic steps: storage of phone numbers, PSTN first call, validation and caching, and SIP call.

5.3.1. Storage of Phone Numbers

The first step is that the call agents form a single, worldwide P2P network, using RELOAD [P2PSIP-BASE] with the Chord algorithm. This P2P network forms a distributed hash table (DHT) running amongst all participating domains. A distributed hash table is like a simple database, allowing storage of key-value pairs, and lookup of objects by key. Unlike a normal hash table, which resides in the memory of a single computer, a distributed hash table is spread across all of the servers which make up the P2P network. In this case, it is spread across all of the domains participating in the ViPR federation.

The neat trick solved by Chord (and by other DHT algorithms), is an answer to the following: given that the desired operation is to read or write an object with key K, which node in the DHT is the box that currently stores the object with that key? Chord provides a clever algorithm which routes read and write operations through nodes in the DHT until they eventually arrive at the right place. With Chord, this will take no more than $\log_2 N$ hops, where N is the number of nodes in the DHT. Consequently, for a DHT with 1024 nodes, 10 hops are required in the worst case. For 2048, 11 hops. And so on. The logarithmic factor allows DHTs to achieve incredible scale and to provide enormous storage summed across all of the nodes that make up the DHT.

This logarithmic hopping behavior also means that each node in the DHT does not need to establish a TCP/TLS connection to every other node. Rather, connections are established to a smaller subset - just $\log(N)$ of the nodes.

In DHTs, each participating entity is identified by a Node-ID. The Node-ID is a 128 bit number, assigned randomly to each entity. They have no inherent semantic meaning; they are not like domain names or IP addresses.

In the case of ViPR, each call agent is identified by one or more Node-IDs. For purposes of discussion, consider the case where the call agent has just one. Each participating domain, including b.com in our example, uses the DHT to store a mapping from each phone number that it owns, to its own Node-ID. In the case of b.com, it would store 1000 entries into the DHT, each one being a mapping from

one of its phone numbers, to its own Node-ID. Furthermore, when the mappings are stored, the mapping is actually from the SHA-1 hash of the phone number, to the Node-ID of the call agent which claims ownership of that number.

Pretending that the Node-ID of the call agent in domain b.com is 0x1234 (a shorter 16 bit value to simplify discussion), the entries stored into the DHT by b.com would be:

Key	Value
SHA1(+14085555000)	0x1234
SHA1(+14085555001)	0x1234
SHA1(+14085555002)	0x1234
.....	
SHA1(+14085555999)	0x1234

Figure 2: DHT Contents

It is important to note that the DHT does not contain phone numbers (it contains hashes of them), nor does it contain IP addresses or domain names. Instead, it is a mapping from the hash of a phone number (in E.164 format) to a Node-ID.

b.com will store this mapping when it starts up, or when a new number is provisioned. The information is refreshed periodically by b.com. The actual server on which these mappings are stored depends on the Chord algorithm. Typically, the entries will be uniformly distributed amongst all of the call agents participating in the network.

5.3.2. PSTN First Call

At some point, a user (Alice) in a.com makes a call to +1 408 555 5432, which is her colleague Bob. Even though both sides have ViPR, the call takes place over the plain old PSTN. Alice talks to Bob for a bit, and they hang up.

At a random point of time after the call has completed, the call agent in a.com "wakes up" and says to itself, "that's interesting, someone in my domain called +1 408 555 5432, and it went over the PSTN. I wonder if that number is reachable over IP instead?". To make this determination, it hashes the called phone number, and looks it up in the DHT. It is important to note that this lookup is not at the time of an actual phone call - this lookup process happens outside of any phone call, and is a background process.

The query for +1 408 555 5432 will traverse the DHT, and eventually arrive at the node that is responsible for storing the mapping for that number. Typically, that node will not be b.com, but rather one of the other nodes in the network (for example. c.com). In many cases, the called number will not find a matching mapping in the DHT. This happens when the number that was dialed is not owned by a domain participating in ViPR. When that happens, a.com takes no further action. Next time there is another call to the same number, it will repeat the process and check once more whether the dialed number is in the DHT.

In this case, there is a match in the DHT, and a.com learns the Node-ID of b.com. It then proceeds to the validation step. It is also possible that there are multiple matches in the DHT. This can happen if another domain - d.com for example - also claims ownership of that number. When there are multiple matching results, a.com learns all of them, and performs the validation step with each.

5.3.3. Validation and Caching

Why not just store the domain in the DHT, instead of the Node-ID? In that case, once a.com performed the lookup, it would immediately learn that the number maps to b.com, and could then make a direct SIP call next time.

The main reason this doesn't work is security. The information in the DHT is completely untrusted. There is nothing so far that enables a.com to know that b.com does, in fact, own the phone number in question. Indeed, if multiple domains make a claim on the number, it has no way to know which one (if any) actually owns it.

To address this critical problem, ViPR utilizes a technique called phone number validation. Phone number validation is the key concept in ViPR. The essential idea is that a.com will connect to the b.com server, by asking the DHT to form a connection to b.com's Node-ID. Once connected, a.com demands proof of ownership of the phone number. This proof comes in the form of demonstrated knowledge of the previous PSTN call. When a call was placed from a.com to +1 408 555 5432, the details of that call - including its caller ID, start time, and stop time, create a form of shared secret - information that is only known to entities that participated in the call. Thus, to obtain proof that b.com really owns the number in question, a.com will demand a knowledge proof - that b.com is aware of the details of the call. The only way that b.com could know these details is if it had received the call, and the only way it could have received the call is if it owned the phone number.

There are a great many details required for this validation protocol

to be secured. It needs to handle the fact that call start and stop times won't exactly match on both sides. It needs to deal with the fact that many calls start on the top of the hour. It needs to deal with the fact that caller ID is not often delivered, and when it is delivered, is not reliable. It needs to deal with the fact that a.com may in fact be the attacker, trying to use the validation protocol to extract the shared secret from b.com. All of this is, in fact, handled by the protocol. The protocol is based on the Secure Remote Password for TLS Authentication (SRP-TLS) [RFC5054], and is described more fully in [ViPR-PVP].

At the end of the validation process, both a.com and b.com have been able to ascertain that the other side did in fact participate in the previous PSTN call. At that point, a.com sends its domain name to b.com (this is described in more detail below), and b.com sends to a.com - all over a secured channel - a SIP URL to use for routing calls to this number, and a ticket. The ticket is a cryptographic object, opaque to a.com, but used by b.com to allow incoming SIP calls. It is similar in concept to kerberos tickets - it is a grant of access. In this case, it is a grant of access for a.com to call +1 408 555 5432, and only +1 408 555 5432.

The a.com call agent receives the SIP URI and ticket, and stores both of them in an internal cache. This cache builds up slowly over time, containing the phone number, SIP URI, and ticket, for those numbers which are called by a.com and validated using ViPR. Because the cache entries are only built for numbers which have actually been called by users in the enterprise, the size of the cache self-scales. A call agent supporting only ten users will build up a cache proportional to the volume of numbers called by ten people, whereas a call agent supporting ten thousand users will build up a cache which is typically a thousand times larger.

5.3.4. SIP Call

At some point in the future, another call is made to +1 408 555 5432. The caller could be Alice, or it could be any other user attached to the same call agent. This time, the call agent notes that it has a cached route for the number in question, along with a SIP URI that can be used to reach that route. It also has a ticket.

The a.com call agent attempts to contact the SIP URI by establishing a TCP/TLS connection to the SIP URI it learned. If this connection cannot be made, it proceeds with the call over the PSTN. This ensures that, in the event of an Internet failure or server failure, the call can still proceed. Assuming the connection is established, the a.com call agent sends a traditional SIP INVITE to the terminating call agent, over this newly formed secure connection.

The SIP call setup request also contains the ticket, placed into a new SIP header in the message.

When this call setup request arrives at the b.com border element, it can extract the ticket from the new SIP header. This ticket is an object, opaque to a.com, that was previously generated by the b.com call agent. Figure 3 illustrates how this ticket is generated and used.

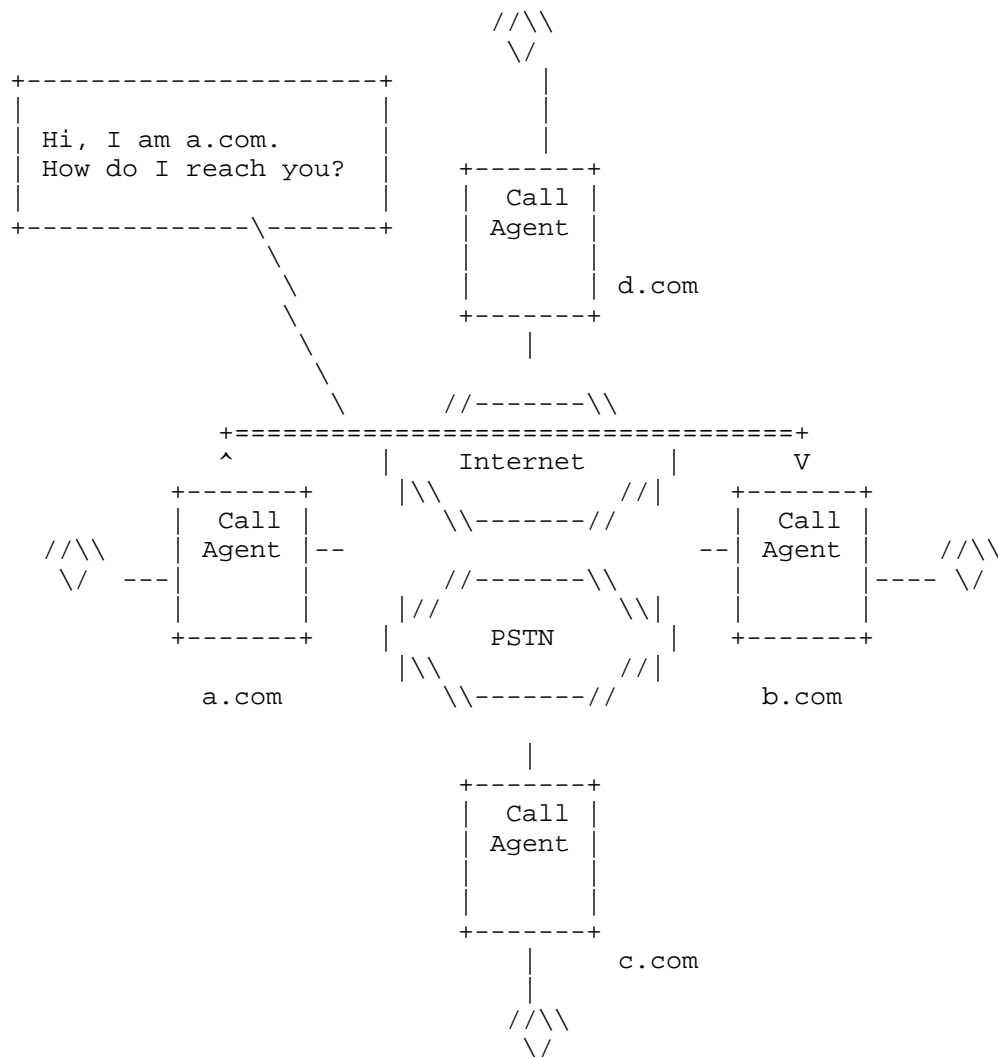


Figure 3: Ticket Validation Step 1

Towards the end of the validation process, domains a.com and b.com had determined that each was, in fact in possession of the shared secret information about the prior PSTN call. However, neither side has any information about the domain names of the other side. The originating domain - a.com - tells b.com that its domain name is a.com. It offers no proof of this assertion at this time.

Next, the b.com domain generates the ticket. The ticket has three fundamental parts to it:

1. The phone number that was just validated - in this case, +1 408 555 5432.
2. The domain name that the originating side claims it has - a.com in this case.
3. A signature generated by b.com, using a key known to itself only, over the other two pieces of information.

This ticket is then sent back to a.com at the end of the validation process, as shown in Figure 4.

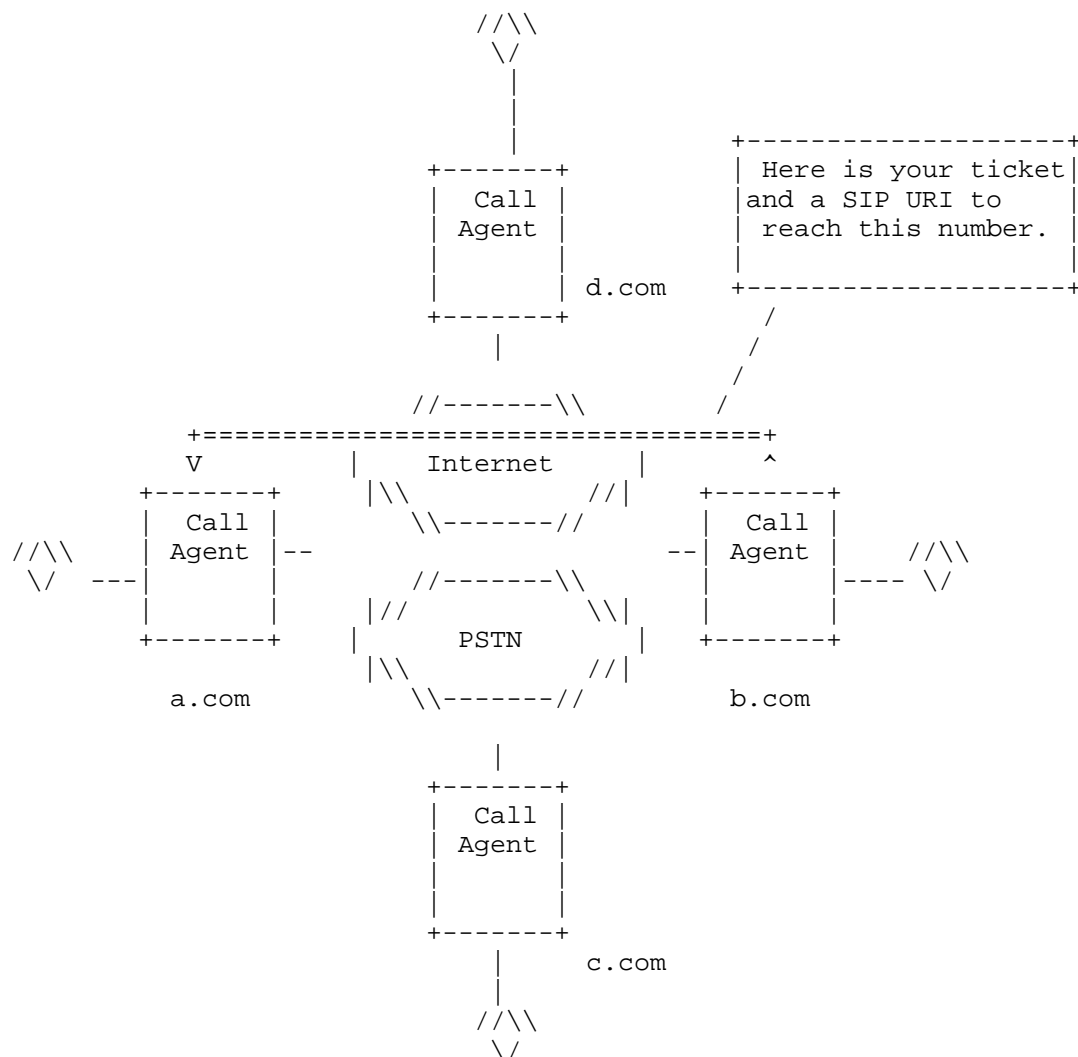


Figure 4: Ticket Validation Step 2

When a.com generates a SIP INVITE, it will contain this ticket. The INVITE arrives at the b.com call agent over the mutually authenticated TLS connection established between the domains.

The b.com border element looks for the SIP header field in the INVITE that contains the ticket. First, it verifies the signature over the ticket. Remember that the b.com agent is the one that generated the ticket in the first place; as such, it is in possession of the key

required to validate the signature. Once validated, it performs two checks:

1. It compares the phone number in the call setup request (the Request URI) against the phone number stored in the ticket.
2. It compares the domain name of the calling domain, learned from the certificates in the mutual TLS exchange, against the domain name stored in the ticket.

If both match, the b.com call agent knows that the calling party is in fact the domain they claimed previously, and that they had in fact gone through the validation process successfully for the number in question. A consequence of this is that the following property is maintained:

A domain can only call a specific number over SIP, if it had previously called that exact same number over the PSTN.

This property is key in fighting spam and denial-of-service attacks. Because calling numbers on the PSTN costs money - especially international calls - ViPR creates a financial disincentive for spammers. For a spammer to ring every phone in a domain with a SIP call, it must have previously called every number in the domain with a PSTN call, and had a successfully completed call to each and every one of them. Of course, once that PSTN call had been placed, the spammer would have already achieved their goals, and at cost. The additional VoIP call is not so exciting.

This property also means that, in order for an attacker to spam call numbers on VoIP, it must have already spam-called those same numbers on the PSTN. This means that the attacker would clearly be subject to regulations and laws governing usage of the PSTN for calling. As an example, a spammer in the United States would have already violated U.S. do-not-call rules by initiating the spam calls to the PSTN numbers.

It is important to note that ViPR does not completely address the spam problem. A large spamming clearing house organization could actually incur the costs of launching the PSTN calls to numbers, and then, in turn, act as a conduit allowing other spammers to launch their calls to those numbers for a fee. The clearinghouse would actually need to transit the signaling traffic (or, divulge the private keys to their domain name), which would incur some cost. As such, while this is not an impossible situation, the barrier is set reasonably high to start with - high enough that it is likely to deter spammers until it becomes a highly attractive target, at which point other mechanisms can be brought to bear. This is, again, an example of the incremental deployability philosophy that ViPR takes -

let not the perfect be the enemy of the good.

6. Architecture Components and Functions

The architecture in Figure 1 is overly simplistic. ViPR allows the functionality embedded within the call agent to be split up into three components, as shown in Figure 5:

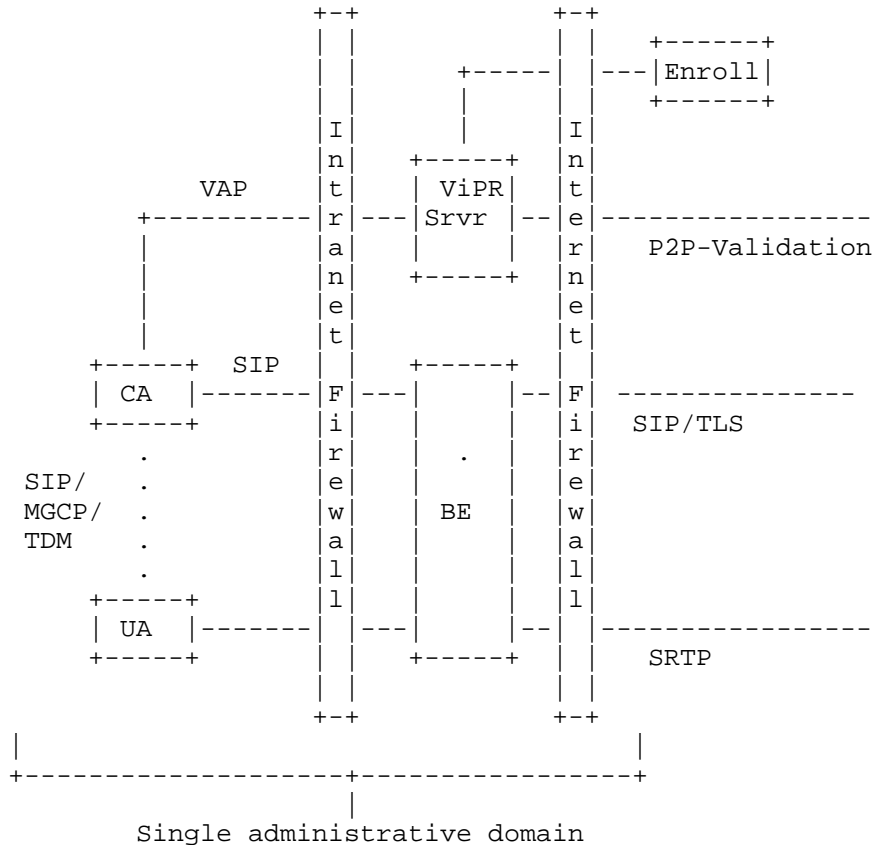


Figure 5: Architecture

Within each domain, there are three components that are ViPR-aware. These are the ViPR server, the call agent (CA), and the border element (BE). Outside of the domain, there is a P2P network and an enrollment server. A domain will typically have firewalls - an Internet firewall and an intranet firewall.

The sections which follow describe the roles and responsibilities of

each component in more detail.

6.1. ViPR Server

The ViPR server is the heart of the system. It performs several key functions:

1. It implements the P2P protocol, acting as one or more nodes in the DHT. By placing this function separate from the call agent, it allows the call agent to be isolated from the traffic and security concerns that are often associated with a P2P network.
2. It implements the validation mechanism. It is informed of call events by the call agent, and sometime after the call, looks up the number in the DHT, and if found, attempts to connect to the node claiming ownership of the number, and then validates it.
3. It pushes newly learned routes to the call agent once validation has occurred. The ViPR server does not hold the call routes; this eliminates the need for an off-box query to perform call routing logic.
4. It stores numbers into the DHT. The call agent informs the ViPR servers of numbers to be published, and the ViPR server places them into the P2P network. Refreshing the stored numbers (by asking the ViPR server to re-store them) is the responsibility of the call agent.
5. It implements a distributed quota enforcement algorithm, ensuring that malicious ViPR servers cannot store excessive data into the network.
6. It implements a policing function, pacing its store and fetch requests into the DHT to ensure that the network is not overwhelmed.

In order to join the P2P network and be able to receive incoming validation requests, the ViPR server must have open access to the public Internet. For this reason, it is typically placed into the DMZ. The Internet firewall will not require any pinholes to be opened towards the ViPR server.

It is important to understand that the ViPR server does not perform any call processing. It does not process SIP or RTP traffic. It is a non-real-time server that performs validation processing in the background, outside of actual call attempts.

The ViPR server needs to connect with the call agent. This is done through the ViPR Access Protocol (VAP). VAP is described in more detail below.

6.2. Call Agent

The call agent is a box within the domain which performs call processing on behalf of one or more phones within the domain. ViPR can work with a wide variety of call agents, as long as they meet some specific criteria:

- o The call agent must be know of the start time, stop time, caller ID, and called numbers of calls placed from phones towards the PSTN.
- o The call agent must be capable of making routing decisions for outbound calls from phones that would otherwise go to the PSTN, directing them towards the PSTN or towards other domains (based on ViPR routing rules).

Based on this definition, many different types of products typically found within a domain could act as the call agent. An IP PBX or TDM PBX with a SIP interface can be the call agent. A Session Border Controller (SBC) that connects calls from a PBX to the PSTN, can act as the call agent. An IMS application server can act as the call agent. A PSTN gateway, used for all calls egressing a domain from a set of phones, can act as a call agent.

A SIP proxy can act as a call agent; as long as it is capable of stashing the relevant call information into Record-Route headers for usage at the end of the call, it can even operate without retaining call state.

A single phone can also act as the call agent, representing itself and its own phone number.

In ViPR, the call agent performs several key functions specific to ViPR:

- o It informs the ViPR server of the phone numbers to be stored in the DHT for its domain.
- o It refreshes those numbers in the DHT, redoing the storage operation periodically.
- o At the end of a call, the call agent sends a ViPR Call Record (VCR) to the ViPR server, containing the start time, stop time, caller ID and called party number.
- o It learns validated routes from the ViPR server. These routes consist of a phone number, a SIP URI to utilize when contacting that phone number, and a corresponding ticket. The call agent is responsible for storing those routes.
- o When a call is to be made towards a PSTN number, the call agent is responsible for checking whether there is a route for that number, learned via a prior notification from the ViPR server. If so, it

is responsible for sending the INVITE towards the learned SIP URI, and for including the ticket the ViPR-Ticket header field.

Those functions which require communications with the ViPR server are done by implementing VAP. VAP is a client-server protocol, with the call agent acting as the client, and the ViPR server acting as the server. For this reason, the call agent is sometimes called the VAP client or ViPR client.

6.3. Border Element

The border element is responsible for the SIP layer perimeter security functions. In particular:

- o The border element ensures that all egress SIP traffic is carried over TLS. Border elements must reject any incoming SIP requests which are not over TLS. SIP over TLS is mandatory-to-use in ViPR, and it must be performed using mutual TLS.
- o The border element ensures that all egress RTP traffic is actually carried using SRTP. If the traffic originated by the UA in the domain is inherently SRTP, the criteria is met. However, many domains do not utilize SRTP internally, and if it is not used internally, the border element must convert to SRTP. Similarly, the border element is responsible for rejecting any incoming SIP calls that are not set up with SRTP. SRTP is mandatory in ViPR.
- o The border element ensures that ingress and egress SIP traffic is 'fixed up' so that it can pass through the Internet firewall successfully. Typically, this is done using a traditional SBC/ALG function.
- o The border element inspects all incoming SIP INVITES, and performs ticket verification. In this process, it looks for the ViPR-Ticket header field in the INVITE. If not present, it discards the request. If present, it verifies the signature, and then compares the called number and remote TLS domain against the contents of the ticket. If they do not match, the border element discards the INVITE.

The border element can perform other, non-ViPR tasks, as is common for border elements. These include header inspection and validation, anti-virus checks on embedded content, SIP state machine conformance, policy checks on various services, and so on.

The role of the border element can be fulfilled by any number of products typically found within domains. These include Session Border Controllers and firewalls. Indeed, the border element function can be embedded directly in the Internet firewall.

The border element is connected to the call agent via SIP, and to the

user agent (UA) via RTP. The border element has no direct connection to the ViPR server. However, in order for ticket processing to work in this model, the ViPR server and border element must share a secret that is used to create the tickets. This is discussed in more detail below.

6.4. Enrollment Server

P2P protocols - including RELOAD - require the usage of an enrollment server in order to obtain the certificates that are used to secure the network. ViPR uses, and indeed requires, that all RELOAD traffic be over TCP/TLS with mutual authentication. The certificates used are obtained through an enrollment process. The details on how P2P enrollment is done are beyond the scope of this document.

6.5. P2P Network

The collection of ViPR servers form a single, worldwide, P2P network utilizing RELOAD and the Chord algorithm.

It is very important to understand that the DHT is never accessed in real-time. It is not queried at call setup time. This is because the DHT is slow, involving many hops. Queries could take seconds. Furthermore, we don't want to rely on proper operation of the DHT to actually make calls.

7. Protocols

The overall ViPR solution utilizes several protocols, each performing a different function.

7.1. P2P: RELOAD

ViPR utilizes the RELOAD protocol [P2PSIP-BASE] to run amongst each of the ViPR servers. Each ViPR server acts as one or more nodes in the DHT. The number of nodes that the ViPR server implements directly determines the quota allocated to that ViPR server, and in turn, the amount of work it must perform storing data.

ViPR, however, does not implement the SIP usage that has been defined for RELOAD [P2PSIP-SIP]. That is because the DHT is not used as a traditional distributed registrar. Instead, it implements a new usage - the ViPR usage - which stores phone numbers. It also utilizes the DHT for storage of certificates, using a certificate usage.

7.1.1.1. ViPR Usage

The ViPR usage is described in detail in [VIPR-RELOAD-USAGE]. This section provides a brief overview.

The ViPR usage makes use of the dictionary type. Each resource-ID is a key, computed by taking the SHA1 hash of an E.164 formatted phone number. The value stored at this resource-ID is a dictionary. The dictionary entries are the set of virtual ViPR servers which claim ownership of those numbers.

Since a ViPR server might support a multiplicity of call agents from different domains, it is necessary to logically segment a ViPR server so that - from a security perspective - it operates logically like different virtual ViPR servers, one for each call agent. Each virtual instance of a ViPR server is called a VService. Thus, the entries in the dictionary are key value pairs whose key is the concatenation of the Node-ID and an identifier for the VService within that node. The value at each key is the Node-ID to contact for validation.

When a node in the DHT receives a Store request, and it is the responsible node for the resource-ID, it will verify that the Node-ID in both the key and value of the dictionary entry match the Node-ID in the certificate it presents. This ensures that one ViPR server can never overwrite data from another ViPR server.

The ViPR usage also specifies a quota mechanism. Unlike the SIP usage, where there are very specific rules about what resource-IDs a node may store into the DHT, with ViPR, there is no way to restrict what resource-IDs may be stored by a ViPR server. This is because, in ViPR, the resource-IDs are derived from phone numbers, and at the time of storage, there is no way to know whether the node performing the store actually owns this phone number. Consequently, a responsible node will accept stores from any node for any resource-ID. However, to limit malicious users from consuming all of the resources of the DHT, the ViPR usage imposes a quota on storage. Each node performing a store is allocated a fixed quota on the number of records it can place into the DHT. A probabilistic enforcement model is utilized at each responsible node based on the fraction of the hashspace owned by that responsible node. Roughly speaking, if the system quota is 10,000 phone numbers per Node-ID, if a responsible node owns 10% of the DHT, it will accept an average of 1000 phone numbers from any one single Node-ID.

7.1.2. Certificate Usage

Further details pending.

7.2. ViPR Access Protocol (VAP)

The ViPR Access Protocol (VAP) is documented in [VIPR-VAP].

VAP is a client-server protocol that runs between the call agent and the ViPR server. VAP is a simple, binary based, request/response protocol. It utilizes the same syntactic structure and transaction state machinery as STUN [RFC5389], but otherwise is totally distinct from it. VAP clients initiate TCP/TLS connections towards the ViPR server. The ViPR server never opens connections towards the call agent. This allows the ViPR servers to run on the public side of NATs and firewalls.

Once the connections are established, the call agent sends a Register message to the ViPR server. This register message primarily provides authentication and connects the client to the ViPR server. VAP provides several messages for different purposes:

- o Publish: The Publish message informs the ViPR server of service information. There are two types of Publishes supported in ViPR. The first is the ViPR Service (VService). This informs the ViPR server of the SIP URIs on the call agent and black and white lists used by the ViPR server to block validations. The ViPR server stores that information locally and uses it during the validation process, as described above. The second Publish is the ViPR number service. The ViPR server, upon receiving this message, performs a Store operation into the DHT.
- o UploadVCR: This message comes in two flavors - an originating and terminating message. An originating UploadVCR comes from a call agent upon completion of a non-ViPR call to the PSTN. A terminating UploadVCR comes from an agent upon completion of a call received FROM the PSTN. The ViPR server behavior for both messages is very different. For Originating UploadVCR, the ViPR server will store these, and at a random time later, query the DHT for the called number and attempt validation against the ViPR servers that are found. For a terminating UploadVCR, the ViPR server will store these, awaiting receipt of a validation against them.
- o Subscribe: Call agents can subscribe for information from the ViPR server. There is one service that the call agent can subscribe for: number Service. When a new number is validated, the ViPR server will send a Notify to the call agent, containing the validated number, the ticket, and a set of SIP trunk URIs.

- o Notify: The ViPR server sends this message to the call agent when it has an event to report for a particular subscription.

The VAP protocol provides authentication by including an integrity object in each message. This integrity message is the hash of the contents of the message and a shared secret between the ViPR server and the client. VAP can also be run over TLS, which enhances security further.

The P2P network introduces rate limits for the purposes of performance management and limiting denial of service attacks. Each node in the DHT comes with it a limit on the amount of stores per second, reads per second, and total amount of data it can store in the DHT. The ViPR server rigorously follows those limits.

As a consequence, when numbers are stored into the DHT, they are written in slowly based on the rate limits. The call agent will send a Publish operation for each individual number. The ViPR server will perform the store in a rate-limited fashion. When the store is complete, the ViPR server responds to the Publish, and the call agent can move to the next DID to publish. Thus, it may take hours or even days to fully store the set of numbers into the DHT. The process then repeats several days later in order to refresh the data in the DHT.

7.3. Validation Protocol

The core of ViPR is the validation protocol. The validation protocol is used by one ViPR server to connect to another, demand proof-of-knowledge of a previous PSTN call, and once proven, securely learn a SIP URI and ticket for usage in future SIP calls between domains.

The validation protocol is documented in [VIPR-PVP].

The validation protocol is built using TLS-SRP [RFC5054]. TLS-SRP creates a secure TLS connection, but instead of using certificates, utilizes a password. TLS-SRP was designed for cases where the passwords are relatively weak. In the case of the validation protocol, the passwords are formed from parameters of a previous PSTN call. Once a secure TLS connection is formed, a simple request/response protocol is run over it. The request contains the domain name of the originating ViPR server, and the response contains the SIP URI and ticket for that number.

The validation protocol properly handles time offsets between the two domains for the start and stop times of the calls, the relatively weak entropy of a single phone call, the grand chessmaster attack, and non-delivery or inaccurate delivery of caller-ID, amongst other

issues. The validation protocol can be tuned by administrators to allow for arbitrary levels of security, measured in terms of equivalent entropy. The equivalent entropy is the number of bits of entropy that must be demonstrated, as if the domains were authenticating each other using a password with that amount of entropy. This gives domains a 'nerd knob' they can turn to trade off security for performance.

Because the validation protocol utilizes TLS-SRP, it does not run directly through the DHT. This is why a ViPR server requires a separate pinhole to be opened for the validation protocol.

7.4. SIP Extensions

The connection between the call agents in different domains is SIP. ViPR requires that the inter-domain connections run over TLS, and furthermore, utilize SRTP keyed with Sdescriptions.

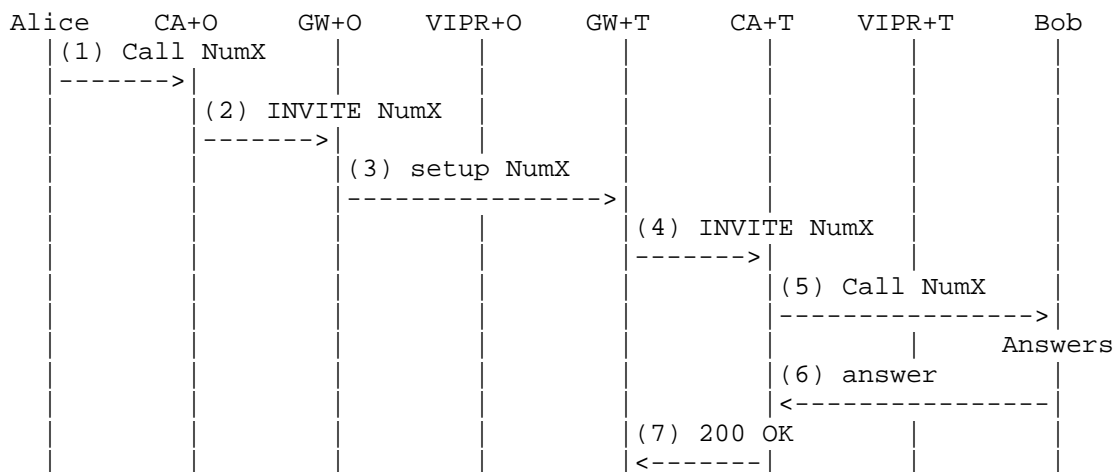
ViPR extends SIP with its anti-spam mechanism. This takes the form of a ticket, present in a SIP header field. [VIPR-SIP-ANTISPAM] defines this header field and the format of the ticket it contains.

8. Example Call Flows

This section provides call flows for the key use cases.

8.1. PSTN Call and VCR Upload

A call flow for the initial PSTN call and VCR upload is shown in Figure 6.



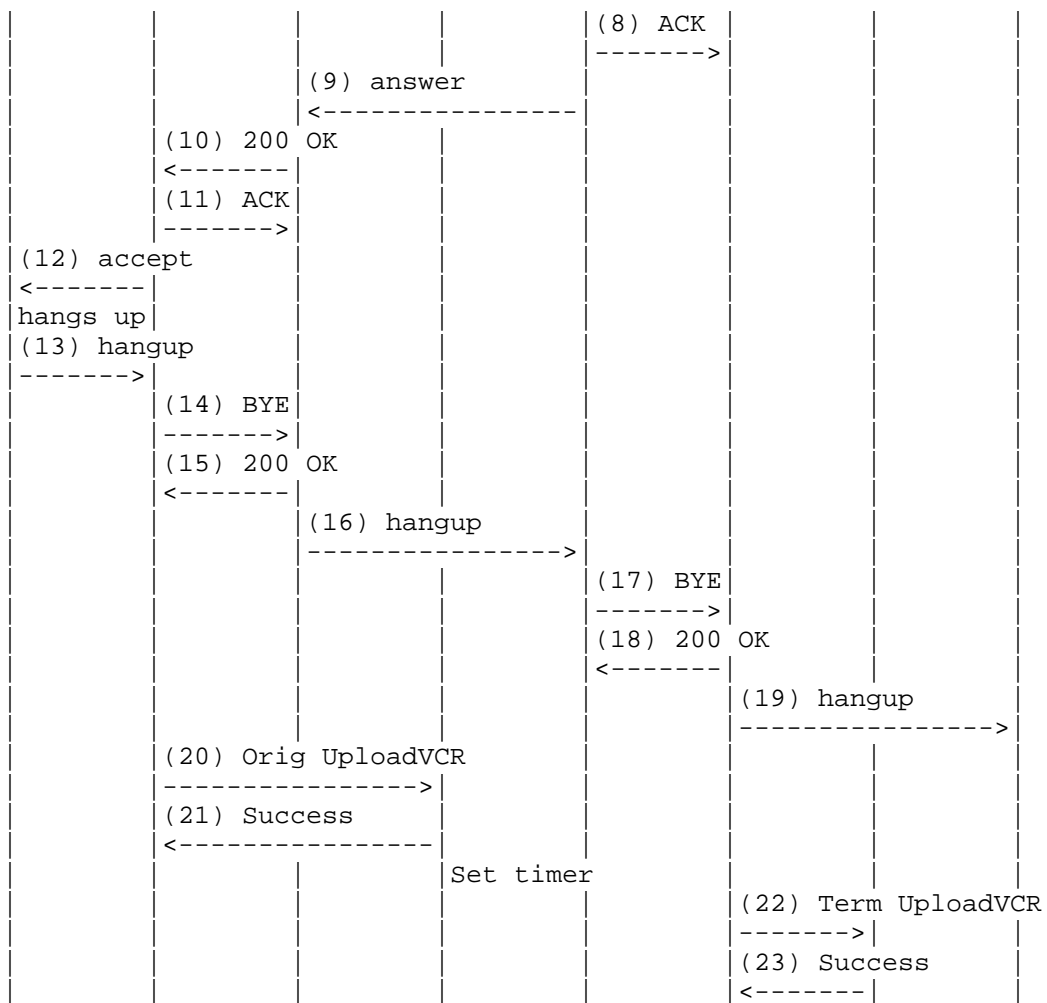


Figure 6: PSTN Call and Upload

In message 1, Alice calls the number of her colleague, Bob. This is NumX. This call is routed over the PSTN, through the terminating call agent, and rings Bob's phone (messages 1-5). Bob answers the phone, and this is propagated back to Alice (messages 6-12). Bob and Alice talk for a while, and then Alice hangs up. This hangup is propagated to Bob, and the call is terminated (messages 13-19).

The originating call agent notes that this call went to the PSTN, and might be a candidate for a future SIP call. It sends an UploadVCR message to its ViPR server (message 20), containing the start time,

stop time, callerID and called party number. The ViPR server acknowledges this (message 21), and then sets a timer for a random time into the future, at which point it will attempt validation. The terminating side is similar; it sends an UploadVCR to its ViPR server (message 22), which is acknowledged (message 23). The terminating side does not set a timer; it waits for a possible validation attempt which may or may not arrive in the future.

8.2. DHT Query and Validation

This section provides the call flow for what happens on the originating ViPR server when the timer fires, in Figure 7.

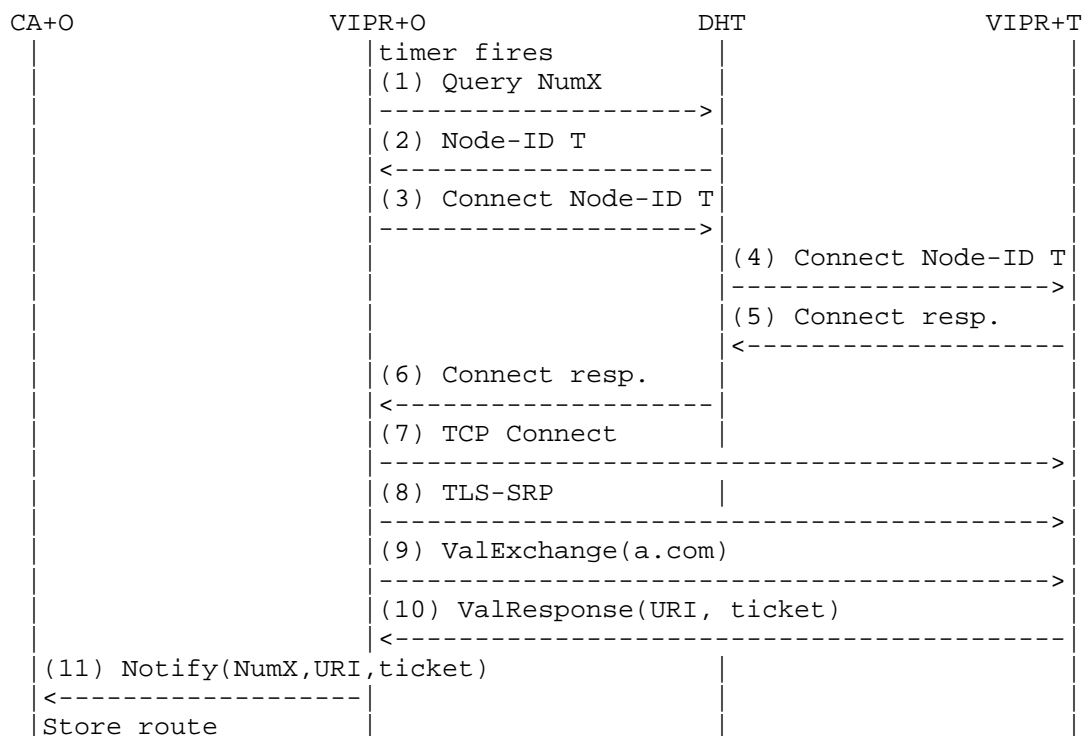


Figure 7: Validation Flow

First, the timer that was set by the originating ViPR server in Figure 6 fires. When it fires, the ViPR server examines the called party number from the VCR. It performs a query into the DHT, to see if this number has been stored by any domain (message 1). In this case, it has, and the DHT returns with a successful query response (message 2). This response indicates that the terminating ViPR

server, with node-ID T, claims ownership of the number.

The originating ViPR server asks the DHT to form a connection between itself and the terminating ViPR server. This message exchanges IP addresses and ports through which a TCP connection can be attempted; details are omitted (messages 3-6). Now, the originating ViPR server can establish a TCP connection to the terminating ViPR server (message 7). Next, the originating ViPR server begins negotiation of a TLS-SRP connection. The TLS-SRP uses the caller ID and called number as a "username" for this exchange, and the start time and stop time of the call as a password. As both sides share the same values for this secret, the secure connection is established. This is now a TLS connection between the two ViPR servers.

Over this secure connection, the originating ViPR server sends a ValExchange request. This request contains the domain name that is claimed by the originating ViPR server (this claim is not verified at this time) (message 9). This is received by the terminating ViPR server, which then creates a ticket for that domain and NumX, and passes the ticket and the SIP URI back to the originating ViPR server (message 10). The originating ViPR server sends this information to its call agent (message 11), which then stores it for usage in a future call.

8.3. DHT Query and No Match

In this case, after the PSTN call of Figure 6, the timer fires, but the originating ViPR server finds no match in the DHT. This is an alternative case to the flow in Figure 7.

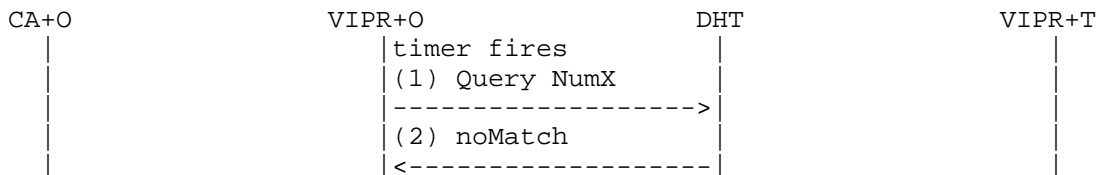


Figure 8: DHT No-Match

8.4. SIP Call

In this case, shown in Figure 9, a user makes a call to a number which has been learned via ViPR.

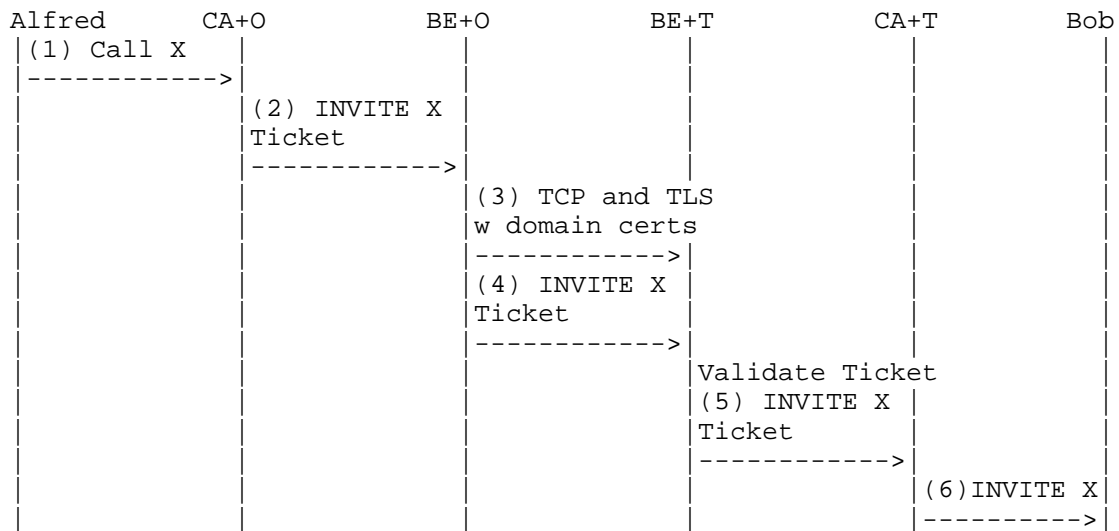


Figure 9: SIP Call

First, a user in the originating domain - Alfred - calls Bob's number (message 1). The originating call agent notes that it has a cached route for that number. It extracts the SIP URI, using it as the topmost Route header field, and then attaches the ticket to the ViPR-Ticket header field. This INVITE is sent to a default next hop border element (message 2). The border element establishes a TCP/TLS connection with the domain in the Route header. It uses a traditional domain certification for this TLS connection (message 3). Once established, it sends the INVITE over the connection (message 4).

This arrives at the terminating call agent, which extracts the ticket and verifies it. To verify it, it checks the signature using the key that was used to create the ticket. Then, it compares the domain name in the ticket with the domain name from the TLS connection handshake. Finally, it compares the called party number in the Request-URI with the value from the ticket. Assuming they all match, the call is forwarded to the terminating call agent (message 5), where it is finally delivered to Bob (message 6).

9. Security Considerations

Security is incredibly important for ViPR. This section provides an overview of some of the key threats and how they are handled.

9.1. Attacks on the DHT

Attackers could attempt to disrupt service through a variety of attacks on the DHT.

Firstly, it must be noted that the DHT is never used at call setup time. It is accessed as a background task, solely to learn NEW numbers and routes that are not already known. If, by some tragedy, an attacker destroyed the P2P network completely, it would not cause a single call to fail. Furthermore, it would not cause calls to revert to the PSTN - calls to routes learned previously would still go over the IP network. The only impact to such a devastating attack, is that a domain could not learn *new* routes to new numbers, until the DHT is restored to service. This service failure is hard for users and administrators to even notice.

That said, ViPR prevents many of these attacks. The DHT itself is secured using TLS - its usage is mandatory. Quota mechanisms are put into place that prevent an attacker from storing large amounts of data in the DHT. Other attacks are prevented by mechanisms defined by RELOAD itself, and are not ViPR specific.

9.2. Theft of Phone Numbers

The key security threat that ViPR is trying to address is the theft of phone numbers. In particular, a malicious domain could store, into the DHT, phone numbers that it does not own, in an attempt to steal calls targeted to those numbers. This attack is prevented by the core validation mechanism, which performs a proof of knowledge check to verify ownership of numbers.

An attacker could try to claim numbers it doesn't own, which are claimed legitimately by other domains in the ViPR network. This attack is prevented as well. Each domain storing information into the DHT can never overwrite information stored by another domain. As a consequence, if two domains claim the same number, two records are stored in the DHT. An originating domain will validate against both, and only one will validate - the real owner.

An attacker could actually own a phone number, use it for a while, validate with it, and build up a cache of routes at other domains. Then, it gives back the phone number to the PSTN provider, who allocates it to someone else. However, the attacker still claims ownership of the number, even though they no longer have it. This attack is prevented by expiring the learned routes after a while. Typically, operators do not re-assign a number for a few months, to allow out-of-service messages to be played to people that still have the old number. Thus, the TTL for cached routes is set to match the

duration that carriers typically hold numbers.

An attacker could advertise a lot of numbers, most of which are correct, some of which are not. ViPR prevents this by requiring each number to be validated individually.

An attacker could make a call so they know the call details of the call they made and use this to forge a validation for that call. They could then try to convince other users, which would have to be in the same domain as the attacker, to trust this validation. This is mitigated by not sharing validations inside of domains where the users that can originate call from that domain are not trusted by the domain.

9.3. Spam

Another serious concern is that attackers may try to launch VoIP spam (also known as SPIT) calls into a domain. ViPR prevents this by requiring that a domain make a PSTN call to a number before it will allow a SIP call to be accepted to that same number. This provides a financial disincentive to spammers. The current relatively high cost of international calling, and the presence of national do-not-call regulations, have prevented spam on the PSTN to a large degree. ViPR applies those same protections to SIP connections.

As noted above, ViPR still lowers the cost of communications, but it does so by amortizing that savings over a large number of calls. The costs of communications remain high for infrequent calls to many numbers, and become low for frequent calls to a smaller set of numbers. Since the former is more interesting to spammers, ViPR gears its cost incentives away from the spammers, and towards domains which collaborate frequently.

Of course, ViPR's built-in mechanism is not a guarantee. A SPIT clearinghouse could shoulder the costs of the PSTN calls, and then re-sell its access for a fee. However, this still causes the clearinghouse to utilize non-trivial resources in its attack. Though these costs are less than the PSTN, they are more than zero, and should act as a deterrent for a long while.

9.4. Eavesdropping

Another class of attacks involves outsiders attempting to listen in on the calls that run over the Internet, or obtain information about the call through observation of signaling.

All of these attacks are prevented by requiring the usage of SIP over TLS and SRTP. These are mandatory to use.

10. IANA Considerations

This specification does not require any actions from IANA.

11. Acknowledgements

Thanks for review comments from Ken Fischer, Rob Maidhof, Michael Procter, and others. Thanks to Theo Zourzouvillys for pointing out the 5th thief of phone numbers attack.

12. References

12.1. Normative References

[P2PSIP-BASE]

Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol", draft-ietf-p2psip-base-11 (work in progress), October 2010.

[P2PSIP-SIP]

Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., and H. Schulzrinne, "A SIP Usage for RELOAD", draft-ietf-p2psip-sip-05 (work in progress), July 2010.

[VIPR-RELOAD-USAGE]

Rosenberg, J., Jennings, C., and M. Petit-Huguenin, "A Usage of Resource Location and Discovery (RELOAD) for Public Switched Telephone Network (PSTN) Verification", draft-rosenberg-dispatch-vipr-reload-usage-03 (work in progress), October 2010.

[VIPR-SIP-ANTISPAM]

Rosenberg, J., Jennings, C., and M. Petit-Huguenin, "Session Initiation Protocol (SIP) Extensions for Blocking VoIP Spam Using PSTN Validation", draft-rosenberg-dispatch-vipr-sip-antispam-03 (work in progress), October 2010.

[VIPR-VAP]

Rosenberg, J., Jennings, C., and M. Petit-Huguenin, "Verification Involving PSTN Reachability: The ViPR Access Protocol (VAP)", draft-rosenberg-dispatch-vipr-vap-03 (work in progress), October 2010.

[VIPR-PVP]

Rosenberg, J., Jennings, C., and M. Petit-Huguenin, "The Public Switched Telephone Network (PSTN) Validation Protocol (PVP)", draft-rosenberg-dispatch-vipr-pvp-03 (work in progress), October 2010.

12.2. Informative References

- [RFC2543] Handley, M., Schulzrinne, H., Schooler, E., and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 2543, March 1999.
- [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.
- [RFC3263] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, June 2002.
- [RFC5039] Rosenberg, J. and C. Jennings, "The Session Initiation Protocol (SIP) and Spam", RFC 5039, January 2008.
- [RFC3761] Faltstrom, P. and M. Mealling, "The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)", RFC 3761, April 2004.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC5067] Lind, S. and P. Pfautz, "Infrastructure ENUM Requirements", RFC 5067, November 2007.
- [RFC5054] Taylor, D., Wu, T., Mavrogiannopoulos, N., and T. Perrin, "Using the Secure Remote Password (SRP) Protocol for TLS Authentication", RFC 5054, November 2007.
- [CLF-SYNTAX] Roach, A., "Binary Syntax for SIP Common Log Format", draft-roach-sipping-clf-syntax-01 (work in progress), May 2009.
- [SESSION-ID] Kaplan, H., "A Session Identifier for the Session Initiation Protocol (SIP)", draft-kaplan-sip-session-id-02 (work in progress), March 2009.

Appendix A. Release notes

This section must be removed before publication as an RFC.

A.1. Modifications between rosenberg-04 and rosenberg-03

- o Nits.
- o Shorter I-Ds references.
- o Changed phone numbers to follow E.123 presentation.
- o Expanded P2P initialisms.
- o Uses +1 408 555 prefix for phone numbers in examples.

Authors' Addresses

Cullen Jennings
Cisco
170 West Tasman Drive
MS: SJC-21/2
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@cisco.com

Jonathan Rosenberg
jdrosen.net
Monmouth, NJ
US

Email: jdrosen@jdrosen.net
URI: <http://www.jdrosen.net>

Marc Petit-Huguenin
Stonyfish

Email: marc@stonyfish.com

P2PSIP
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2012

J. Rosenberg
jdrosen.net
C. Jennings
Cisco
M. Petit-Huguenin
Stonyfish
July 4, 2011

Proportional Quota in REsource LOcation And Discovery (RELOAD)
draft-petithuguenin-p2psip-proportional-quota-01

Abstract

This document defines an extension to RELOAD [I-D.ietf-p2psip-base] that limits the number of a specific kind element that can be stored by one RELOAD peer.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to format it for publication as an RFC or to translate it into languages other than English.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on January 5, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Quota Algorithm	3
4. Overlay Configuration Document Extension	4
5. Security Considerations	5
6. IANA Considerations	5
7. Acknowledgements	5
8. References	5
8.1. Normative References	5
8.2. Informative References	5
Appendix A. Release notes	5
A.1. Modifications between draft-petithuguenin-p2psip-proportional-quota-01 and draft-petithuguenin-p2psip-proportional-quota-00	5
A.2. Modifications between draft-petithuguenin-p2psip-proportional-quota-00 and draft-petithuguenin-vipr-reload-usage-00	6
A.3. Running Code Considerations	6
Authors' Addresses	6

1. Introduction

The base specification of RELOAD defines two variables to set the storage quota. The first one is the maximum size of an element of a specific kind, the second one is the maximum number of elements of a specific kind that can be stored on a specific peer. The combination of the two variables permits to limit the quantity of data that a peer have to store.

For kinds that are used with an access control policy that does not restrict the Resource-IDs, a different algorithm is needed to force storing nodes to behave. This document describes a quota algorithm that limits the number of elements of a specific kind that a node can store in the overlay, independently of the Resource-ID used. Another way to look at this quota algorithm is that an entity must provide a number of peers that is proportional to the number of elements of a specific kind that this entity wants to store.

It is important to understand that this quota works only for an overlay comprising only peers, so with a configuration file containing a <clients-permitted> element set to false.

2. Terminology

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 [RFC2119].

3. Quota Algorithm

A peer responsible for storing kinds using the quota algorithm described in this document MUST maintain a count of the number of unique entries being stored per signer for each kind. This operation does not require to add a field containing the Node-ID as the Node-ID of the signer is always available in the signature field of each element stored.

For example if a peer is storing 5 Resource-IDs and at each of those 5 there are two keys whose first 16 bytes correspond to a particular Node-ID, it means this node is currently storing 10 unique dictionary entries for that Node-ID.

When performing the quota checks for an entry, the peer starts by verifying that the size of the entry is consistent. It then takes the <max-count> configuration parameter for this overlay, which measures the amount of entries of a specific kind a particular node

can store when a `<max-count-per>SIGNER</max-count-per>` configuration parameter is also present. That value is multiplied by the number of replicas used by the topology plugin (i.e. 3 for Chord) and then divided by the fraction of the overlay owned by this peer. If the result is less than one, it is rounded up to two. This is the maximum number of unique entries that can be stored for this signer. If the peer is not yet storing this many entries for that Node-ID, the store is allowed.

Note that when evaluating a Store Request containing multiple entries per kind, the count of unique entries used for the evaluation is incremented after each successful check, but the count will be reset to its initial value if one of the check fails.

The algorithm takes in account only the duplications made by the topology plugin. If another level of duplication is done at the application level, the `<max-count>` value must be adjusted accordingly.

Note that because of imperfect distribution of Resource-IDs across the ring, new entries can be rejected even if the total count is under the limit. It is the responsibility of the storing entity to calculate the maximum acceptable probability of rejection and to increase the number of peers accordingly.

4. Overlay Configuration Document Extension

This document extends the overlay configuration document by defining a new element in the "urn:ietf:params:xml:ns:p2p:quota" namespace.

The `<max-count-per>` element changes the meaning of the `<max-count>` variable. The value "PEER" forces the `<max-count>` to be interpreted as been per storing peer, which is the default quota algorithm when this extension is not used. The value "SIGNER" forces the `<max-count>` to be interpreted as been per signer, which is the algorithm defined by this document.

The Compact Relax NG Grammar for this element is:

```
namespace pqt = "urn:ietf:params:xml:ns:p2p:quota"
```

```
kind-parameter &= element pqt:max-count-per { max-count-per-type }  
max-count-per-type | = "PEER"  
max-count-per-type | = "SIGNER"  
max-count-per-type | = xsd:string # extensions
```

5. Security Considerations

TBD

6. IANA Considerations

This document adds the following URN to the "XML Namespaces" class of the "IETF XML Registry":

urn:ietf:params:xml:ns:p2p:quota

7. Acknowledgements

This document was written with the xml2rfc tool described in [RFC2629].

8. References

8.1. Normative References

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

[I-D.ietf-p2psip-base]
Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol", draft-ietf-p2psip-base-15 (work in progress), May 2011.

8.2. Informative References

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

Appendix A. Release notes

This section must be removed before publication as an RFC.

- A.1. Modifications between
draft-petithuguenin-p2psip-proportional-quota-01 and
draft-petithuguenin-p2psip-proportional-quota-00

- o Changed "storing peer" to "signer" as this works also for replica peers.
- o The default quota algorithm is per storing peer, not per Resource-ID.
- o Changed the constants in the XML extension accordingly.
- o Added running code considerations for reference implementation.

A.2. Modifications between
draft-petithuguenin-p2psip-proportional-quota-00 and
draft-petithuguenin-vipr-reload-usage-00

- o Instead of having a StorageQuota parameter that gives the maximum number of entries, reused the max-count parameter (that is mandatory anyway) and changes its meaning.
- o Removed the 3x multiplier to account for the application layer COPY, as it is application specific.
- o Removed the additional 3x multiplier to compensate for imperfect distribution, and moved the responsibility to the storing nodes.

A.3. Running Code Considerations

- o Reference Implementation
(<http://debian.implementers.org/testing/source/reload.tar.gz>).
Marc Petit-Huguenin. Implements version -01.

Authors' Addresses

Jonathan Rosenberg
jdrosen.net
Monmouth, NJ
US

Email: jdrosen@jdrosen.net
URI: <http://www.jdrosen.net>

Cullen Jennings
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@cisco.com

Internet-Draft

Proportional Quota

July 2011

Marc Petit-Huguenin
Stonyfish

Email: marc@stonyfish.com

VIPR
Internet-Draft
Intended status: Standards Track
Expires: December 14, 2011

J. Rosenberg
jdrosen.net
C. Jennings
Cisco
M. Petit-Huguenin
Stonyfish
June 12, 2011

The Public Switched Telephone Network (PSTN) Validation Protocol (PVP)
draft-petithuguenin-vipr-pvp-01

Abstract

One of the main challenges in inter-domain federation of Session Initiation Protocol (SIP) calls is that many domains continue to utilize phone numbers, and not email-style SIP URI. Consequently, a mechanism is needed that enables secure mappings from phone numbers to domains. The main technical challenge in doing this securely is to verify that the domain in question truly is the "owner" of the phone number. This specification defines the PSTN Validation Protocol (PVP), which can be used by a domain to verify this ownership by means of a forward routability check in the PSTN.

Legal

This documents and the information contained therein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION THEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on December 14, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. The Wrong Way	6
3. EKE Protocols	12
4. Terminology	15
5. Protocol Overview	15
6. Username and Password Algorithms	16
7. Originating Node Procedures	18
7.1. Establishing a Connection	18
7.2. Constructing a Username and Password	18
7.2.1. Method A	18
7.2.2. Method B	22
7.3. Requesting Validation	23
8. Terminating Node Procedures	24
8.1. Waiting for SRP-TLS	24
8.2. Receiving Validation Requests	26
9. Syntax Details	27
10. Security Considerations	27
10.1. Entropy	27
10.2. Forward Routing Assumptions	28
11. IANA Considerations	28
12. Acknowledgements	28
13. References	28
13.1. Normative References	28
13.2. Informative References	29
Appendix A. Release notes	29
A.1. Modifications between vipr-01 and vipr-00	29
A.2. Modifications between vipr-00 and dispatch-03	29
A.3. Modifications between dispatch-03 and dispatch-02	29
Authors' Addresses	29

1. Introduction

The validation protocol is the key security mechanism in ViPR. It is used to couple together PSTN calls with IP destinations based on shared knowledge of a PSTN call. This document relies heavily on the concepts and terminology defined in [VIPR-OVERVIEW] and will not make sense if you have not read that document first.

The protocol assumes that two enterprises, the originating one (enterprise O) initiates a call on the PSTN to an E.164 number ECALLED that terminates on the terminating enterprise (enterprise T). Each enterprise has a ViPR server, acting as a P2P node. The node in enterprise O is PO, and the node in enterprise T is PT. This PSTN call completes successfully, and knowledge of this call is known to PO and PT. Later on, PO will query the P2P network with number ECALLED. It comes back with a Node-ID PCAND for a node. At this time, PO can't know for sure that PCAND is in fact PT. All it knows is that some node, PCAND, wrote an entry into the DHT claiming that it was the owner of number ECALLED. The objective of the protocol is for PO to determine that node PCAND can legitimately claim ownership of number ECALLED, by demonstrating knowledge of the previous PSTN call. It demonstrates that knowledge by demonstrating it knows the start time, stop timer, and possibly caller ID for the PSTN call made previously.

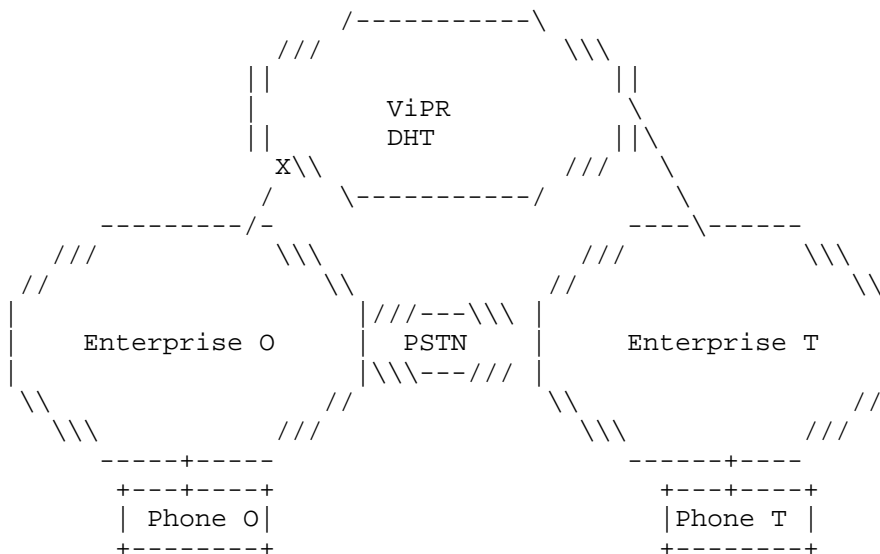


Figure 102: Validation Model

If node PCAND can demonstrate such knowledge, then enterprise O can assume that node PCAND had in fact received the call, which could only have happened if it had knowledge of the call to number ECALLED, which could only have happened if PCAND is in enterprise T, and thus it is PT. This is because PSTN routing is assumed to be "secure", in that, if someone calls some number through the PSTN, it will in fact reach a terminating line (whether it be analog, PRI, or other) which is the rightful "owner" of that number. If enterprise T was not the owner of the number, it would not have received the call, would not know its start/stop/caller ID, not be able to provide that information to PT, and not be able to satisfy the knowledge proof. This basic approach is shown in Figure 102.

A first question commonly asked is, why not just do regular authentication? What if we give each node a certificate, and then have the nodes authenticate each other? The answer is that a certificate certifies that a particular node belongs to a domain - for example, that node PT is part of example.com. A certificate does not assert that, not only is PT example.com, but example.com owns the following phone numbers. Therefore simple certificate authentication does not provide any guarantee over ownership of phone numbers.

In principle, it might be possible to ask certificate authorities, such as Verisign, to assert just that. However, traditionally, certificate authorities have been extremely hesitant to certify much at all. The reason is, the certifier needs to be able to assure that the information is correct. How can a certifier like Verisign verify that, in fact, a particular enterprise owns phone numbers? It could make a few test calls, perhaps, to check if they look right. However, these test calls are disruptive to users that own the numbers (since their phones will ring!). If the test calls are done for a subset of the numbers, it is not secure. If the certifier simply required, as part of the business agreement, that the enterprises provided correct information, the certifier might avoid legal liability, but the legitimacy of the service will be compromised and customers will stop using it. Furthermore, it has proven incredibly hard to do this kind of certification worldwide with a single certificate authority.

ViPR has, as a goal, to work anywhere in the world and do guarantee correct call routing with five nines of reliability. Consequently, traditional certificates and authentication do not work. It turns out to be quite hard to design a secure version of this validation protocol. To demonstrate this, we will walk through some initial attempts at it, and show how they fail.

2. The Wrong Way

The first attempt one might make is the following. PO takes the caller ID for the call, ECALLING and called number ECALLED for the call, and sends them to candidate node PCAND. These two identifiers - the called number E and the caller ID, form a unique handle that can be used to identify the call in question. Node PCAND looks at all of the ViPR Call Records (VCRs) of the calls over the last 48 hours, and takes those with the given called party number and calling party number. If there is more than one match, the most recent one is used. We now have a unique call.

Now, node PCAND demonstrates knowledge of this call by handing back the start and stop times for this call in a message back to PO. This approach is shown in Figure 103.

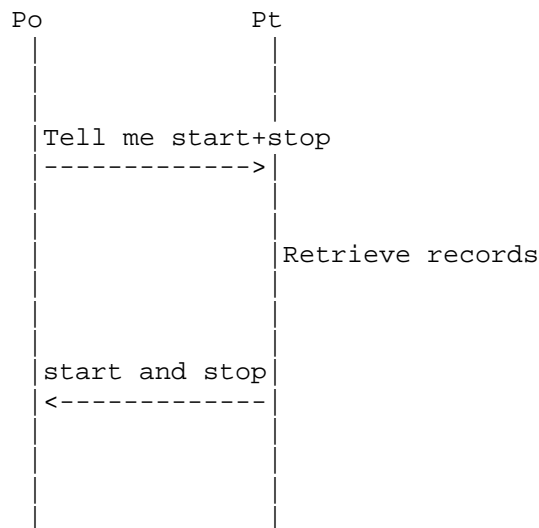


Figure 103: Incorrect Validation Protocol: Take 1

Unfortunately, this method has a major problem, shown in Figure 104.

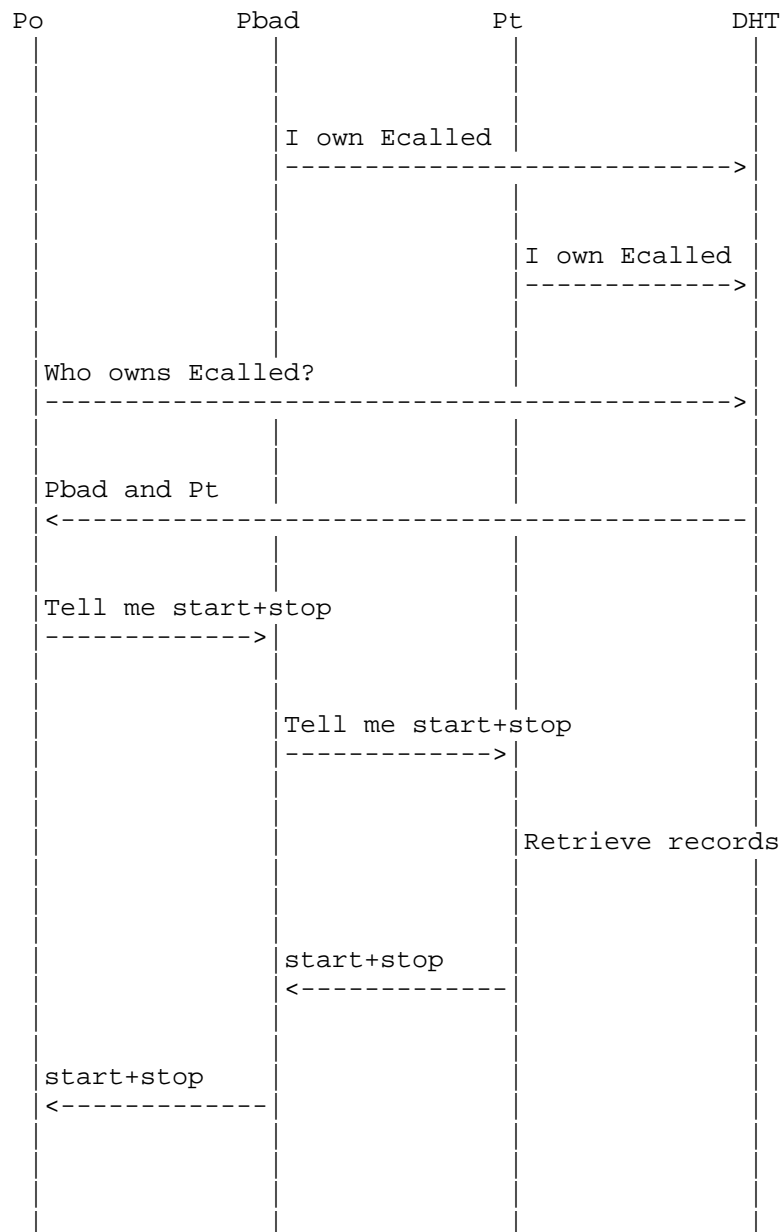


Figure 104: Attack for Incorrect Validation Protocol

Consider an attacker BadGuy PBAD. PBAD joins the P2P network, and advertises a number prefix they do NOT own, but which is owned by

enterprise T and node PT. Now, when PO queries the DHT with number ECALLED, it comes back with two results - the one from PBAD and the one from node PT. Details of querying the DHT are provided in [VIPR-RELOAD-USAGE]. It begins validation procedures with both. PBAD will now be asked to show the start and stop times for the call, given ECALLED and ECALLING. It doesn't know that information. However, node PT does. So now, PBAD, acting as if it were the originating party, begins the validation protocol with node PT. It passes the calling and called numbers sent by PO. PT finds a match and returns the call start and stop times to PBAD. PBAD, in turn, relays them back to PO. They are correct, and as a consequence, PO has just validated PBAD!

Typically, the first response to this is, "Well the problem is, you let two separate people write the same number into the DHT. Why don't you make sure on the right one is allowed to write it in?". That is not possible, since there is no mechanism by which an arbitrary node in the DHT can determine who is the rightful owner of this number. "OK", the reader responds, "So instead, why don't you define a rule that says, if there are two entries in the DHT for a particular number, consider this an attack and don't try to validate the number". That would prevent the attack above. However, it introduces a Denial of service attack. An attacker can pick a target number, write it into the DHT, and prevent successful validation from happening towards that number. They can't misroute calls, but they can stop ViPR from working for targeted numbers. That is not acceptable. ViPR has to be immune from attacks like this; it should not be possible, through simple means such as configuration, for an attacker to cause a targeted number to never be validated.

One might be tempted to add a signature over the call start and stop times, but it does not help. BadGuy can just resign them and relay them on.

In essence, this simple approach is like a login protocol where the client sends the password in the clear. Such mechanisms have serious security problems.

Realizing the similarities between the validation protocol and a login protocol, a next attempt would be to use a much more secure login mechanism - digest authentication. To do this, domain O takes the called number E and the caller ID, and send them to node P. Node P treats these as a "username" of sorts - an index to find a single matching call. The start time and stop times of the call become the "password". Enterprise O also sends a big random number - a nonce - to node P. Node P then takes the random number, takes the password, hashes them together, and sends back the hash. All of this is done over a TLS connection between enterprise O and node P. Digest over

TLS is very secure, so surely this must be secure too, right? Wrong!

It is not. Indeed it is susceptible to EXACTLY the same attack described previously. This is shown in Figure 105.

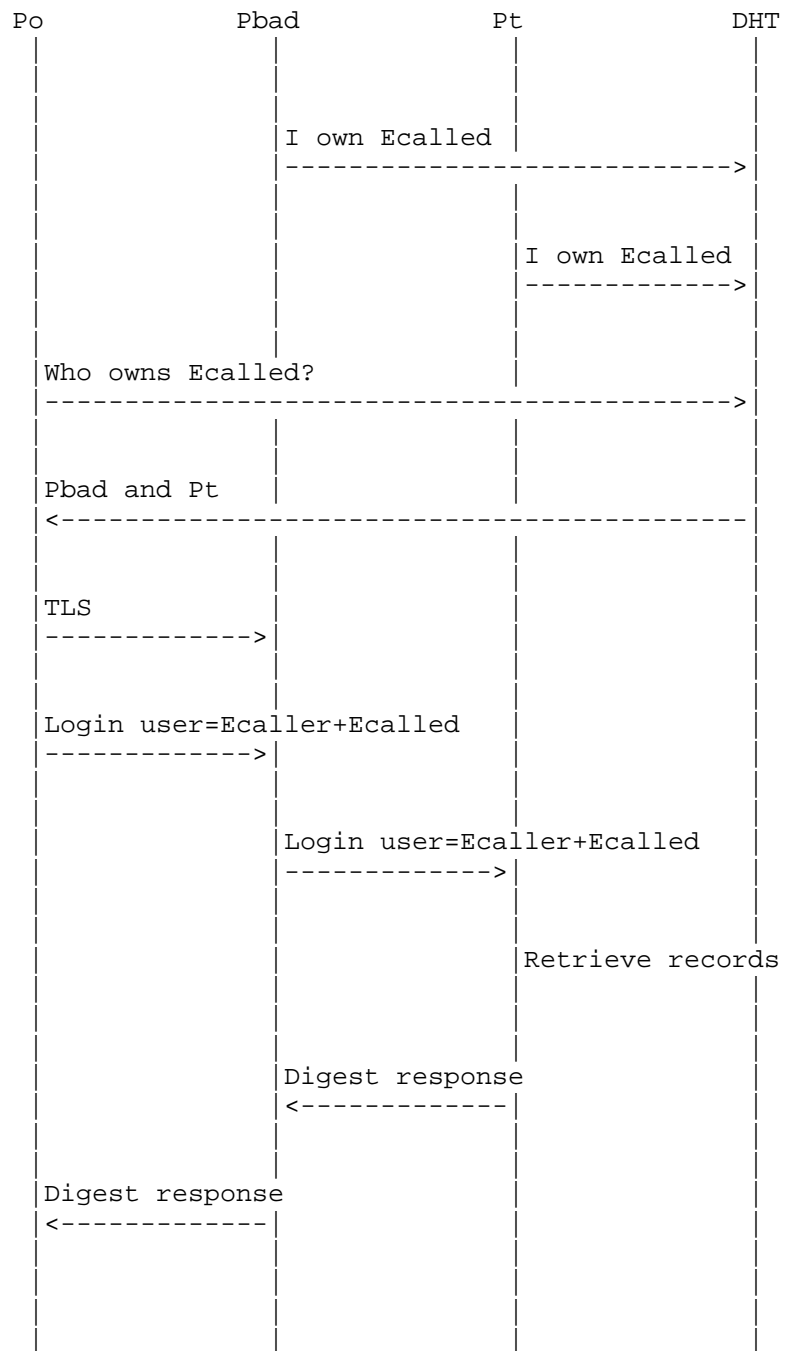


Figure 105: Trying Digest for Validation

In a similar attack, PBAD could pick a random called number it is interested in, query the P2P network for it, find node PT. Then, provide node PT the number ECALLED to attack, and ECALLING, assuming it can guess a likely caller ID. It then takes the received digest response, and goes through every possible start/stop time over the last 24 hours, running them through the hash function. When the hash produces a match, the PBAD has just found a full VCR for node PT. It can then write into the DHT using number E as a key, pointing to itself, and satisfy validation requests against it, without even needing to ask node P again. Our first attempt is susceptible to this attack too.

The problem here is that the call start and stop times have "low entropy" - they are not very random and are easily guessable, just like a poorly chosen password.

What we really want to do here is have a "login" protocol that creates a secure connection between a client and a server, where we use the called number and caller ID as a "username" to identify a PSTN call, and then use the start and stop times as a "password". But our login protocol has to have some key features:

1. Someone posing as a server, but which does not have the username and password, cannot determine the username and password easily as a consequence of an authentication operation started by a valid client, aside from successfully guessing in the one attempt it is given on each connection attempt.
2. Someone posing as a client, but which does not have the username and password, cannot determine the username and password as a consequence of an authentication operation started against a valid server, aside from guessing in the one attempt it is given on each TLS connection attempt.
3. An active MITM, who is explicitly on the path of the exchanges and has visibility and the ability to modify messages, cannot obtain the shared secret, nor can it observe or modify information passed between the client and real server.
4. It is impossible for a passive observer to view the exchange and obtain the shared secret or any of the material that is exchanged.
5. It is impossible for a rogue client or rogue server to participate in a login with a legitimate peer, and then take the messages exchanged, and run an offline dictionary attack to work through every possible combination of start and stop times. Fortunately, these properties are provided by a class of password authentication protocols called Encrypted Key Exchange or EKE protocols.

3. EKE Protocols

EKE protocols were proposed in 1992 by Steve Bellovin. Since their proposal, numerous variations have been defined. One of them, the Secure Remote Password protocol, was standardized by the IETF in RFC 2945 [RFC2945]. A TLS mode of SRP was later defined in RFC 5054 [RFC5054]. It is the latter protocol which is actually used by ViPR. A high level overview of EKE protocols is shown in Figure 106. Alice and Bob share a shared secret P. Alice generates a public/private keypair. She then takes her public key, and encrypts it using her password as a symmetric encryption key. She sends this encrypted key to Bob. Bob, who shares the password, uses it as a symmetric key and decrypts the message, obtaining Alice's new public key. Bob then constructs a big random number R, which is to be used as a session key. Bob then encrypts R with the public key he just got from Alice, and sends that to Alice. Now, Alice, using her public key, decrypts the message and obtains the session key R.

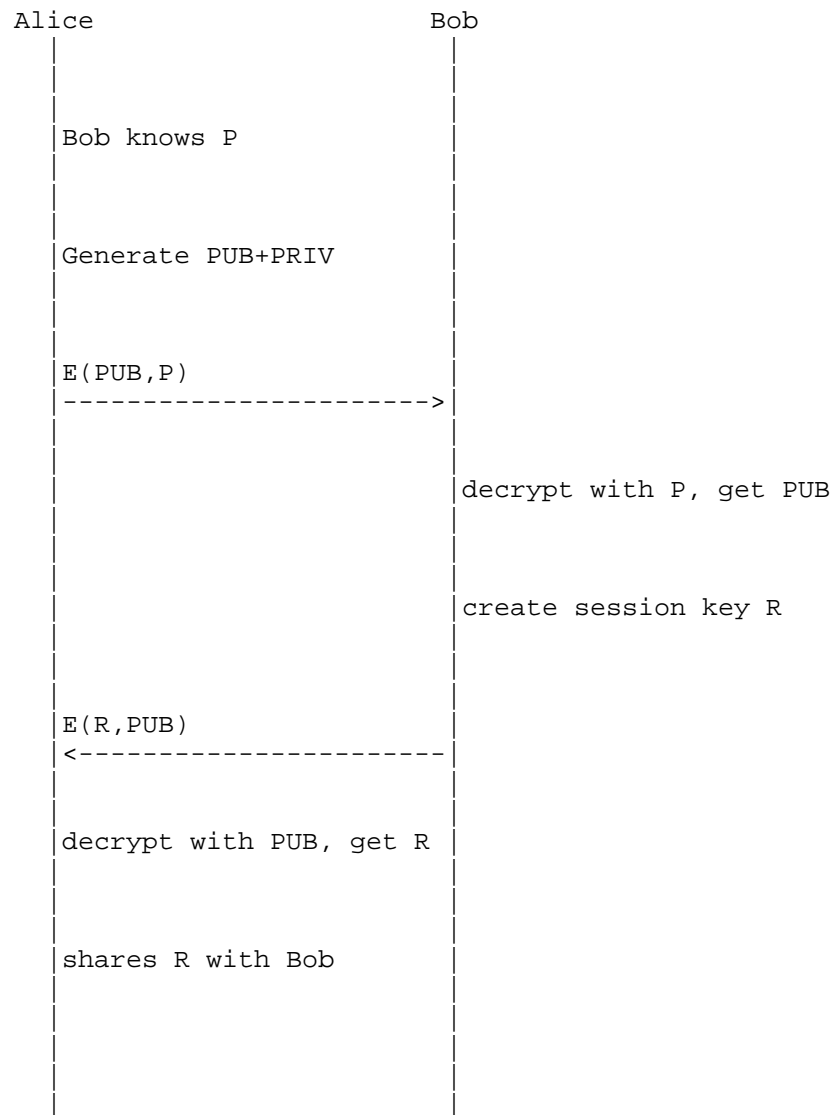
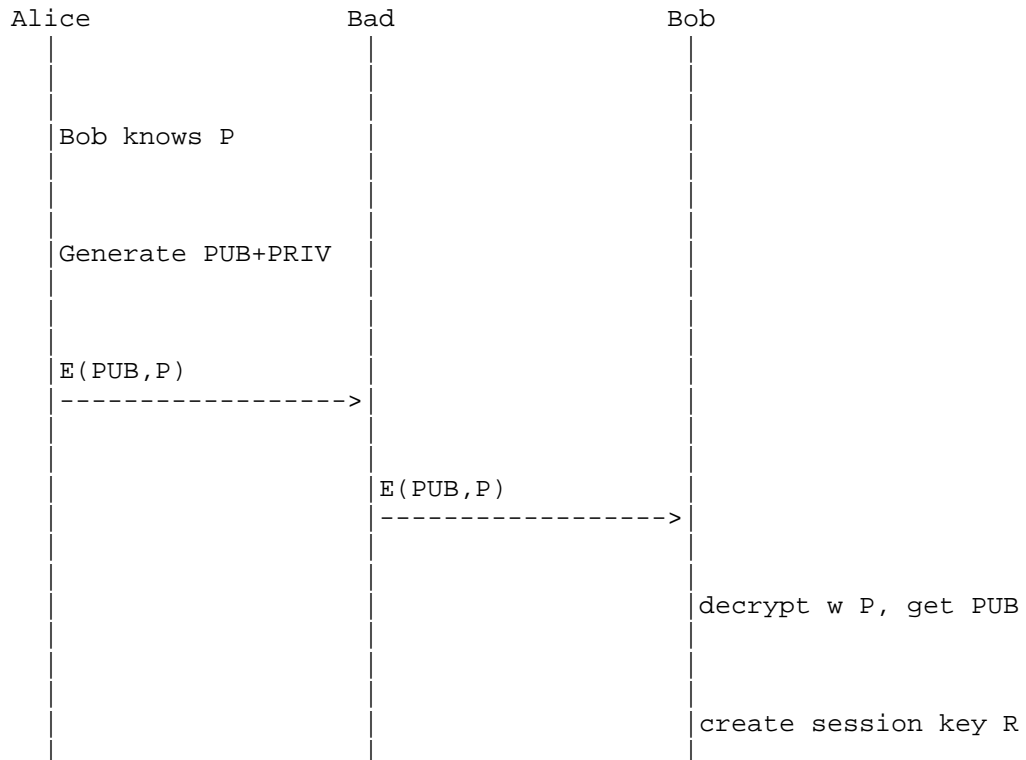


Figure 106: High Level EKE Model

At this point Alice and Bob share a session key R which can be used for authentication (by having Alice and Bob prove to each other that they have the same value for R) or for encrypting data back and forth. How does this help? Consider our man-in-the-middle attack again, in Figure 107. Once again, Alice shares a password with legitimate user Bob. However, she begins the "login" process with

BadGuy. She passes $E(PUB,P)$ to BadGuy. BadGuy doesn't know P , so he can't decrypt the message. More importantly, he can't run through each possible password P and decrypt the message. If he did, he wouldn't be able to tell if he got it right, since PUB appears random; the decryption process would produce a random string of bits whether it was successful or not. So for now, BadGuy can only pass it on. BadGuy now intercepts $E(R,PUB)$. Now, BadGuy can try the following. He can run through each P , decrypt $E(PUB,R)$, obtain PUB . However, since we are using asymmetric encryption (i.e., public key encryption), even with PUB he cannot DECRYPT $E(R,PUB)$! BadGuy does not have the private key, which he needs to decrypt. Given a public key, he cannot guess the private key either. That is how public/private keying systems work. That is the secret here to making this work. So, once again, BadGuy has no choice but to pass the message on. Now, Alice and Bob share R but it is unknown to BadGuy. Bob now takes his Node-ID, encrypts it with R , and sends to Alice. Once again, BadGuy doesn't have R and can't get it, so he has no choice but to pass it on. Alice decrypts this Node-ID with R , and now knows that she is actually talking to Bob - since she has Bob's Node-ID. Other data can be substituted for the Node-ID, and indeed this is what happens in the actual validation protocol.



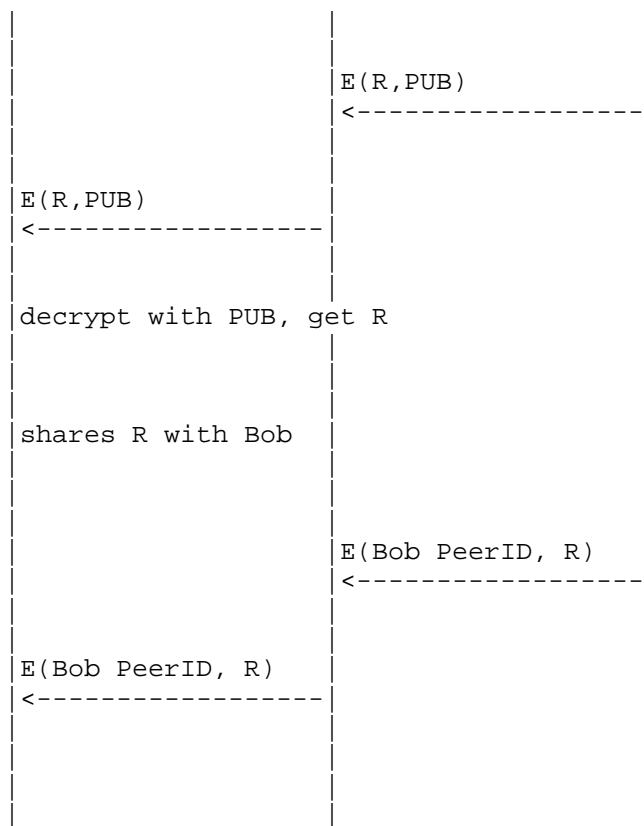


Figure 107: Attacking EKE Protocols

However, the main point of this exercise is to demonstrate that EKE protocols have the desired properties.

4. Terminology

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 [RFC2119].

5. Protocol Overview

The validation protocol begins with the following assumptions:

1. Node PO wishes to validate with node PCAND, and has its Node-ID (which it obtained via the DHT) and VServiceID (which it also obtained via the DHT Fetch).
2. Node PO and PCAND have a series of call records over the last 48 hours, uploaded by their call agents. Each call record contains an E.164 calling and called party number, and a start and stop time in NTP time. On the terminating side, each call record is also associated with a VServiceID.
3. Node PO is seeking to validate a call to called number ECALLED with caller ID ECALLING.

The validation protocol operates by having the originating node make a series of attempts to connect to, and "login" to the terminating node. Each "login" attempt consists of establishment of a TCP connection, and then execution of TLS-SRP procedures over that connection. TLS-SRP[RFC5054] relies on a shared secret - in the form of a username and password - in order to secure the connection. In ViPR, the username and password are constructed by using information from a target VCR along with the VServiceID learned from the DHT. The "username", instead of identifying a user, identifies a (hopefully) unique VCR shared between the originating and terminating nodes. The "password" is constructed from the VCR such that its knowledge of the information is unique to knowledge of the VCR itself.

Unfortunately, it is difficult to construct usernames and passwords that always uniquely identify a VCR. To deal with this, the validation protocol requires the originator to construct a series of usernames and passwords against a series of different nodes and their corresponding IP addresses and ports, and then run through them until a connection is securely established.

6. Username and Password Algorithms

ViPR provides two different algorithms for mapping from a particular VCR to a username and password:

1. Method A: This method makes use of the called party and calling party number to form the username, and the start and stop times of the call to form the password.
2. Method B: This method makes use of the called party number, along with a point in time in the middle of the call, as the username, and then the start and stop times to form the password.

The originating node will first try validations with method A, and if those all fail, then try with method B. The method itself, along with necessary information on how to use the method, is encoded into the

username itself. The format of the username is (using ABNF [RFC4234]):

```
username = method-a / method-b / future-method
future-method = method ":" method-data
method = 1ALPHA
method-data = 1*(alphanum / method-unreserved)

method-a = "a:" vserviceid originating-number terminating-number
           rounding-time
method-b = "b:" vserviceid terminating-number timekey rounding-time

vserviceid = "vs=" 1*32HEXDIG ";"
originating-number = "op=+" 1*15DIGIT ";"
terminating-number = "tp=+" 1*15DIGIT ";"
timekey = "tk=" 1*10DIGIT "." 1*10DIGIT ";"
rounding-time = "r=" 1*6DIGIT ";" ; Cannot be equal to 0
```

This format starts with the method, followed by a colon, followed by a sequence of characters that are specific to the method. Both methods a and b rely on conveyance of information attributes that make up the username. Each attribute follows a specific format.

Examples include:

```
a:vs=7f5a8630b6365bf2;op=+17325552496;tp=+14085553084;r=1000;
b:vs=7f5a8630b6365bf2;tp=+14085553084;tk=172636364.133622;r=1000;
```

Both methods use a rounding factor R. This is used to round the start and stop times in the password to a specific nearest multiple of R (which is in milliseconds). This rounding is done because the passwords need to be bit exact and we need to compensate for different measured values.

If we will fallback to method B (which works more often), why have both? There are two answers:

1. The caller ID mechanism (method A) will work, and the non-caller ID (method B) won't work, for numbers like 8xx.
2. Method A has much higher entropy (see analysis in Section 10.1). Validating with it provides greater confidence in the validity of the number. In this phase, nothing is done with this "confidence". However, in later phases, it is anticipated that low-confidence numbers will require multiple validations for different calls to occur before they are trusted. To allow for this feature to be added later, validation with both methods must be present in the initial release.

The sections below detail precisely how these are constructed.

7. Originating Node Procedures

Most of the work for validation is on the side of the originator. It establishes connections and performs a series of validation checks.

7.1. Establishing a Connection

The first step in the process is to establish a TCP connection to PCAND. To do that, PO sends a RELOAD AppAttach message targeted towards PCAND, using the Application-ID defined in [VIPR-RELOAD-USAGE]. Once connected, TLS-SRP is run over the connection.

7.2. Constructing a Username and Password

When a terminating node receives a username in a format it doesn't understand, it fails the validation. This allows for graceful upgrade to new mechanisms in the future.

7.2.1. Method A

The PO examines the VCR it is using for validation. It extracts the calling and called party numbers, both of which are E.164 based. This VCR will have been uploaded at a previous point in time. PO then examines the VCRs posted in the time since this one was uploaded, and looks for any more recent VCRs with the same calling and called party numbers regardless of VService. If it finds one or more, it takes the most recent one (as measured by its end time). If it finds no more recent, it uses the VCR which triggered the validation in the first place.

Why do this? This deals with the following case. User A calls user B, causing a VCR to be uploaded. The originating node sets a timer, which fires 12 hours later. However, within that 12 hour period, A called B again. If node A provides the caller ID and called party numbers as the "key" to select a VCR, it will match multiple records over the past day. We need to pick one, so the most recent is always used. This requires the originating node to know and use the most recent VCR. Furthermore, we must choose the most recent VCR regardless of its VService, because the originating Upload VCRs are sent using an arbitrary VService. Thus, the more recent call may have been done using a different VService than the one which triggered the validation. Since the actual Vservices are not common between originating and terminating sides, we must choose the VCR on the originating side regardless of VService. The username is

constructed by using the syntax for method A described above. The calling party number is set to "op", and the called party number is set to "tp", and "r" is the value of Tr as an integral number of milliseconds. The VServiceID learned from the dictionary entry is used as the value of "vs".

This username will select the identical VCR at the terminating node, under the following conditions:

1. PT is aware of all calls made to the called party number. This property is true so long as each incoming number is handled by a single call agent within a domain, and furthermore, the VCR for calls to that number is always posted to a ViPR server which advertises that number into the DHT. These properties are readily met by ViPR for typical single user numbers. For 8xx numbers, which are translated within the PSTN and may route to a multiplicity of non-8xx numbers, it is more difficult. ViPR will only work with 8xx numbers if all calls to those numbers get sent to agents which share the same ViPR server.
2. PO is aware of all calls made to the called party number with the given caller ID. This property can be hard to meet. If the caller ID for a call is set to the number of the calling phone, and all VCRs made from that phone are posted to the same ViPR server, that server will know about all calls made by the domain with the given DID in the caller ID. However, in domains that set the PSTN caller ID to the attendant line number, it is possible that there would be two separate agents, each utilizing different ViPR servers. A user in each agent calls the same number, and the same PSTN caller ID is used. However, one ViPR server knows about one of the calls, and a different ViPR server knows about the other call. However, PT knows about both. In that case, validation from one of the ViPR servers will fail, and from the other, succeed.
3. There were no calls on the PSTN to the called party which spoofed the caller ID to match the caller ID used by the valid enterprise. In that case, PT will have a VCR for a call with a matching calling/called party number, but this VCR is unknown to PO since the call was not actually made by the originating enterprise. This attack is described in more detail in XXXX.

Next, the password is selected. The password is basically the start and stop times for the call. However, the SRP protocol requires a bit exact agreement on the password. Unfortunately, the calling and called parties will not have the same values for the start and stop times, for several reasons:

1. The call start time at the originating and terminating ends will differ by the propagation delay of the call acceptance message through the PSTN. This can be several hundreds of milliseconds.
2. The clocks at the originating and terminating ends may not be synchronized, which can also introduce different values for the start and stop times.
3. The call termination time at the originating and terminating ends will also differ by the propagation time; this propagation time may in fact be different for the call acceptance and termination.

It is also important to note that agreement on a call acceptance and termination time assumes an explicit signaling message is sent for these two events. In the case of analog FXO ports, there is no signaling at all, and consequently, these points in time cannot be measured. It is possible to agree upon other call characteristics when analog lines are in use, but they have much worse accuracy and consequently much, much lower entropy. For this reason, this specification of ViPR only works in telephony systems with explicit messaging for call acceptance and termination, which includes PRI, SS7, BRI, analog trunks with answer and disconnect supervision, and CAS trunks.

To deal with these inaccuracies in timing, the start and stop times need to be rounded. Let Tr be the rounding interval, so that each time is rounded to the value of $N*Tr$ for integral N , such that $N*Tr$ is less than the start or stop time, and $(N+1)*Tr$ is greater than it. In other words, "round down". If $Tr=1$ second, this would round down to the nearest second.

Unfortunately, rounding doesn't fully help. Lets say that the difference between the start times on the originating and terminating nodes is δ . We can still have different values for the start time if one side rounds to one value, and the other side to a different value. If $\delta=100\text{ms}$ and $Tr=1\text{s}$, consider a start time of 10.08 seconds on one side, and 9.98 on the other side. One side will round to 10 seconds and the other to nine seconds. The probability of this happening is approximately δ/Tr . We could just make Tr really large to compensate, but this reduces the entropy of the system (see below).

To deal with this, the originating node will actually compute FOUR different passwords. For the start time and stop time both, the originating node will round down as follows. Let T be the time in question. Let N be the value such that $N*Tr \leq T < (N+1)*Tr$. In other words, $N*Tr$ is the nearest round-down value, and $(N+1)*Tr$ is the nearest round up. Let $T1$ and $T2$ be the two rounded values of T . We have:

```

if (T >= ((2N+1)/2) * Tr) then:
    T1 = N*Tr
    T2 = (N+1)*Tr
if (T < ((2N+1)/2) * Tr) then:
    T1 = N*Tr
    T2 = (N-1)*Tr

```

In other words, if T is in the top half of the rounding interval, we try the rounded values above and below. If T is in the bottom half, we try the rounded values below, and below again. Pictorially:

[[TBD]]

Figure 108: Rounding mechanism for validation protocol

These are tried in the following sequence:

1. Try Tstart-1 and Tend-1.
2. Try Tstart-2 and Tend-1.
3. Try Tstart-1 and Tend-2.
4. Try Tstart-2 and Tend-2.

For example, if the originating side has a start time of 10.08 and a stop time of 30.87, the four start and stop times with Tr=1s are:

Start	Stop
10	30
9	30
10	31
9	31

Each of these times is represented in 64 bit NTP time (Tr can be configured to less than 1s in which case there will be non-zero values in the least significant 32 bits). Each password is then computed by taking the 64 bit start time, followed by the 64 bit end time, resulting in a 128 bit word. This word is base64 encoded to produce an ASCII string representation of 21 characters. To perform the caller ID based validation, the SRP-TLS procedure is done four times, once with each of the four username/password combinations (of course the username is identical in all four cases). As long as delta is less than Tr/2, one of this is guaranteed to work.

7.2.2. Method B

Unfortunately, in many cases caller ID cannot be used as an identifier for the VCR. This is because:

1. CallerID is frequently suppressed in the PSTN, and not delivered. This is especially true in international cases.
2. CallerID is sometimes munged by the PSTN, and delivered, but with a different value than was sent by the originator. This happens in certain arbitrage interexchange carriers.

Consequently, if no caller ID was delivered at all, the terminating side will not have a matching record. In that case, it informs the calling side that it should abort and revert to method B. If munged, it will also abort for the same reason.

If the caller ID attempt aborts, PO now tries a different approach. In this approach, the "username" is the combination of the called party number and a point during the call, selected at random. The password is equal to the start and stop times of the call. This method uses the method-tag "B" in the username.

Unlike method A, with method B, the VCR which triggered the validation is used, regardless of whether there were other, more recent, calls to the same calledparty number! This is because, in method B (unlike method A), the time itself is used as a key to select a VCR. Furthermore, using a more recent VCR does not interact properly with multi-tenancy. The called number and point during that call will select an identical VCR on the terminating side if the following conditions are met:

1. For the called party number, there was not more than one call in progress made to that number at the same time. This is generally true for numbers for a single user; typically there is only one active call at a time. Of course, it is possible a user receives a call, and then gets another. It then puts the first on hold while the second call is taken. In these cases, it is possible that the "username" will select a different VCR on PT, in which case the validation fails. More troubling are numbers representing call centers, conference bridges, 8xx numbers, and attendant numbers, all of which frequently have multiple calls in progress to them at the same time. As a consequence, for these types of called numbers, validation is typically only going to work if caller ID is delivered. Fortunately, 8xx numbers are only national in the first place, so it is likely that this will work.

2. PO is aware of all calls made from within its enterprise to ECALLED. This can fail if there are multiple ViPR servers serving different agents, and a call is made from one agent, sent to one ViPR server, and a call to that same number is made on a different agent, sent to a different ViPR server. As in the caller ID case, this will still be OK in many cases - the validation from one ViPR server succeeds, the other fails.
3. PT is aware of all calls made to ECALLED. The same caveats as described above for the caller ID mechanism apply. PO takes the VCR, and chooses a time Tkey which is uniformly distributed between Tstart+Tr and Tstop-Tr. The usage of the Tr here is to make sure that Tkey is squarely inside of the call start and stop for PT as well. Note that, because Tkey is not a password, it is sent in the clear and does not need to be rounded.

The username encodes the called party number, Tkey, the DHT, and the VServiceID learned from the DHT query. The password is computed identically to method A.

7.3. Requesting Validation

Once the SRP-TLS connection is up, data is exchanged. This is done through a single VAP transaction initiated by PO. This transaction is only VAP in the sense that it utilizes the basic syntax (the header and TLV attribute structure), and its request/response model. Other than that, it is effectively a different protocol - the validation protocol.

PO sends a VAP request with method ValExchange (0x00d). It contains one attribute, Domain. The originating ViPR server obtains this domain by looking at the VService of the VCR that was eventually used for the validation. Note that, in cases where the VCR which triggered the validation, is different than the one actually used for validation (because a more recent VCR to the same number was found), it is important to use the VService associated with the VCR which was actually used for validation, and NOT the VService associated with the VCR which triggered the attempt. Multi-tenancy does not work properly without this. The domain from the VService is placed into the message. This is basically the domain name of the originating enterprise. It is included since it is needed by PT to compute the ticket.

PO will then receive a response. If it never receives a response within a timeout, it considers the validation to have failed, and continues to the next choice. If it receives any kind of error response, including a rejection due to a blacklisted domain, it considers validation to have failed, and continues to the next choice. If it is a success response, it will contain one attribute -

ServiceContent, which contains a ValInfo XML object. ValInfo is an XML object which contains the SIP URIs and the ticket. The ViPR server must parse the ValInfo XML object and perform verification on it to avoid attacks. The following checks are done:

1. Extract the <number> element. This will contain a single number. That value is compared with the E.164 called party number which was just validated. If they do not match, this is a potential attack, and the XML is discarded and the ViPR server acts as if validation failed. However it does not generate an alarm.
2. Remove any extensions to the XML which are not supported by the ViPR server (no extensions defined, so in this version, any elements except for the <ticket>, <number>, <route>s and their embedded <SIPURI> are removed.
3. Verify that the <route> element contains a single element, <SIPURI>.
4. Verify that the SIP URI is not larger than 614 characters, contains a domain name that is a valid set of domain name characters, contains a user part that is a valid set of characters, if it contains maddr, that the maddr is a valid domain or IP and less than 255 characters, and if there is a port, it is within 0-65535. This is for security purposes; to make sure a malicious ViPR server on the terminating side cannot send invalid URI and attack the call agent.
5. Verify that each SIP URI contains the same domain name. Once the checks and fixes are done, the patched XML is passed to subscribers in a Notify as described in [VIPR-VAP].

8. Terminating Node Procedures

8.1. Waiting for SRP-TLS

PT will wait for an AppAttach request on the Application-ID defined in [VIPR-RELOAD-USAGE] and the connection is established, it begins waiting for SRP-TLS. The TLS messaging will provide PT with a username.

It parses the username and determines the method. If the value of the method is not "a" or "b", this is a new method not supported by the node. The SRP-TLS procedures should be failed. If the method is "a", it is the caller ID mechanism. The called number, calling number, VService, and rounding time are extracted. PT then searches through its VCRs over the last 48 hours for one with a matching called number and caller ID and VService whose VServiceID matches the one from the username:

1. If none are found, PT proceeds with the SRP-TLS exchange, but using a fake username and password. This will cause the validation to eventually fail.
2. If one is found, it is used.
3. If more than one is found, the one with the most recent end time is used.

The start and stop times from the selected VCR are taken. Using the value of *Tr* from the username, both times are rounded down to the nearest multiple of *Tr*. Note that, this rounding is different than the one used on the originating side. The values are ALWAYS rounded down. So if the stop time is 10.99 and *Tr* is one second, the rounded down value of 10 is used. The start and stop times are then represented as 64 bit NTP times (after rounding), concatenated, and base64ed to produce a 21 character password. This is the password used with SRP-TLS.

Note that, the originating node will try up to four different password combinations. One of these should work, the others will cause SRP-TLS to fail due to differing shared secrets. However, it is the job of the originator to perform these four; to the terminating node, they are four separate attempts. Processing of SRP-TLS login attempts is stateless on the terminating side. This means that each attempt is treated independently by PT. It performs identical processing on each SRP-TLS attempt - examine the username, find a matching VCR, extract password, and fail the attempt or continue to success. The originating side has the main burden of sequencing through the various mechanisms.

If the method is "b", PT uses the extracted called party ID and a time in the middle of the call. It searches through all VCR records whose called number matches and whose VServiceID matches, and of those, takes the ones where *Tkey* is between *Tstart* and *Tstop*. Of those, if more than one match, the one with the most recent *Tstop* is used. *Tstart* and *Tstop* for that VCR are extracted, and converted to a password just as is done for the PO. The resulting SRP-TLS procedure will then either succeed or fail. Note that, if a domain has multiple Vservices that contain the same number, there will be multiple VCRs for calls to that number, and there will be multiple validation attempts, one for each of the Vservices.

Note that there could be multiple successful validations coming from different domains for one specific VCR, so VCRs should not be removed before the end of the 48 hours period. This can happen when a calling domain uses a PSTN provider that is itself ViPR enabled.

8.2. Receiving Validation Requests

PT listens for incoming VAP/validation requests once the TLS connection is up. It rejects anything but a ValExchange method with a 400 response. This allows for future extensibility of the validation protocol. If the request was ValExchange, it extracts the domain name. This will be something like "example.com". PT knows the VCR against which validation succeeded. That VCR is associated with a VService. The ViPR server checks the domain in the ValExchange request against the black/white list associated with that VService. If no VService is currently active, the ValExchange is rejected with a 403. If there is one active, and if the domain appears on the black list, or does not appear in the white list, the ViPR server rejects the ValExchange request with a 403 error response, indicating that this domain is not allowed to call.

If the domain was in the whitelist or not in the blacklist, or there was no whitelist/blacklist, PT constructs a successful response to the ValExchange request. It contains one attribute: ServiceContent. It has a ValInfo XML object, which contains a number, a ticket, and a series of routes.

The number is always the E.164 number which was just validated, including the plus sign. Note that this will also appear in the ticket. The route element is the sequence of route elements for each instance associated with the vservice.

Details of the ticker are provided in [VIPR-SIP-ANTISPAM] but the ticket attribute is constructed as follows:

1. A ticket unique ID TLV is created, containing a randomly chosen 128 bit value as the ticket ID. That is the first TLV in the ticket.
2. A salt TLV is created, containing a random 32 bit value. This is the second TLV in the ticket.
3. The validity has the start time set using the current time as the start time, and the current time + the ticket lifetime as the end time. The ticket lifetime is a per-DHT configurable parameter. The terminating ViPR server will have performed the validation using a particular VService; the DHT for that VService is used to find the right value for this parameter.
4. Number: This is the terminating number, in E.164 format, which was just validated.
5. Granting node: this is set to one of the Node-IDs associated with this ViPR server. Any will do.
6. Granting domain: This value is taken from the domain part of the SIP URI associated with the VService in which the validated VCR was found.

7. Granted-To domain: This is formed using the Domain sent in the ValExchange request.
8. Epoch: This is the current epoch associated with the password.
9. Integrity: Using the current password, this is computed from the rest of the Ticket.

The resulting sequence of TLVs is base64 encoded and that is placed into the ticket element in the ServiceContent attribute in the ValExchange response.

9. Syntax Details

This section enumerates the methods and attributes used by PVP.

The methods and their corresponding method values, are:

Method	Value
-----	-----
ValExchange	0x00d

Figure 1: PVP Methods

The attribute names and corresponding types are:

Attribute Name	Type
-----	----
Domain	0x3001

Figure 2: PVP Attributes

10. Security Considerations

[[This section is mostly missing and needs to be done.]]

10.1. Entropy

[[The entropy obtained in the information from the PSTN calls significantly impacts the security of this protocol. This section needs to provide an analysis of how much entropy actually exists in this information.]]

[[Defines the worst case of conference calls and resulting entropy]]

[[Describe the idea of doing multiple validations to aggregate entropy]]

10.2. Forward Routing Assumptions

[[Discuss forward routing security in PSTN and explain how this protocol is reliant on that.]]

11. IANA Considerations

[[TBD Define ports used.]]

12. Acknowledgements

Thanks to Patrice Bruno for his comments, suggestions and questions that helped to improve this document.

13. References

13.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.
- [RFC5054] Taylor, D., Wu, T., Mavrogiannopoulos, N., and T. Perrin, "Using the Secure Remote Password (SRP) Protocol for TLS Authentication", RFC 5054, November 2007.
- [VIPR-OVERVIEW] Jennings, C., Rosenberg, J., and M. Petit-Huguenin, "Verification Involving PSTN Reachability: Requirements and Architecture Overview", draft-jennings-vipr-overview-00 (work in progress), April 2011.
- [VIPR-VAP] Jennings, C., Rosenberg, J., and M. Petit-Huguenin, "Verification Involving PSTN Reachability: The ViPR Access Protocol (VAP)", draft-jennings-vipr-vap-00 (work in progress), April 2011.
- [VIPR-SIP-ANTISPAM] Rosenberg, J., Jennings, C., and M. Petit-Huguenin, "Session Initiation Protocol (SIP) Extensions for Blocking VoIP Spam Using PSTN Validation",

draft-petithuguenin-vipr-sip-antispam-01 (work in progress), June 2011.

[VIPR-RELOAD-USAGE]

Rosenberg, J., Jennings, C., and M. Petit-Huguenin, "A Usage of Resource Location and Discovery (RELOAD) for Public Switched Telephone Network (PSTN) Verification", draft-petithuguenin-vipr-reload-usage-01 (work in progress), June 2011.

13.2. Informative References

[RFC2945] Wu, T., "The SRP Authentication and Key Exchange System", RFC 2945, September 2000.

Appendix A. Release notes

This section must be removed before publication as an RFC.

A.1. Modifications between vipr-01 and vipr-00

- o Added text explaining that VCRs should not be removed before the end of the 48 hours delay.
- o Inserted Terminology section.
- o Fixed the timekey ABNF.
- o Specified that rounding-time cannot be equal to 0.

A.2. Modifications between vipr-00 and dispatch-03

- o Moved to new Working Group.

A.3. Modifications between dispatch-03 and dispatch-02

- o Nits.
- o Shorter I-Ds references.
- o Removed sentence saying that Tkey is converted to base64.
- o Added ValExchange method and Domain attribute definitions.
- o Fixed the last sentence of 7.2 - the ticket goes into the ticket element in the ServiceContent attribute.
- o Expanded first usage of VCR initialism.
- o Replaced any instance of peerID by Node-ID.
- o Rewrote the ABNF.

Authors' Addresses

Jonathan Rosenberg
jdrosen.net
Monmouth, NJ
US

Email: jdrosen@jdrosen.net
URI: <http://www.jdrosen.net>

Cullen Jennings
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@cisco.com

Marc Petit-Huguenin
Stonyfish

Email: marc@stonyfish.com

VIPR
Internet-Draft
Intended status: Standards Track
Expires: January 11, 2012

J. Rosenberg
jdrosen.net
C. Jennings
Cisco
M. Petit-Huguenin
Stonyfish
July 10, 2011

A Usage of Resource Location and Discovery (RELOAD) for Public Switched
Telephone Network (PSTN) Verification
draft-petithuguenin-vipr-reload-usage-02

Abstract

Verification Involving PSTN Reachability (VIPR) is a technique for inter-domain SIP federation. VIPR makes use of the RELOAD protocol to store unverified mappings from phone numbers to RELOAD nodes, with whom a validation process can be run. This document defines the usage of RELOAD for this purpose.

Legal

This documents and the information contained therein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION THEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on January 11, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Registering an E.164 number	3
4. Fetching a registration	4
5. VIPR-REGISTRATION Kind Definition	5
6. Overlay Configuration Document Extension	6
7. Security Considerations	6
8. IANA Considerations	6
8.1. Access Control Policies	6
8.2. Application-ID	7
8.3. Data Kind-ID	7
8.4. IETF XML Namespaces Registry	7
9. References	7
9.1. Normative References	7
9.2. Informative References	8
Appendix A. Examples	8
Appendix B. Release notes	9
B.1. Modifications between vipr-02 and vipr-01	9
B.2. Modifications between vipr-01 and vipr-00	9
B.3. Modifications between vipr-00 and dispatch-03	10
B.4. Modifications between dispatch-03 and dispatch-02	10
B.5. Running Code Considerations	10
Authors' Addresses	10

1. Introduction

This document relies heavily on the concepts and terminology defined in [VIPR-OVERVIEW] and will not make sense if you have not read that document first. As it defines a usage for RELOAD [P2PSIP-BASE], it assumes the reader is also familiar with that specification.

2. Terminology

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 [RFC2119].

3. Registering an E.164 number

To register an E.164 number a VIPR server stores a `ViprRegistration` structure using the fully qualified E.164 based number without any non-digit characters but the '+' character as Resource Name. For redundancy purpose, the VIPR server MUST store the same `ViprRegistration` structure two more times by using the same Resource Name prepended with the "COPY1" and "COPY2" character string respectively.

The contents of the `ViprRegistration` structure are as follow:

```
enum { reserved(0), node_id(8200), (65535) } ViprRegistrationType;

struct {
    select (ViprRegistrationType) {
        case node_id:
            NodeId          pvp_provider;

            /* This type can be extended */
    } ViprRegistrationData;

struct {
    ViprRegistrationType  type;
    uint16                length;
    ViprRegistrationData  data;
} ViprRegistration;
```

The `ViprRegistration` structure contains the following values:

type

The type of the registration

length

The length of the data structure, i.e. not counting the type and length fields

pvp_provider

The Node-ID of the peer to which an AppAttach request should be sent to initiate the PVP protocol

The Node-ID used in the pvp_provider field and in the key MUST be ready to process AppAttach requests for Application-ID 8470 at the time the registration is done.

VIPR supports multiple registrations for a single E.164 number by using a Dictionary Data Model. The dictionary key is the concatenation of the Node-ID and the VServiceId, resulting in a 24 bytes long value. Using the Node-ID of the node performing the store segments the keyspace of the dictionary so that no two peers ever store using the same key. Using the VService allows for a single VIPR server to service multiple clusters, and to ensure that numbers published by one cluster (using one VServiceID) do not clobber or step on numbers published by another cluster (using a different VServiceID).

The Store operations are paced into the overlay at a fixed rate. The VIPR server maintains a queue that is filled with store requests. The VIPR server services that queue at a fixed, provisioned rate, which is stored in a kind configuration variable named <store-rate-limit>.

4. Fetching a registration

A VIPR server wishing to validate a E.164 number will start 3 Fetch transactions using respectively the fully qualified E.164 based number without any non-digit characters but the '+' character as Resource Name, the same Resource Name prepended with the "COPY1" character string and finally the same Resource Name prepended with the "COPY2" character string.

The VIPR server will then inspect the elements in the 3 dictionary returned and will keep only the registrations that have the same key in at least 2 of the 3 dictionary returned. For each registration kept, the VIPR server will fetch the certificates associated with the Node-ID in the key using the CERTIFICATE_BY_NODE usage and will

verify that the signature of the registration is valid.

The VIPR server can then send an AppAttach to the Node-ID found in the key and registration, using the Application-ID 8470. After the connection is established, the VIPR server can start PVP as specified in [VIPR-PVP].

5. VIPR-REGISTRATION Kind Definition

Name VIPR-REGISTRATION

Kind Ids The Resource Name for the VIPR-REGISTRATION Kind-ID is a fully qualified E.164 based number without any non-digit characters but the '+' character, prepended by either an empty character string, the "COPY1" character string or the "COPY2" character string. The data stored is a ViprRegistration, which contains the Node-ID of the peer to which an AppAttach request should be sent to initiate the PVP protocol.

Data Model The data model for the VIPR Kind-ID is dictionary. The dictionary key is the concatenation of the Node-ID and the VServiceId, resulting in a 24 bytes long value. Using the Node-ID of the node performing the store segments the keyspace of the dictionary so that no two peers ever store using the same key and using the VService allows for a single VIPR server to service multiple clusters, and to ensure that numbers published by one cluster (using one VServiceID) do not clobber or step on numbers published by another cluster (using a different VServiceID).

Access Control The VIPR-MATCH policy can only be used with a VIPR-REGISTRATION Kind-ID. In this policy, a given value MUST be written (or overwritten) if and only if the Node-ID in the pvp_provider field of the ViprRegistration structure is equal to the first 16 bytes of the dictionary key and if the same Node-ID is the one indicated in the SignerIdentity of the value. Note that VIPR always let the values stored expire, so the exists field is always equal to TRUE.

Quota This kind MUST use the proportional quota algorithm described in [RELOAD-QUOTA] by adding the <max-count-per> element with a value of "SIGNER" to the configuration file. The <max-count> value, which measures the amount of E.164 numbers a particular node can store, MUST be adjusted to account for the application-layer copies (COPY1 and COPY2). A VIPR server MUST provide enough Node-IDs to store all the E.164 numbers it is responsible for by dividing this count by the <max-count> value, itself adjusted by an additional 3x factor to make sure that the probability is low

that a rejection occurs due to imperfect distribution of Resource-IDs across the ring.

[[Open Issue: need to adjust the multiplier - basically birthday problem!]]

The method for merging data after a partition follows the normal RELOAD rules around temporal ordering.

6. Overlay Configuration Document Extension

This document extends the overlay configuration document by defining a new element in the "urn:ietf:params:xml:ns:p2p:vipr" namespace.

The <store-rate-limit> defines the maximum rate in seconds that a VIPR server must use to execute Store requests.

The Compact Relax NG Grammar for this element is:

```
namespace vipr = "urn:ietf:params:xml:ns:p2p:vipr"
```

```
kind-parameter &= element vipr:store-rate-limit { xsd:unsignedInt }
```

7. Security Considerations

TBD

8. IANA Considerations

8.1. Access Control Policies

This document adds a new access control policy to the "RELOAD Access Control Policy" Registry:

+-----+-----+
Access-Policy RFC
+-----+-----+
VIPR-MATCH This document
+-----+-----+

This access control policy was described in Section 5.

8.2. Application-ID

This document adds a new application ID to the "RELOAD Application-ID" Registry:

Application	Application-ID	Specification
PVP	8470	This document

This access control policy was described in Section 5.

8.3. Data Kind-ID

This document adds a new Data Kind-ID to the "RELOAD Data Kind-ID Registry":

KIND	Kind-ID	RFC
VIPR-REGISTRATION	17	This document

This Kind-ID was defined in Section 5.

8.4. IETF XML Namespaces Registry

This document adds the following URN to the "XML Namespaces" class of the "IETF XML Registry":

urn:ietf:params:xml:ns:p2p:vipr

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.

[P2PSIP-BASE] Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol", draft-ietf-p2psip-base-16 (work in progress), July 2011.

[VIPR-OVERVIEW]

Jennings, C., Rosenberg, J., and M. Petit-Huguenin,
 "Verification Involving PSTN Reachability: Requirements
 and Architecture Overview",
 draft-jennings-vipr-overview-00 (work in progress),
 April 2011.

[VIPR-PVP]

Rosenberg, J., Jennings, C., and M. Petit-Huguenin, "The
 Public Switched Telephone Network (PSTN) Validation
 Protocol (PVP)", draft-petithuguenin-vipr-pvp-01 (work in
 progress), June 2011.

[RELOAD-QUOTA]

Rosenberg, J., Jennings, C., and M. Petit-Huguenin,
 "Proportional Quota in REsource LOcation And Discovery
 (RELOAD)",
 draft-petithuguenin-p2psip-proportional-quota-01 (work in
 progress), July 2011.

9.2. Informative References

[ACCESS-CONTROL]

Petit-Huguenin, M., "Configuration of Access Control
 Policy in REsource LOcation And Discovery (RELOAD) Base
 Protocol", draft-petithuguenin-p2psip-access-control-03
 (work in progress), July 2011.

Appendix A. Examples

The Resource Names and Resource-IDs for the E.164 number +1 408 555 5432 are:

Resource Name	Resource-ID
+14085555432	6abaec4308294521e2f600ab5fd01e5
COPY1+14085555432	9038006a8de78f818d318b60ed149d9
COPY2+14085555432	3d288c777bcf3aad38b077355026718

The VIPR-MATCH access control can be implemented with the following
 code (Using the notation in [ACCESS-CONTROL]):

```
var equals = function(a, b) {  
  if (a.length !== b.length) return false;  
  for (var i = 0; i < a.length; i++) {  
    if (a[i] !== b[i]) return false;  
  }  
  return true;  
};  
var length = configuration.node_id_length;  
return equals(entry.key.slice(0, length),  
  entry.value.slice(4, length + 4))  
  && equals(entry.key.slice(0, length), signature.node_id);
```

Appendix B. Release notes

This section must be removed before publication as an RFC.

B.1. Modifications between vipr-02 and vipr-01

- o Made clear in the access control policy that exists is always equal to TRUE.
- o Updated with new version of proportional-quota.
- o The access control code now uses the configuration parameter.
- o Assigned values to Application-ID and Kind-ID.
- o Added running code section.
- o Nits

B.2. Modifications between vipr-01 and vipr-00

- o Moved most of the quota algorithm to a separate I-D named draft-petithuguenin-p2psip-proportional-quota.
- o Removed the text saying that the same DHT can also be used for a RELOAD SIP usage, as it contradicts text in overview. Also the quota algorithm does not work with clients, but SIP registration uses clients.
- o Added Terminology section
- o Converted the TLV value stored to a structure using the syntax described in p2psip-base to not be dependent on VAP. The new structure is bit compatible with the old definition.
- o Changed the dictionary key to be binary based instead of text based.
- o Copied text from VAP explaining that the Store operations are queued and that the rate is limited.
- o Added voting algorithm when fetch returns different results for the 3 copies.
- o Added explanation on how to verify the signatures.

- o Added text on how to form the PVP connection
- o Rewrote some of the text so it looks more like a regular RELOAD usage.
- o Removed section 3 "PeerID Shim" now that support for multiple Node-ID in certificates is fully integrated in RELOAD base.
- o Filled IANA section
- o Added examples of conversion from E.164 number to Resource-ID
- o Added example code for the VIPR access control

B.3. Modifications between vipr-00 and dispatch-03

- o Moved to new Working Group.

B.4. Modifications between dispatch-03 and dispatch-02

- o Nits.
- o Shorter I-Ds references.
- o Fixed the peerID and VServiceID to be hexadecimal.
- o Fixed the description of the dictionary entry
- o Fixed the description of the TLV.
- o Used +1 408 555 prefix for phone numbers in examples.
- o Replaced peerId by Node-ID
- o Replaced resourceID by Resource-ID

B.5. Running Code Considerations

- o Reference Implementation for the kind and access control policy (<http://debian.implementers.org/testing/source/reload.tar.gz>). Marc Petit-Huguenin. Implements version -02.

Authors' Addresses

Jonathan Rosenberg
jdrosen.net
Monmouth, NJ
US

Email: jdrosen@jdrosen.net
URI: <http://www.jdrosen.net>

Cullen Jennings
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@cisco.com

Marc Petit-Huguenin
Stonyfish

Email: marc@stonyfish.com

VIPR
Internet-Draft
Intended status: Standards Track
Expires: January 12, 2012

J. Rosenberg
jdrosen.net
C. Jennings
Cisco
M. Petit-Huguenin
Stonyfish
July 11, 2011

Session Initiation Protocol (SIP) Extensions for Blocking VoIP Spam
Using PSTN Validation
draft-petithuguenin-vipr-sip-antispam-02

Abstract

Verification Involving PSTN Reachability (ViPR) is a new technique for inter-domain federation of SIP calls. ViPR makes use of the PSTN as an introduction mechanism to verify the correctness of mappings from phone numbers to domains. The PSTN introduction mechanism can also be used as a technique for blocking spam - a SIP caller is only authorized when its calling domain has previously called that same number over the PSTN. This document describes an extension to SIP which enables authorization of SIP calls based on a prior PSTN introduction.

Legal

This documents and the information contained therein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION THEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on January 12, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. Terminology	5
3. Terminating Side Procedures	5
4. Originating Side Procedures	6
5. Tickets	6
6. Security Considerations	7
7. IANA Considerations	7
7.1. IANA Registration of ViPR-Ticket Header Field	7
8. Acknowledgements	8
9. References	8
9.1. Normative References	8
9.2. Informative References	8
Appendix A. Release notes	9
A.1. Modifications between vipr-02 and vipr-01	9
A.2. Modifications between vipr-01 and vipr-00	9
A.3. Modifications between vipr-00 and dispatch-03	9
A.4. Modifications between dispatch-03 and dispatch-02	9
Authors' Addresses	9

1. Introduction

The anti-spam tickets described in this specification are the key security mechanism in ViPR for mitigation of SPAM. The domain originating a call inserts a ticket in the SIP INVITE sent to the other domain. The Border Element in the domain receiving the call (see Figure 1) can check the ticket to ensure that this originating domain has been authorized by the terminating domain. This document relies heavily on the concepts and terminology defined in [VI-PR-OVERVIEW] and will not make sense if you have not read that document first.

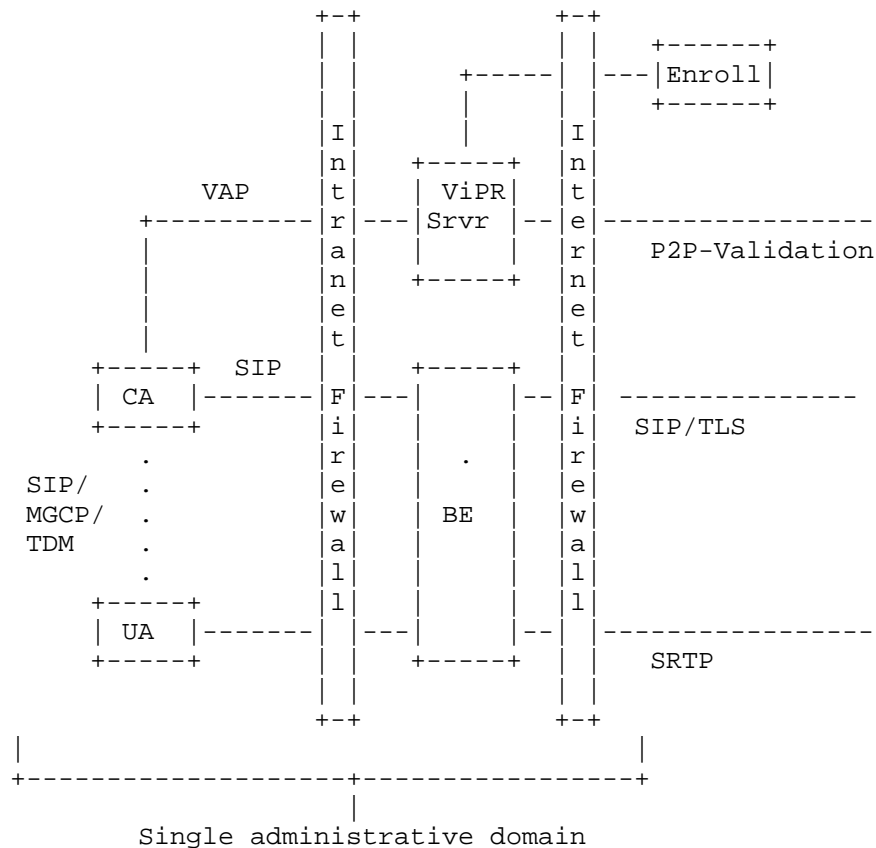


Figure 1: Architecture

2. Terminology

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 [RFC2119].

3. Terminating Side Procedures

The Border Element will receive the TLS ClientHello which begins the TLS handshake. The Border Element will present its own configured cert. Once TLS handshaking is complete, the Border Element notes the domain from the SubjectAltName on the other side of the TLS connection, and associates it with that connection.

Next, the Border Element will receive an INVITE. This INVITE will contain a ticket in the ViPR-Ticket header field value. The Border Element extracts this header field. This call flow assumes it is present. The Border Element parses it, and obtains the epoch value encoded in the ticket. This is matched against the current epoch value for the configured password. If they match, processing continues. The Border Element verifies the signature is correct. Next, it examines the start and stop time of the validity. If the current time is between the start and stop times, the check is passed. Next, the Border Element checks the granted-to domain in the ticket. It compares that domain against the domain name in the SubjectAltName of the peer on the other side of the TLS connection, as noted above. Next, it takes the Request-URI of the SIP INVITE. That will be of the form sip:+number@domain. If it is not in that form, and if the number does not begin with a plus, the request is dropped. The value, including the plus, is then compared to the number in the ticket. If it is equal, the check has passed. The Border Element leaves the header field in the request, but forwards to the Call Agent.

In addition, the Border Element will typically be configured to apply its SIP message validation logic, and enforce restrictions on the sizes of various SIP header fields. This provides an additional layer of security in case malicious SIP messages are sent.

The Border Element will also apply port forwarding in the case of NAT, so that the incoming request is forwarded to the appropriate Call Agent node.

The Call Agent will receive incoming SIP INVITES. The Request-URI of the INVITE will contain an E.164 number as indicated by a leading plus. If the Request-URI is not an E.164, the request must be rejected with a 403. Only E.164 numbers can be accepted on a ViPR

trunk.

4. Originating Side Procedures

The routes stored to other domains in the Call Agent will each store a ticket to utilize with calls to that route. The Call Agent learns about these routes and the information needed to construct the ticket from the VAP protocol [VIPR-VAP]. When sending a SIP request to one of these domains, the Call Agent MUST include the ticket in any dialog forming request or request that is not in an existing dialog.

5. Tickets

This ticket is a sequence of characters. These MUST be placed into a ViPR-Ticket SIP header field value. Consequently the format for this header field is:

```
Ticket = "ViPR-Ticket" HCOLON ticket-val  
ticket-val = 1*(alphanum / "-" / "_" / ".")
```

This header field MUST be utilized in all dialog forming requests and all out-of-dialog requests. It is not utilized in responses. The ticket-value is a modified base64 encoded version of an object that is composed of a series of TLVs. Each TLV is a 16 bit type, a 16 bit length, and a variable length value. The length field refers to the length of the value portion of the TLV, measured in bytes. The following TLV types are defined:

1. Ticket Unique ID: This TLV has a type of 0x0001. It contains a 128 bit ID that has a unique identifier for this ticket. The value MUST contain a 128 bit UUID defined by [RFC4122]. This TLV MUST be present. However at this time it is used for diagnostic purposes only.
2. Salt: This TLV has a type of 0x0002. It contains a value which MUST be at least 32 bits, and contains a random number. Its presence ensures that each ticket contains sufficient randomness. This TLV MUST be present.
3. Validity: This TLV has a type of 0x0003. It contains two 64 bit NTP times. The first is the start of the validity of the ticket, the next is the end time for the validity of the ticket. This TLV MUST be present.
4. Number: This TLV has a type of 0x0004. It contains a string which has an E.164 number, included the "+", which may be called using this ticket. The TLV has variable length. This TLV MUST be present.

5. Granting Node: This TLV has a type of 0x0005. It contains a 128 bit value which is the Node-ID of the node which granted the ticket. This TLV MUST be present.
6. Granting Domain: This TLV has a type of 0x0006. The domain which granted the ticket. A string, up to 256 characters, each of which must be a valid domain name character. The TLV has variable length. This TLV MUST be present.
7. Granted-To Domain: This TLV has a type of 0x0007. The domain to which the ticket is granted. A string, up to 256 characters, each of which must be a valid domain name character. The TLV has a variable length. This TLV MUST be present.
8. Epoch: This TLV has a type of 0x0008. It contains a 32 bit epoch value. It is used to select a key. This TLV MUST be present.
9. Integrity: This TLV has a type of 0x0009. It contains a 160 bit integrity value, computed using HMAC-SHA1. This TLV MUST be present and MUST be the last TLV in the object.

The base64 encoding uses the base64url encoding from RFC4648 [RFC4648], with the exception of the pad character, which is a "." instead of an "=". This ensures that the output is a valid SIP token.

To compute the MAC, the following is done. First, the key is obtained. The key is actually a 128 bit key, configured into the system. The key, P , is then used to compute K_m :

$$K_m = \text{HMAC-SHA1}(P, S \parallel \text{Epoch})$$

Based on PBKDF2 from PKCS #5 [RFC2898] with HMAC-SHA1 as PRF and iteration count of 1. Where S is the 32 bit salt and Epoch is the 32 bit Epoch, from the ticket. This produces a 160 bit K_m . The MAC is then computed as another HMAC-SHA1, over the entire ticket up to but not including the Integrity itself, using K_m as the key. This produces the 160 bit MAC.

6. Security Considerations

TBD

7. IANA Considerations

7.1. IANA Registration of ViPR-Ticket Header Field

This document defines a new SIP header field: ViPR-Ticket. Its syntax is defined in Section 5. This header field must be registered

by IANA in the SIP Parameters registry under the Header Fields subregistry with the following information:

Header Name: ViPR-Ticket
Compact Form: none

8. Acknowledgements

Thanks to Patrice Bruno for his comments, suggestions and questions that helped to improve this 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.
- [RFC2898] Kaliski, B., "PKCS #5: Password-Based Cryptography Specification Version 2.0", RFC 2898, September 2000.
- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.
- [VIPR-OVERVIEW] Jennings, C., Rosenberg, J., and M. Petit-Huguenin, "Verification Involving PSTN Reachability: Requirements and Architecture Overview", draft-jennings-vipr-overview-01 (work in progress), July 2011.

9.2. Informative References

- [VIPR-VAP] Jennings, C., Rosenberg, J., and M. Petit-Huguenin, "Verification Involving PSTN Reachability: The ViPR Access Protocol (VAP)", draft-jennings-vipr-vap-01 (work in progress), July 2011.

Appendix A. Release notes

This section must be removed before publication as an RFC.

A.1. Modifications between vipr-02 and vipr-01

- o Renamed Ticket to ViPR-Ticket to synchronize with -overview.

A.2. Modifications between vipr-01 and vipr-00

- o Renamed X-Cisco-ViPR-Ticket to Ticket and filled the IANA section.

A.3. Modifications between vipr-00 and dispatch-03

- o Moved to new Working Group.

A.4. Modifications between dispatch-03 and dispatch-02

- o Added terminology section.
- o Nits
- o Shorter I-Ds references.
- o Changed issued-to to granted-to.
- o Fixed the ABNF.
- o The tickets is used in all dialog forming requests, not only INVITE.
- o The Number TLV has a variable length.
- o The Integrity TLV MUST be the last in the object.
- o Fixed a discrepancy in the epoch length.

Authors' Addresses

Jonathan Rosenberg
jdrosen.net
Monmouth, NJ
US

Email: jdrosen@jdrosen.net
URI: <http://www.jdrosen.net>

Cullen Jennings
Cisco
170 West Tasman Drive
MS: SJC-21/2
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@cisco.com

Marc Petit-Huguenin
Stonyfish

Email: marc@stonyfish.com

