Session PEERing for Multimedia
INTerconnect
Internet-Draft
Expires: February 11, 2007

O. Lendl
enum.at

August 10, 2006

The Domain Policy DDDS Application
draft-lendl-domain-policy-ddds-02

Status of this Memo

Copyright Notice

Abstract

This documents proposes the use of the DNS to publish a domain's policy regarding incoming communication. The algorithm used is defined as a new application of the Dynamic Delegation Discovery System (DDDS). Such policy announcements can be used to facilitate selective SIP peering.

Table of Contents

1. Introduction

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 [7].

The DNS [5] is widely used to map a domain name to the lower layer parameters needed to connect to the service offered by this domain. At a basic level, an A record maps a domain name to an IP address. MX and SRV [8] records offer application specific mappings and define sets of alternative routes and their associated priorities.

NAPTR records as used in RFC 3263 [4] provide even more abstraction: They announce over which transport protocol a Session Initiation Protocol (SIP [9]) server prefers to be contacted.

This document defines another abstraction: Even if the service is announced by MX or SRV records and suitable A (or AAAA) records, access controls may apply. Not all URIs are reachable to anybody on the public Internet. Access restrictions can be implemented on various layers (IP, TLS, Application) and due to various reasons (technical, security, commercial, ...). Somebody wanting to use the service announced by MX or SRV records has usually no means to find out whether his connection will be accepted other than by trying to connect.

Such trial and error behavior is good enough for most applications, but in some cases it is vital that the service can announce to prospective clients the conditions under which a connection attempt is likely to succeed. This is mainly relevant for applications with the following properties:

- Real-time services: Any delay in establishing the connection is visible (and annoying) to the user. Waiting for a timeout is no problem for an email transmission or a background file transfer. This is an issue for interactive applications like e.g. web or VoIP services.

- Alternatives exist: If there are multiple technical ways to fulfill the customer's request then a fail-over from one method to another is possible. The sooner the sender knows that the first method he tries will fail, the earlier he can switch to the alternative.

The prime example for such an application is VoIP peering: Users are very sensitive to delays in the call-setup time, thus the originating network should not waste time on fruitless tries to reach the destination network via a direct SIP call. Currently the public

switched telephone network (PSTN) acts as a backup interconnection network over which VoIP networks based on E.164 telephone number can interconnect.

This document thus defines a protocol with which a domain can announce its policies regarding incoming communications for a specific protocol.

2. Policy Rules

The Domain Policy DDDS Application is built upon the concept of atomic policy rules. Complex policies are defined as a boolean combination of individual rules. This document does not define the semantics of any individual rule, this will be done by companion documents.

2.1. Atomic Policy Rules

Individual policy rules in the Domain Policy framework are expressed as Uniform Resource Identifiers (URI, RFC 2396 [1]). URIs can be used both as compact identifiers for standards (using e.g. "urn:ietf:rfc:XXXXX", "http://sippeering.example.com/release1") as well as URLs pointing to more elaborate policy statements.

Using URIs as identifiers is a common technique in the XML world. See for example XML namespaces [11] or XML DSIG [12] (1.3). The fact that these URIs are often URLs may be a bit confusing at first. Clients are not supposed to fetch and interpret the document to which these URLs refer to. These URLs are just convenient as unique and descriptive identifiers.

Simple protocol parameters (e.g. a list of X.509 CAs) can be included into the policy rule URI (e.g. in the query part) itself.

2.1.1. Notification

Policy rules can contain notifications about policies: Such rules do not by themselves describe the conditions under which calls will be accepted. Instead, such atomic rules reference other, potentially complex documents which need not be available online. Examples are contracts or other non-technical documents. On the sender side the support for such rules will have to be manually configured (e.g. "we are a member of federation X").

2.1.2. Publication

On the other hand, policy rules can describe very concrete

restrictions, e.g. the support for a certain protocol standard. In that case, a software package implementing the sender side can include a default set of policy rules which it fulfills out of the box.

2.2. Complex Policies

To link individual policy rules into one complex policy the disjunctive normal form of boolean logic is used. In simple words: The policy is written as a set of valid (according to the policy) alternatives. These alternatives may themselves consist of individual rules which must all be fulfilled to yield a positive result.

If the simple boolean logic and the limited space in the DNS answers are insufficient for an application then a type of policy rule URIs can be defined which links to policy statements written in a fully featured policy description language like SAML [13] or XACML [14].

2.3. Policy Enforcement

This document does not specify if and how the announced policies are enforced. The Domain Policy DDDS just gives operators the option to document and publish what kind of communications their servers are configured to accept.

3. DDDS Specification

This section contains the formal definition of the Domain Policy DDDS application according to RFC 3401 [2].

RFC 3401 describes DDDS as follows:

> The Dynamic Delegation Discovery System is used to implement lazy binding of strings to data, in order to support dynamically configured delegation systems. The DDDS functions by mapping some unique string to data stored within a DDDS Database by iteratively applying string transformation rules until a terminal condition is reached.

The Domain Policy DDDS Application maps a domain name and a protocol name to an ordered boolean expression of atomic policy rules. It does not define how individual rules should be interpreted (except that unknown rules must be regarded as not fulfilled). It defines how these rules are retrieved and how they are combined and processed.

As a domain's policy may be different for the various protocols (e.g. SMTP vs. SIP), the name of the protocol is another input into the DDDS application.

3.1. Application Unique String

The Application Unique String is the URI which describes the service the originating network wants to access.

Examples: "sip:user@example.com", "mailto:sales@example.org"

3.2. First Well Known Rule

The First Well Known Rule extracts the domain part of the URI. This key thus has the form of a fully qualified domain name.

3.3. Expected Output

The Expected Output of this algorithm is a set of URIs which describe the conditions the sender can fulfill.

3.4. Valid Databases

This DDDS application uses the DNS as the database as defined in RFC 3403 [3]. The rewriting rules are stored in NAPTR DNS resource records. As the key is already in the form of a FQDN, no transformations are necessary.

3.4.1. Services Parameter

The "services" field in the NAPTR record is used to
o identify this NAPTR as part of the Domain Policy DDDS application,
o define to which protocol this policy applies to, and
o identify which type of policy is contained in this record.
This document does not define actual policy types, that is left to companion documents. The formal specification for the service field is as follows:

```
service-field       = "D2P+" protocol *1policy
protocol            = 1*32(ALPHA / DIGIT)
policy              = ":" policy-type
policy-type         = 1*32(ALPHA / DIGIT)
```

In other words, the service-type starts with "D2P+" (which marks this as part of the Domain Policy DDDS) followed by the name of the protocol to which this policy applies to (e.g. "sip", "xmpp", "smtp", ...). Optionally the policy-type (which follows after a colon) indicates what kind of policy rule is contained in the regexp field.

All matching operations on the service-field MUST be done in a case-insensitive way.

The Domain Policy DDDS Application client MUST ignore all records where the protocol in the service-field does not match the protocol for which the policy is sought.

3.4.2. Flags

The "flags" field in the NAPTR record signals when the DDDS algorithm has finished.

An empty (non-existent) flag means that this rule is non-terminal and the client MUST use the key resulting from this rule as the input into a new DDDS loop. Such non-terminal NAPTRs have an empty "regexp" field and contain a new domain name in the "replacement" field. They MUST NOT contain a policy-type element in the service-field.

Such non-terminal NAPTRs are referrals: They indicate that communications to the original domain can be routed via the replacement domain. This implies that if this referral is selected then all further DNS lookups (both for policy and signalling parameters) MUST be done by querying the new domain.

A non-terminal NAPTR can lead to another non-terminal NAPTR record. Clients need to limit the recursion depth to prevent looping.

A flag containing "U" signals that the algorithm has found a valid policy record for this domain. This rule MUST be considered in conjunction with all other rule which share the same protocol (from the service-field), the same flag and the same order field. If the client can determine that he is able to fulfill all the requirements in this set of atomic rules then the DDDS algorithm terminates and the result is this set of rules.

If the policy-type of a rule is not known to the client it MUST consider this rule as not fulfill-able.

No other values for the flag field are defined as of now and clients MUST ignore all records not containing either "U" or and empty flag field.

The treatment of the "order" and "preference" fields as defined here deviates slightly from RFC 3403. The reason is that this DDDS application does not return a single URI as found in a single NAPTR record but a set of URIs which are generated by grouping NAPTR records based on their "order" field. The "preference" field is not

evaluated.

3.5. DNS considerations

The NAPTR records containing the policy announcements for a domain can be quite large for DNS responses, especially if elaborate rules are used or if this mechanism is used for more than one protocol.

To accommodate these larger DNS responses the DNS servers and the clients MUST utilize EDNS0 [6] to minimise fall-backs to TCP queries. This requirement is the equivalent of [10].

Storing policy rules directly in the DNS is very efficient as long as the rule-set is small. One must take care not to overload this database. If more elaborate rules are needed it is recommended to use the DNS only to refer to a policy statement stored elsewhere.

4. Registration mechanism for policy-types

The service-field of the NAPTR records used in this document contain three fields:

- The constant string "D2P".
- The protocol name. Protocol names are to be taken from the IANA registry at http://www.iana.org/assignments/port-numbers. No new registry functions are needed for this field.
- The policy-type. This field indicates the interpretation rules for the URI contained in the NAPTR record. An IANA registration procedure is thus needed for this field.

New entries in the IANA policy-type registry need to specify not only the name of the new policy-type, but also need to include the allowed URI schemes, a functional specification on how to interpret the URI, security considerations, intended usage, and any other information needed for to evaluate policy rules of this type. In order to be a registered policy-type, the entire specification, including the template, requires approval by the IESG and publication of the policy-type registration specification as an RFC.

4.1. Functionality Requirement

A registered policy-type acts as selector within the policy evaluation engine of the client. The specification in the registration MUST be sufficient such that the client can determine whether he can fulfill the policy requirements encoded in the URI. Specifically, a registered policy-type MUST specify the URI scheme(s) that may be used, and, when needed, other information which will have to be transferred into the policy evaluation process itself.

4.2. Naming requirement

Registered policy-types must by unique and conform to the ABNF specified in Section Section 3.4.1, and MUST NOT start with the facet "X-" which is reserved for experimental, private use.

4.3. Publication Requirements

Proposals for policy-type registrations MUST be published as one of the following documents; RFC on the Standards Track, Experimental RFC, or as a BCP.

IANA will retain copies of all policy-type registration proposals and "publish" them as part of the policy-type Registration tree itself.

4.4. Registration Template

Policy Type:

URI Scheme(s):

Functional Specification:

Security considerations:

Intended usage: (One of COMMON, LIMITED USE or OBSOLETE)

Author:

Any other information that the author deems interesting:

5. Examples

In order to give meaningful examples, we need to define a few types of rules. The definitions here are purely meant to illustrate the possibilities. They MUST NOT be considered as valid examples of real entries.

- The policy-type "fed" shall indicate that the URI denotes a federation of service providers. All members of a federation know how to talk to each other.
- The policy-type "std" shall indicate that the URI indicates a standard the sender has to fulfill.
- A record with policy-type "saml" shall contain an URL of a SAML document which contains further policy information.

- In this example "customer.example.org" is a customer domain for which the service-provider "example.com" hosts the SIP server. Thus, for incoming SIP calls for customer.example.org an originating VSP is directed to to use example.com as the next hop, and use the supported policies of example.com to deliver the call.

```
$ORIGIN customer.example.org.
;      order pref flags service regexp replacement
 IN NAPTR 10 50   "" "D2P+SIP" ""   example.com.
```

  Please note that there is no policy-type in this record. The protocol name is needed though, as e.g. this customer could have his email hosted at a different provider and thus refer to a different domain with a "D2P+SMTP" NAPTR.

  An orginating SIP provider trying to establish a call to <sip:bob@customer.example.org> will thus look at the domain example.com for further policy rules. If he finds matching ones (eg. a shared federation) he will use "example.com" and not "customer.example.org" as the input into the RFC 3263 "Locating SIP Servers" algorithm.

- In the second example the SIP services at "example.com" is only reachable via a private interconnection arrangement maintained by a federation called "http://sipxconnect.example.org/".

```
$ORIGIN example.com.
 IN NAPTR 10 50 (                                ; order priority
       "U" "D2P+SIP:fed"                         ; flags service
       "!^.*$!http://sipxconnect.example.org/!" . ; regexp repl
       )
```

- In this example the SIP services at "example.com" from above also buys transit services from "example.net". Other providers (especially those which are not members of "http://sipxconnect.example.org/") can additionally reach users of "example.com" by routing calls via "example.net".

```
$ORIGIN example.com.
 IN NAPTR 10 50 (                                ; order priority
       "U" "D2P+SIP:fed"                         ; flags service
       "!^.*$!http://sipxconnect.example.org/!" . ; regexp repl
       )
 IN NAPTR 20 50   "" "D2P+SIP" ""   example.net.
```

- In the next example the "example.com" also allows incoming connections as long they use SIP over TLS. Calls according to federation rules are preferred.

```
$ORIGIN example.com.
 IN NAPTR 10 50 (                                ; order priority
       "U" "D2P+SIP:fed"                         ; flags service
       "!^.*$!http://sipxconnect.example.org/!" . ; regexp repl
       )
 IN NAPTR 20 10 (                                ; order priority
       "U" "D2P+SIP:std"                         ; flags service
       "!^.*$!urn:ietf:rfc:2246!" .              ; regexp repl
       )
```

- If the carrier example.com only accepts SIP calls (in addition to PSTN interconnect) if the PSTN emulation is good, he might publish a policy like this:

```
$ORIGIN example.com.
;      order pref flags service       regexp           replacement
 IN NAPTR 10 10   "U" "D2P+SIP:std" "!^.*$!urn:ietf:rfc:3578!" .
 IN NAPTR 10 11   "U" "D2P+SIP:std" "!^.*$!urn:ietf:rfc:3666!" .
 IN NAPTR 10 12   "U" "D2P+SIP:std" "!^.*$!urn:ietf:rfc:3960!" .
```

- A restrictive SIP service might only accept calls from peers from two federations, while subjecting calls from the public Internet to a complex policy. The policy records could look like this:

```
$ORIGIN example.com
 IN NAPTR 10 10 (                                ; order priority
       "U" "D2P+SIP:fed"                         ; flags service
       "!^.*$!http://sipxconnect.example.org/!" . ; regexp repl
       )
 IN NAPTR 20 10 (                                ; order priority
       "U" "D2P+SIP:fed"                         ; flags service
       "!^.*$!http://sip.federation.com/!" .     ; regexp repl
       )
 IN NAPTR 30 10 (                                ; order priority
       "U" "D2P+SIP:saml"                        ; flags service
       "!^.*$!http://www.example.com/sip-policy.saml!" .
       )
```

6. Security Considerations

The publishing of the access policy via the DNS RR described in this draft will reduce the amount of unwanted communication attempts, as

all well-meaning clients will follow them, but these records cannot substitute measures to actually enforce the published policy.

In the case of SIP Peering, the NAPTRs proposed here publish information which is not public if service providers just rely on private interconnection agreements. These records are similar to the public routing registries for BGP4 as maintained by the RIRs. They are just records indicating who peers with whom but do not hold details on how the interconnection is achieved.

The published technical requirements for incoming calls could be used by malicious callers to find possible attack vectors.

7. IANA Considerations

This document defines an IANA registry for the policy-type field. See Section 4.

8. Acknowledgements

The author would like to thank Richard Shockey, Michael Haberler, Alexander Mayrhofer, Richard Stastny, and John Todd for their contributions.

9. References

9.1. Normative References

[1] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[2] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part One: The Comprehensive DDDS", RFC 3401, October 2002.

[3] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database", RFC 3403, October 2002.

[4] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, June 2002.

[5] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[6] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, August 1999.

9.2. Informative References

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

[8] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[9] 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.

[10] Conroy, L. and J. Reid, "ENUM Requirement for EDNS0 Support", draft-conroy-enum-edns0-01 (work in progress), October 2005.

[11] Hollander, D., Bray, T., and A. Layman, "Namespaces in XML", W3C REC REC-xml-names-19990114, January 1999.

[12] Solo, D., Reagle, J., and D. Eastlake, "XML-Signature Syntax and Processing", W3C REC REC-xmldsig-core-20020212, February 2002.

[13] Maler, E. and J. Hughes, "Security Assertion Markup Language (SAML) V2.0 Technical Overview", July 2005.

[14] Godik, S. and T. Moses, "OASIS eXtensible Access Control Markup Language (XACML)", OASIS Committee Working Draft xacml-schema-policy, July 2002, <http://www.oasis-open.org/committees/xacml/repository/draft-xacml-schema-policy-15.doc>.

Author's Address

Otmar Lendl
enum.at GmbH
Karlsplatz 1/9
Wien A-1010
Austria

Phone: +43 1 5056416 33
Email: otmar.lendl@enum.at
URI: http://www.enum.at/

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.