Network Working Group
Internet-Draft
Intended status: Experimental
Expires: August 27, 2011

D. Crocker
Brandenburg InternetWorking
M. Kucherawy
Cloudmark

February 23, 2011

MIME Content Authentication using DOSETA (MIMEAUTH)
draft-crocker-doseta-mimeauth-00

Abstract

MIME is a method of packaging and labeling aggregations of data; it is used both for email and the Web. Many usage scenarios would benefit by having an objective method of assessing the validity of MIME data, based on an authenticated identity. MIMEAUTH leverages technology developed for DKIM to provide such a method. Its use can be extended to cover specific header-fields of a containing email message or World Wide Web HTTP content. Existing authentication mechanisms have achieved only limited success due to challenges with administration and use. MIMEAUTH has very low administration and use overhead, through self-certifying keys in the DNS and a labeling method that can be transparent to end-users. For relayed and mediated sequences, MIMEAUTH can be implemented within a service and therefore can be transparent to end-system software.

Table of Contents

1. Introduction

MIME is a core data-packaging mechanism for Internet applications; it is used both for email and the Web. Many usage scenarios would benefit by having an objective method of assessing the validity of MIME data, based on an authenticated identity. Existing authentication mechanisms have achieved only limited use. MIMEAUTH is based on DOSETA [I-D.DOSETA] to provide such a method. Its use can be extended to cover specific header-fields of a containing email message or World Wide Web HTTP content. MIMEAUTH has very low administration and use overhead, through self-certifying keys in the DNS and a labeling method that can be transparent to end-users. For relayed and mediated sequences, MIMEAUTH can be implemented within a service and therefore can be transparent to end-system software.

The approach taken by MIMEAUTH differs from previous approaches to message authentication, such as Secure/Multipurpose Internet Mail Extensions (S/MIME) [RFC1847] and OpenPGP [RFC4880], in that:

- the signature is written as an associated attribute in a header field, rather than being integrated into the data itself, so that neither human recipients nor existing MUA (Mail User Agent) software is confused by signature-related content appearing in the data;
- there is no dependency on having public and private key pairs being issued by well-known, trusted certificate authorities;
- there is no dependency on the deployment of any new Internet protocols or services for public key distribution or revocation;
- authentication is distinct from encryption;

MIMEAUTH:

- is compatible with the existing email and Web infrastructure and is transparent to it, to the fullest extent possible;
- requires minimal new infrastructure;
- can be implemented independently of clients in order to reduce deployment time;
- can be deployed incrementally;
- allows delegation of signing to third parties.

1.1. Signing Identity

MIMEAUTH separates specification of the identity of the MIMEAUTH signer from the purported author of the content. Verifiers can use the signing information to decide how they want to process the data. In particular, the authentication identity specified by a MIMEAUTH signature is not required to match any other identifier the content or the header. However when the identity does match other, specific identities, specific semantics are assigned.

1.2. Terminology and Definitions

This specification incorporates the terminology defined in [I-D.DOSETA].

Syntax descriptions use Augmented BNF (ABNF) [RFC5234].

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].

Additional terminology:

Internal IDentfier (IID): An optional textual literal token that can be included with a signature and can be part of communications back to the signer, as a reference to the signature. For DOSETA processing, the domain name portion of the IID has only basic domain name semantics; any possible owner-specific semantics are outside the scope of DOSETA. That is, for MIMEAUTH, the entire string is an undifferentiated literal. It is specified in Section 3.1 .

1.3. Open Issues

Still to be resolved:

- Precise semantics of a signature. Does it merely declare "responsibility" by the signer, or "validity" of the content, or something else?

- Should there be a flag that differentiates among different possible semantics, such as defaulting to "responsibility" but able to flag assertion of validity? How dangerous would such a flag be?

NOTE: A variety of assurances can be made about an email From: field, such as validity of the domain portion, validity of the entire email address, and validity of the <display-name> string. This, of course, is separate from whether to trust /any/ assurances being made...

2. Signing and Verifying Protocol

MIMEAUTH uses the DOSETA "Generic Header/Content Signing Service Template" [I-D.DOSETA] as its base.

The DOSETA Template specifies features labeled TEMPLATE that need to be tailored to a specific signing service. For MIMEAUTH, the tailored features are:

Signature Association: The DOSETA-Signature data are stored in a MIME Content-Authentication: header field that is part of the MIME object being authenticated. This contains all of the signature and key-fetching data, per [I-D.DOSETA].

Semantics Signaling: The presence of a MIME Content-Authentication: header field signals the use of MIMEAUTH.

Semantics: A MIMEAUTH signature means that the owner of the DDI is taking direct responsibility for the signed content and for the content of any referenced header fields, if present. Hence, the payload, or output, of MIMEAUTH is:

+ The DDI domain name, specifically the "d" parameter in the MIME Content-Authentication: header field

+ A list of referenced header fields

+ An indication that the signature verified

NOTE: The semantics of this signature are much stronger than the semantics of a DKIM signature and pertain to the content, not merely the signing domain. [RFC4871]

Header/Content Mapping: MIMEAUTH maps the DOSETA header processing to the cited header fields that are associated with the MIME object. DOSETA Content maps to the MIME body, per [RFC2045]. The Content-type: header field is always covered by the signature.

When a MIMEAUTH signature lists additional header fields in the "h" parameter, MIMEAUTH is asserting that these also have valid data. By including other common header fields that are associated with MIME usage, the scope of a MIMEAUTH signature can apply to a containing protocol data unit, such as an email message or a Web payload. Therefore, when the authentication semantic is intended to assert validity of both the MIME data and the context in which it occurs, a minimum set of additional header fields SHOULD be included in the DOSETA "h" parameter. This is discussed further in Section 3.2.

3. Extensions to DOSETA Template

This section contains specifications that are added to the basic DOSETA H/C Signing Template.

3.1. Signature Data Structure

These are MIMEAUTH-specific tags:

i= This specifies an Internal Identifier (IID) token that can be used when referring to the signed data, back to the signer. Within the MIMEAUTH protocol, the string has no value except as a literal token. Any conventions for the string that are imposed by the signer are unknown to other MIMEAUTH participants.

The syntax is the form of a standard email address where the <local-part> MAY be omitted.

Internationalized domain names MUST be converted using the steps listed in Section 4 of [RFC5890] using the "ToASCII" function.

ABNF:

```
sig-i-tag       = %x69 [FWS] "=" [FWS]
                  [ local-part ] "@" domain-name
```

z= Copied header fields (DOSETA-quoted-printable, but see description; OPTIONAL, default is null). A vertical-bar-separated list of selected header fields present when the message was signed, including both the field name and value. It is not required to include all header fields present at the time of signing. This field need not contain the same header fields listed in the "h=" tag. The header field text itself MUST encode the vertical bar ("|", %x7C) character. That is, vertical bars in the "z=" text are meta-characters, and any actual vertical bar characters in a copied header field MUST be encoded. Note that all whitespace MUST be encoded, including whitespace between the colon and the header field value. After encoding, FWS MAY be added at arbitrary locations in order to avoid excessively long lines; such whitespace is NOT part of the value of the header field, and MUST be removed before decoding.

Copied header field values are for diagnostic use.

Header fields with characters requiring conversion SHOULD be converted as described in MIME Part Three [RFC2047].

ABNF:

```
sig-z-tag      = %x7A [FWS] "=" [FWS]
                 sig-z-tag-copy
                 *( "|" [FWS] sig-z-tag-copy )
sig-z-tag-copy = hdr-name [FWS] ":"
                 qp-hdr-value
```

3.2. Email Signed Header Fields

Some header fields have semantics that are relevant to end users and often are presented to them. If MIMEAUTH is used to sign an email message, it is useful to cover such header fields, in addition to the MIME content. This section provides a generic recommendation intended to apply to the general case of signing a message; specific senders might wish to modify these guidelines as required by their unique situations. Verifiers MUST be capable of verifying signatures even if one or more of the recommended header fields is not signed or if one or more of the dis-recommended header fields is signed. Note that verifiers do have the option of ignoring signatures that do not cover a sufficient portion of the header or content, just as they might ignore signatures from an identity they do not trust.

The signer is encouraged to consider carefully which fields are important to the interpretation of the content and which ones are not. As an example, note what fields are typically displayed to recipients. The following header fields are listed as a default set and SHOULD be included in the signature, if they are present in the message being signed:

- From, Reply-To, Resent-From
- Subject
- Date, Message-ID, Resent-Date, Resent-Message-ID
- To, Cc, Resent-To, Resent-Cc
- Content-Type, Content-ID, Content- Description)
- List-Id, List-Help, List-Unsubscribe, List-Subscribe, List-Post, List-Owner, List-Archive

The following header fields SHOULD NOT be included in the signature:

- Return-Path
- Received
- Comments, Keywords
- Bcc, Resent-Bcc
- Content-Signature (MUST NOT include)

Optional header fields (those not mentioned above) normally SHOULD NOT be included in the signature, due to the possibility of having additional header fields, of the same name, that are added or reordered legitimately, prior to verification. There are likely to be reasonable exceptions to this rule, given the wide variety of application-specific header fields that might be applied to a message, some of which are unlikely to be duplicated, modified, or reordered.

4. Considerations

4.1. Security Considerations

4.2. IANA Considerations

MIMEAUTH uses registries assigned to DOSETA [I-D.DOSETA]. This section specifies additions to these registries.

4.2.1. Content-Authentication Tag Specifications

These values are added to the registry that is now defined in [I-D.DOSETA]:

TYPE	REFERENCE
i	(this document, Section 3.1)
z	(this document, Section 3.1)

Table 1: Content-Authentication Tag Initial Values

4.2.2. Content-Authentication Header Field

IANA has added MIME Content-Authentication: to the "Permanent Message Header Fields" registry (see [RFC3864]) for the "mail" protocol, using this document as the reference.

5. References

5.1. Normative References

[I-D.DOSETA] Crocker, D., Ed. and M. Kucherawy, Ed., "DomainKeys Security Tagging (DOSETA)", I-D draft-ietf-crocker-doseta-base, 2011.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[RFC2047] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Content", RFC 2047, November 1996.

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

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, January 2008.

[RFC5890] Klensin, J., "Internationalizing Domain Names in Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.

5.2. Informative References

[RFC1847] Galvin, J., Murphy, S., Crocker, S., and N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, October 1995.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

[RFC4871] Allman, E., Callas, J., Delany, M., Libbey, M., Fenton, J., and M. Thomas, "DomainKeys Identified Mail (DKIM) Signatures", RFC 4871, May 2007.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., and R. Thayer, "OpenPGP Message Format", RFC 4880, November 2007.

Appendix A. Acknowledgements

Authors' Addresses

D. Crocker
Brandenburg InternetWorking
675 Spruce Dr.
Sunnyvale
USA

Phone: +1.408.246.8253
Email: dcrocker@bbiw.net
URI: http://bbiw.net

M. Kucherawy
Cloudmark
128 King St., 2nd Floor
San Francisco, CA 94107
USA

Email: msk@cloudmark.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 11, 2011

P. Faltstrom, Ed.
Cisco
P. Hoffman, Ed.
VPN Consortium
June 9, 2011

The Unicode code points and IDNA - Unicode 6.0
draft-faltstrom-5892bis-05.txt

Abstract

This memo documents IETF consensus for IDNA derived character properties related to the three code points, existing in Unicode 5.2, that changed property values when version 6.0 was released. The consensus is that no update is needed to RFC 5892 based on the changes made in Unicode 6.0.

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 11, 2011.

Copyright Notice

Table of Contents

1. Introduction

RFC 5892 [RFC5892] specifies an algorithm that was defined when version 5.0 (later updated to version 5.2) [Unicode5.2] was the current version of Unicode, and it also defines a derived property value based on that algorithm. Unicode 6.0 [Unicode6] has changed GeneralCategory of three code points that were allocated in Unicode 5.2 or earlier. This implies the derived property value differs depending on whether the property definitions used are from Unicode 5.2 or 6.0. These are non-backward-compatible changes as described in section 5.1 of RFC 5892.

The three code points are:

1.1. U+0CF1 KANNADA SIGN JIHVAMULIYA

The GeneralCategory for this character changes from So to Lo. This implies that the derived property value changes from DISALLOWED to PVALID.

1.2. U+0CF2 KANNADA SIGN UPADHMANIYA

The GeneralCategory for this character changes from So to Lo. This implies that the derived property value changes from DISALLOWED to PVALID.

1.3. U+19DA NEW TAI LUE THAM DIGIT ONE

The GeneralCategory for this character changes from Nd to No. This implies that the derived property value changes from PVALID to DISALLOWED.

2. IETF Consensus

No change to RFC 5892 is needed based on the changes made in Unicode 6.0.

This consensus does not imply that no changes will be made to RFC 5892 for all future updates of The Unicode Standard.

This RFC is being produced because 6.0 is the first version of Unicode to be released since IDNA2008 was published.

3. IANA Considerations

IANA is to update the derived property value registry according to

RFC 5892 and property values as defined in The Unicode Standard version 6.0.

4. Security Considerations

When the algorithm presented in RFC 5892 is applied using the property definitions of Unicode Standard Version 6.0, the result will be different from when it is applied using the property definitions of Unicode 5.2 for the three code points discussed in this document in addition to the changes for code points being unassigned in Unicode 5.2. The three code points are unlikely to occur in internationalized domain names, however, so the security implications of the changes are minor.

5. Acknowledgements

The main contributors are (in alphabetical order) Eric Brunner-Williams, Vint Cerf, Tina Dam, Martin Duerst, John Klensin, Mark Davis, Pete Resnick, Markus Scherer, Andrew Sullivan, Kenneth Whistler and Nicholas Williams.

Not all contributors believe the solution for the issues discussed in this document is optimal.

6. Normative References

[RFC5892] Faltstrom, P., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, August 2010.

[Unicode5.2] The Unicode Consortium, "The Unicode Standard, Version 5.2.0", Unicode 5.0.0, Boston, MA, Addison-Wesley ISBN 0-321-48091-0, as amended by Unicode 5.2.0 http://www.unicode.org/versions/Unicode5.2.0/, 2009, <http://www.unicode.org/versions/Unicode5.2.0/>.

[Unicode6] The Unicode Consortium, "The Unicode Standard, Version 6.0.0", October 2010.

Authors' Addresses

Patrik Faltstrom (editor)
Cisco

Email: paf@cisco.com

Paul Hoffman (editor)
VPN Consortium

Email: paul.hoffman@vpnc.org

Network Working Group
Internet-Draft
Obsoletes: 5333 (if approved)
Updates: 6118 (if approved)
Intended status: Informational
Expires: September 7, 2011

B. Hoeneisen
Ucom.ch

March 06, 2011

IANA Registration of Enumservices for Internet Calendaring
draft-hoeneisen-rfc5333bis-00

Abstract

This document registers and updates Enumservices for Internet calendaring. Specifically, this document focuses on Enumservices for scheduling with iMIP (iCalendar Message-Based Interoperability Protocol) and for accessing Internet calendaring information with CalDAV (Calendaring Extensions to WebDAV).

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 September 7, 2011.

Table of Contents

1. Introduction

ENUM (E.164 Number Mapping, RFC 6116 [I-D.ietf-enum-3761bis]) is a system that uses DNS (Domain Name System, [RFC1034]) to translate telephone numbers, such as '+12025550100', into URIs (Uniform Resource Identifiers, [RFC3986]), such as 'mailto:user@example.com'. ENUM exists primarily to facilitate the interconnection of systems that rely on telephone numbers with those that use URIs to identify resources. The ENUM registration here could be used to allow phones, for example, to check the free/busy status of a user in their address book or propose a meeting with him or her from the user's phone number.

The Guide to Internet Calendaring [RFC3283] describes the relationship between various Internet calendaring specifications like this: "iCalendar [RFC5545] is the language used to describe calendar objects. iTIP [RFC5546] [iCalendar Transport-Independent Interoperability Protocol] describes a way to use the iCalendar language to do scheduling. iMIP [RFC2447] [iCalendar Message-Based Interoperability Protocol] describes how to do iTIP scheduling via e-mail".

Recently, another Standards Track protocol for calendar and scheduling access has appeared. CalDAV (Calendaring Extensions to WebDAV) [RFC4791] is a WebDAV (Web-based Distributed Authoring and Versioning) [RFC4918] based mechanism for manipulating Internet calendars, viewing free/busy lists, and via a scheduling extension [I-D.desruisseaux-caldav-sched], could be used for proposing calendar events as well in the future.

The existing 'mailto:' URI scheme (defined in [RFC6068]) is already used to address iMIP compatible Calendar Services. Likewise, the existing 'http:' and 'https:' URI schemes (defined in [RFC2616] and [RFC2818]) are already used to address CalDAV compatible Calendar Services.

This document updates the existing Calendaring Enumservices specified in [RFC5333] and in Sections 4.7 - 4.9 of RFC 6118 [I-D.ietf-enum-enumservices-transition] to be in line with RFC 6117 [I-D.ietf-enum-enumservices-guide]. Furthermore it adds new Enumservices for WebDAV scheduling [I-D.desruisseaux-caldav-sched].

This document obsoletes [RFC5333] as well as Sections 4.7 - 4.9 of RFC 6118 [I-D.ietf-enum-enumservices-transition].

2. Enumservice Registrations

As defined in RFC 6117 [I-D.ietf-enum-enumservices-guide], the following templates cover the information needed for the registration of the Enumservices specified in this document:

[Note for RFC Editor: Please perform this following tasks before publication of this document:
- replace any instance of "RFCTHIS" with the RFC number of this document;
- replace any instance of "I-D.desruisseaux-caldav-sched" with the RFC number of that document;
- verify references pointing to specific sections of this document, in particular in the <security> elements.

]

2.1. ical-access:http (update)

```
<record>
  <!-- ical-access:http -->
  <class>Application-Based, Common</class>
  <type>ical-access</type>
  <subtype>http</subtype>
  <urischeme>http</urischeme>
  <functionalspec>
    <paragraph>
      This Enumservice indicates that the resource identified
      can be addressed by the associated URI in order to access
      a user's calendar (for example free/busy status) using
      the CalDAV [RFC 4791] protocol for Internet calendaring.
    </paragraph>
  </functionalspec>
  <security>
    See <xref type="rfc" data="rfcTHIS" />, Section 4.
  </security>
  <usage>COMMON</usage>
  <registrationdocs>
    <xref type="rfc" data="rfcTHIS" />
  </registrationdocs>
  <requesters>
    <xref type="person" data="Rohan_Mahy" />
    <xref type="person" data="Bernie_Hoeneisen" />
  </requesters>
</record>
```

2.2. ical-access:https (update)

```
<record>
  <!-- ical-access:https -->
  <class>Application-Based, Common</class>
  <type>ical-access</type>
  <subtype>https</subtype>
  <urischeme>https</urischeme>
  <functionalspec>
    <paragraph>
      This Enumservice indicates that the resource identified
      can be addressed by the associated URI in order to access
      a user's calendar (for example free/busy status) using
      the CalDAV [RFC 4791] protocol for Internet calendaring.
    </paragraph>
  </functionalspec>
  <security>
    See <xref type="rfc" data="rfcTHIS" />, Section 4.
  </security>
  <usage>COMMON</usage>
  <registrationdocs>
    <xref type="rfc" data="rfcTHIS" />
  </registrationdocs>
  <requesters>
    <xref type="person" data="Rohan_Mahy" />
    <xref type="person" data="Bernie_Hoeneisen" />
  </requesters>
</record>
```

2.3. ical-sched:mailto (update)

```
<record>
  <!-- ical-sched:mailto -->
  <class>Application-Based, Common</class>
  <type>ical-sched</type>
  <subtype>mailto</subtype>
  <urischeme>mailto</urischeme>
  <functionalspec>
    <paragraph>
      This Enumservice indicates that the resource identified
      can be addressed by the associated URI used for
      scheduling using Internet calendaring via Internet mail
      with the iMIP [RFC 2447] protocol.
    </paragraph>
  </functionalspec>
  <security>
    See <xref type="rfc" data="rfcTHIS" />, Section 4.
  </security>
  <usage>COMMON</usage>
  <registrationdocs>
    <xref type="rfc" data="rfcTHIS" />
  </registrationdocs>
  <requesters>
    <xref type="person" data="Rohan_Mahy" />
    <xref type="person" data="Bernie_Hoeneisen" />
  </requesters>
</record>
```

2.4. ical-sched:http (new)

TODO: Do we need this one?

```
<record>
  <!-- ical-sched:http -->
  <class>Application-Based, Common</class>
  <type>ical-sched</type>
  <subtype>http</subtype>
  <urischeme>http</urischeme>
  <functionalspec>
    <paragraph>
      This Enumservice indicates that the resource identified can be
      addressed by the associated URI used for scheduling using
      Internet calendaring via the scheduling extension
      [I-D.desruisseaux-caldav-sched] to the CalDAV [RFC 4791]
      protocol.
    </paragraph>
  </functionalspec>
  <security>
    See <xref type="rfc" data="rfcTHIS" />, Section 4.
  </security>
  <usage>COMMON</usage>
  <registrationdocs>
    <xref type="rfc" data="rfcTHIS" />
  </registrationdocs>
  <requesters>
    <xref type="person" data="Bernie_Hoeneisen"/>
    <xref type="person" data="Rohan_Mahy" />
  </requesters>
</record>
```

2.5. ical-sched:https (new)

```
<record>
  <!-- ical-sched:https -->
  <class>Application-Based, Common</class>
  <type>ical-sched</type>
  <subtype>https</subtype>
  <urischeme>https</urischeme>
  <functionalspec>
    <paragraph>
      This Enumservice indicates that the resource identified can be
      addressed by the associated URI used for scheduling using
      Internet calendaring via the scheduling extension
      [I-D.desruisseaux-caldav-sched] to the CalDAV [RFC 4791]
      protocol.
    </paragraph>
  </functionalspec>
  <security>
    See <xref type="rfc" data="rfcTHIS" />, Section 4.
  </security>
  <usage>COMMON</usage>
  <registrationdocs>
    <xref type="rfc" data="rfcTHIS" />
  </registrationdocs>
  <requesters>
    <xref type="person" data="Bernie_Hoeneisen" />
    <xref type="person" data="Rohan_Mahy" />
  </requesters>
</record>
```

2.6. Requesters' Addresses

```
<people>

  <person id="Rohan_Mahy">
    <name>Rohan Mahy</name>
    <org>Unaffiliated</org>
    <uri>mailto:rohan&ekabal.com</uri>
    <updated>2009-10</updated>
  </person>

  <person id="Bernie_Hoeneisen">
    <name>Bernie Hoeneisen</name>
    <org>Ucom Standards Track Solutions GmbH</org>
    <uri>mailto:bernhard.hoeneisen@ucom.ch</uri>
    <updated>2011-03</updated>
  </person>

</people>
```

3. Examples

TODO: More Examples

```
$ORIGIN 3.2.1.0.5.5.5.2.1.2.1.e164.arpa.
@ NAPTR 10 100 "u" "E2U+ical-access:https" \
   "!^.*$!https://cal.example.com/home/alice/calendars/!" .

$ORIGIN 3.2.1.0.5.5.5.2.1.2.1.e164.arpa.
@ NAPTR 20 100 "u" "E2U+ical-sched:mailto" \
   "!^.*$!mailto:alice@example.com!" .
```

4. Security Considerations

The Domain Name System (DNS) does not make policy decisions about which records it provides to a DNS resolver. All DNS records must be assumed to be available to all inquirers at all times. The information provided within an ENUM record set must therefore be considered open to the public -- which is a cause for some privacy considerations.

Revealing a calendaring URI by itself is unlikely to introduce many privacy concerns, although, depending on the structure of the URI, it might reveal the full name or employer of the target. The use of anonymous URIs mitigates this risk.

As ENUM uses DNS, which in its current form is an insecure protocol, there is no mechanism for ensuring that the answer returned to a query is authentic. An analysis of threats specific to the dependence of ENUM on the DNS is provided in RFC 6116 [I-D.ietf-enum-3761bis], and a thorough analysis of threats to the DNS itself is covered in [RFC3833]. Many of these problems are prevented when the resolver verifies the authenticity of answers to its ENUM queries via DNSSEC (DNS Security, [RFC4035]) in zones where it is available.

More serious security concerns are associated with potential attacks against an underlying calendaring system (for example, unauthorized modification or viewing). For this reason, iTIP discusses a number of security requirements (detailed in [RFC5546]) that call for authentication, integrity and confidentiality properties, and similar measures to prevent such attacks. Any calendaring protocol used in conjunction with a URI scheme currently meets these requirements. The use of CalDAV with the 'https:' scheme makes use of TLS (Transport Layer Security, [RFC5246]) to provide server authentication, confidentiality, and message integrity.

Unlike a traditional telephone number, the resource identified by a calendaring URI is often already guessable, and it often requires that users provide cryptographic credentials for authentication and authorization before calendar data can be exchanged. Despite the public availability of ENUM records, the use of this information to reveal an unprotected calendaring resource is unlikely in practice.

5. DNS Considerations

These Enumservice registrations add no specific issues related to DNS.

6. IANA Considerations

This document requests registration of the "ical-sched" and "ical-access" Enumservices according to the definitions in Section 2 of this document and RFC 6117 [I-D.ietf-enum-enumservices-guide].

7. Acknowledgments

The author would like to thank Alfred Hoenes for providing feedback resulting in improvement of this document.

As this document obsoletes [RFC5333], please see also the Authors'

and Acknowledgments sections in RFC 5333 for additional acknowledgments.

8. References

8.1. Normative References

[I-D.desruisseaux-caldav-sched]
Daboo, C. and B. Desruisseaux, "CalDAV Scheduling Extensions to WebDAV", draft-desruisseaux-caldav-sched-09 (work in progress), October 2010.

[I-D.ietf-enum-3761bis]
Bradner, S., Conroy, L., and K. Fujiwara, "The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)", draft-ietf-enum-3761bis-09 (work in progress), June 2010.

[I-D.ietf-enum-enumservices-guide]
Hoeneisen, B., Mayrhofer, A., and J. Livingood, "IANA Registration of Enumservices: Guide, Template and IANA Considerations", draft-ietf-enum-enumservices-guide-22 (work in progress), October 2010.

[I-D.ietf-enum-enumservices-transition]
Hoeneisen, B. and A. Mayrhofer, "Update of legacy IANA Registrations of Enumservices", draft-ietf-enum-enumservices-transition-06 (work in progress), July 2010.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC2447] Dawson, F., Mansour, S., and S. Silverberg, "iCalendar Message-Based Interoperability Protocol (iMIP)", RFC 2447, November 1998.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC4791] Daboo, C., Desruisseaux, B., and L. Dusseault, "Calendaring Extensions to WebDAV (CalDAV)", RFC 4791,

March 2007.

[RFC4918] Dusseault, L., "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, June 2007.

[RFC5333] Mahy, R. and B. Hoeneisen, "IANA Registration of Enumservices for Internet Calendaring", RFC 5333, October 2009.

[RFC5545] Desruisseaux, B., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, September 2009.

[RFC5546] Daboo, C., "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 5546, December 2009.

[RFC6068] Duerst, M., Masinter, L., and J. Zawinski, "The 'mailto' URI Scheme", RFC 6068, October 2010.

8.2. Informative References

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC3283] Mahoney, B., Babics, G., and A. Taler, "Guide to Internet Calendaring", RFC 3283, June 2002.

[RFC3833] Atkins, D. and R. Austein, "Threat Analysis of the Domain Name System (DNS)", RFC 3833, August 2004.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

Appendix A. Document Changelog

[RFC Editor: This section is to be removed before publication]

draft-hoeneisen-rfc5333bis-00 (compared to RFC 5333)

- bernie: Updated References: 2445 -> 5545, 2446 -> 5546
- bernie: Updated existing Enumservices to be in line with RFC 6117
- bernie: Added new Enumservices for WebDAV scheduling

- bernie: several editorials

Appendix B. Open Issues

[RFC Editor: This section should be removed before publication]

- Major: Do we need "ical-sched:http"?
- Minor: References in citation: should those be RFC244[5-6] or RFC554[5-6]?
- Minor: Reference to RFC 5689 (Updates RFC 4791) needed?
- Minor: Should Rohan be listed as author?
- Minor: More examples

Author's Address

Bernie Hoeneisen
Ucom Standards Track Solutions GmbH
CH-8000 Zuerich
Switzerland

Phone: +41 44 500 52 44
Email: bernie@ietf.hoeneisen.ch (bernhard.hoeneisen AT ucom.ch)
URI: http://www.ucom.ch/

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 11, 2012

P. Hoffman
VPN Consortium

September 8, 2011

Specifying That a Server Supports TLS
draft-hoffman-server-has-tls-05

Abstract

A server that hosts applications that can be run with or without TLS may want to communicate with clients whether the server is hosting an application only using TLS or also hosting the application without TLS. Many clients have a policy to try to set up a TLS session but fall back to insecure if the TLS session cannot be set up. If the server can securely communicate whether or not it can fall back to insecure tells such a client whether or not they should even try to set up an insecure session with the server. This document describes the use cases for this type of communication and a secure method for communicating that information.

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 March 11, 2012.

Copyright Notice

1. Introduction

Most client-server applications standardized in the IETF have two modes: an insecure mode that involves no authentication or integrity protection, and a secure mode that requires (at a minimum) that the client authenticate the server and set up a communication channel with integrity protection. In most cases, the secure mode is achieved by starting a TLS [RFC5246] session and, when successful, running the insecure mode inside of it.

People within the IETF and application developers have historically had widely varying views on what a client should and should not do about the two modes. Phrases like "assured security" and "client flexibility" are used, often without clear definition. Deployed clients and servers from different vendors act differently for the two modes, often relegating the control of the two modes to "advanced" configuration options (if such control is given to the user at all).

Section 2 of this document lays out the choices for clients and servers for handling the two modes in different circumstances, and gives specific semantics for each type of client and server. Section 3 gives a protocol for a domain owner to specify whether they offer one or both modes for any given application, and to specify a client policy preference. Section 4 defines how to implement various policies using the protocol. Using the protocol given here, a server can completely specify what it offers and allows a client to reliably choose which mode it wants to use.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Security Options for Clients and Servers

This section describes the different types of clients and servers that deal with insecure protocols that can be secured by wrapping the protocol in TLS. It also describes the types of security policies that those clients and servers can embody. It explicitly does not argue that one policy is better than another in any particular

environment; instead, it assumes that the server operator and the client implementor (and, hopefully, the human operating the client) can make that decision themselves if given the proper tools.

This discussion assumes a client-server protocol that is defined for an insecure fashion, and is also defined for a secure fashion that uses a TLS session for security. For example, "HTTP run over port 80" and "HTTP-in-TLS run over port 443" would meet this definition; "SMTP without STARTTLS" and "SMTP with STARTTLS" (see [RFC3207]) would also meet this definition. Some peer-to-peer protocols might meet this definition if the startup actions resemble the typical client-server interaction, but this discussion makes no extra attempt to cover such protocols.

Given a particular client application configuration, there are three interesting types of clients:

Insecure Only (CIO) -- The client is configured to only attempt communication for the application in its insecure form. For example, a POP client might be configured to only try insecure POP on port 110.

Secure Only (CSO) -- The client is configured to only attempt communication for the application in its secure, TLS-wrapped form. For example, a POP client might be configured to only try secure POP on port 995.

Allows Fallback From Secure to Insecure (CFB) -- The client is configured to attempt communication for the application in its secure, TLS-wrapped form, but if it fails to set up a TLS session, the client will attempt to attempt communication to the same server using the insecure form. This configuration may be offered for reasons such as if the client doesn’t trust the CA that the server uses to identify itself.

Given a particular server configuration, there are three interesting types of servers:

Insecure Only (SIO) -- The server responds without TLS on the main port for the application. A host for a web server that only responds to HTTP requests on port 80 is an example of this.

Secure Only (SSO) -- The server responds using TLS on the TLS-specific port for the application. For example, a host for a web server only responds to HTTP requests on port 443. Alternately, if the application supports in-band security update (such as STARTTTLS for SMTP), the server responds on the normal port, tries to establish a TLS session, and does not proceed with the protocol

if a TLS session cannot be established.

Serves Both Secure and Insecure (SSB) -- The server responds without TLS on the main port for the application *and* responds using TLS on the TLS-specific port for the application, such as both ports 80 and 443 for HTTP. Alternately, if the application supports in-band security update (such as STARTTTLS for SMTP), the server responds on the normal port, tries to establish a TLS session, and proceeds with the normal protocol if a TLS session cannot be established.

It is expected that client configuration will be per-host. That is, a client that is CSO for some hosts might be CFB for other hosts. The server configuration, of course, applies to all clients accessing it.

In this taxonomy, a CIO can always communicate with an SIO and SSB. A CSO can communicate with an SSO, and can communicate with an SSB as long as the TLS session is set up successfully. A CFB can communicate with an SIO, an SSO, and an SSB.

Given this, a host that only allows clients to use the secure form of a protocol MUST only be configured to be SSO; a client that wants to only communicate with a server securely MUST only be configured to be CSO.

Note that a host might want to serve both insecure and secure form of a protocol, but wants clients to only use the secure form. For example, the insecure form might immediately do an upgrade to the secure form, or it might do a protocol-based redirection to a server doing the secure form. Such a host will want to be able to indicate that, even if it has both secure and insecure ports for a protocol open, it wants clients that can be configured as CFB or CSO to only be configured as CSO.

This taxonomy exposes a problem with the way that clients and servers interact today: a CIO that starts an insecure communication with a server, or a CFB that falls back to insecure communication with a server, has no idea whether the site they wish to communication with even hosts an insecure server. The server might be configured to be any of SIO or SSO or SSB, but the client cannot tell. If a CIO or CFB client knows ahead of time that a host did not support insecure communication, the client would not even start communication because it would either just waste time waiting for a timeout, or it would communicate with an impostor.

Note that the protocol described here is not a discovery protocol. It is perfectly reasonable for a server to be running a service in an

insecure fashion, a secure fashion, or both, without using the protocol that is described here. The purpose of the protocol is to let a client find out, securely, whether a particular server protocol is being run securely, not whether it is being run at all.

3. The HASTLS Resource Record

The HASTLS resource record type, whose value is TBD1, lists an insecure/secure port pair that is served on the host named by the domain name for the application and protocol given in the query. It only applies to applications that are secured with TLS, not to applications that have insecure and secure versions that use some other security protocol.

The presentation format is:

_appname._protoname.hostname IN HASTLS ins-port sec-port pol-pref

The application name ("appname") and protocol name ("protoname") being queried are the same as are used in the SRV RRtype described in [RFC2782] The insecure port number (called "ins-port"), the secure port number (called "sec-port"), and the client policy preference (called "pol-pref") are each two-octet positive integers.

If a server does not offer one of the the two services, that service is indicated by port 0. For protocols that use in-band signaling for security upgrades, "ins-port" and "sec-port" have the same value. A HASTLS record MUST NOT have both the "ins-port" and "sec-port" set to 0.

A query for a particular application MAY return more than one HASTLS resource record, and conformant clients MUST be able to process multiple responses from a single query. For example, a site that offers HTTP on both port 80 and port 8080 might return two records, one for port 80 and its secure counterpart (if any), and one for protport 8080 and its secure counterpart (if any).

The HASTLS record is not useful for service discovery. Clients MUST NOT make any assumptions about an application for which there is not a HASTLS record; the lack of a HASTLS record for a particular application says nothing about whether or not the service is offered on the host on a specific port.

The client policy preference octet specifies the host’s preference for client policy. It has two possible values:

0 -- The host's administrator has no client policy preference for this protocol.

1 -- If the client could be configured as either CFB or CSO for this protocol, and the host's administrator was able to configure the client for the protocol, that administrator would configure the client as CSO. Stated another way, the host's administrator does not want any CFB client to access the host for this protocol.

Note that specifying 1 for the client policy preference when a host does not support a protocol securely makes no sense, but it also does no harm. Further, for a host that is SSO, both policy preferences have an identical result.

For example, the server at www.example.com offers SMTP both securely and insecurely. The host's SMTP administrator has a client policy preference that CFB clients not access the host. The HASTLS record would look like:

```
_smtp._tcp.www.example.com IN HASTLS 25 25 1
```

Another example is the server at www.example.com offering HTTP only securely. The resulting HASTLS records could be either:

```
_http._tcp.www.example.com IN HASTLS 0 443 0
    or
_http._tcp.www.example.com IN HASTLS 0 443 1
```

[[NEED TO ADD: wire format.]]

4. Implementing Policy with HASTLS

Servers that have a policy to declare the server as SIO, SSO, or SSB can use HASTLS to announce that policy for each application it serves. A server whose policy is that it is an SIO would set the ins-port to a non-zero number and the sec-port to 0. A server whose policy is that it is an SSO would set the ins-port to 0 and the sec-port to a non-zero number. A server whose policy is that it is an SSB would set both the ins-port and sec-port to a non-zero number.

The conformance requirements for a client using the HASTLS record depend on the policy configured for the client or the server:

- A client communicating with a server that has set its client policy preference to 1 MUST NOT try to communicate insecurely with that server, even if the server has set the ins-port to a non-zero number; this is the equivalent of temporarily setting its policy

for the server to CSO for that application. This temporary policy based on the server's client policy preference overrides any others for the server.

- A client whose policy is that it is a CIO MUST NOT try to communicate insecurely with a server that has the ins-port set to 0.
- A client whose policy is that it is a CSO MUST only try to communicate securely with a server that has the sec-port set to a non-zero number.
- A client whose policy is that it is a CSO MUST NOT try to communicate with the server if an ins-port value is given.
- A client whose policy is that it is a CFB MUST NOT try to communicate securely with a server that has the sec-port set to 0.
- A client whose policy is that it is a CFB MUST NOT try to communicate insecurely with a server that has the ins-port set to 0.
- A client whose policy is that it is a CFB trying to communicate with a server whose sec-port is set to a non-zero number SHOULD first try to communicate securely over the secure port unless it knows from other sources that the TLS session will not be set up properly.

5. IANA Considerations

This document requests that IANA allocate a new DNS resource record type called HASTLS from the data types range; it will have the value TBD1.

Submission template:

A. Submission Date: Date of this document
B. Submission Type: New RRTYPE
C. Contact Information for submitter: Author of this document
D. Motivation for the new RRTYPE application: Contents of this document
E. Description of the proposed RR type: Contents of this document
F. What existing RRTYPE or RRTYPEs come closest to filling that need and why are they unsatisfactory: None are even close to that given in this document.
G. What mnemonic is requested for the new RRTYPE (optional): HASTLS
H. Does the requested RRTYPE make use of any existing IANA Registry or require the creation of a new IANA sub-registry in DNS Parameters: No
I. Does the proposal require/expect any changes in DNS servers/resolvers that prevent the new type from being processed as an unknown RRTYPE (see [RFC3597]): No
J. Comments: None

6. Security Considerations

In order to prevent a man-in-the-middle (MITM) attack where the attacker can change DNS responses, the data in the HASTLS record needs to be received securely by a DNS requester, such as through validated DNSSEC. At this time, there is no common method for a application client to know whether or not the data it receives from the DNS has been protected with DNSSEC unless that application has a validating resolver, or has an API to the operating system and the operating system has a validating resolver.

Some MITM attacks can do a partial impersonation of the secure server if the client does not have good indications when an SSL connection is established (see, especially, the "sslstrip" attack described in various places in 2009). This protocol cannot protect against all such attacks because the attack is actually on the user interface, not the security of the TLS connection.

If the HASTLS information is received by the client system without security, an attacker could change the HASTLS information to fool the client into thinking that a host provides insecure application services and/or does not provide secure application services. Thus, cryptographic protection of the contents of the HASTLS information (such as with DNSSEC) is mandatory.

7. References

7.1. Normative References

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

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

7.2. Informative References

[RFC3207] Hoffman, P., "SMTP Service Extension for Secure SMTP over Transport Layer Security", RFC 3207, February 2002.

Appendix A. Protocols That Have Both Second Ports and Upgrade Paths

There is a significant open issue with the current proposal: some protocols (hopefully few) have multiple ways of using TLS. One that was mentioned is HTTP, which has both "start TLS on TCP port 443" and "use STARTTLS on TCP port 80 as described in RFC 2817". POP and IMAP have the same issue.

One possibility is for this protocol to say "always use the second port" or "always use in-stream upgrade". This would give the client unambiguous instructions, but would not work for servers that do not implement the specified option.

Another possibility is to have the HASTLS response indicate which way the secure version is supported. This is more flexible than the first proposal, but also more complicated for the client. It also is possibly overkill if there are only two or three protocols of interest.

This issue needs to be resolved before the proposal is finished.

Author's Address

Paul Hoffman
VPN Consortium

Email: paul.hoffman@vpnc.org

Individual submission
Internet-Draft
Intended status: BCP
Expires: January 12, 2012

M. Kucherawy
Cloudmark, Inc.

July 11, 2011

Best Current Practices for Handling of Malformed Messages
draft-kucherawy-mta-malformed-03

Abstract

The email ecosystem has long had a very permissive set of common processing rules in place, despite increasingly rigid standards governing its components, ostensibly to improve the user experience. The handling of these come at some cost, and various components are faced with decisions about whether or not to permit non-conforming messages to continue toward their destinations unaltered, adjust them to conform (possibly at the cost of losing some of the original message), or outright rejecting them.

This memo includes a collection of the best current practices in a variety of such situations, to be used as implementation guidance. It must be emphasized, however, that the intent of this memo is not to standardize malformations or otherwise encourage their proliferation. The messages that are the subject of this memo are manifestly malformed, and the code and culture that generates them needs to be fixed. Nevertheless, many malformed messages from otherwise legitimate senders are in circulation and will be for some time and, unfortunately, commercial reality shows that we cannot simply reject or discard them. Accordingly, this memo presents recommendations for dealing with them in ways that seem to do the least additional harm until the infrastructure is tightened up to match the standards.

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.

Table of Contents

1. Introduction

1.1. The Purpose Of This Work

The history of email standards, going back to [RFC822] and beyond, contains a fairly rigid evolution of specifications. But implementations within that culture have also long had an undercurrent known formally as the robustness principle, but also known informally as Postel's Law: "Be conservative in what you do, be liberal in what you accept from others."

In general, this served the email ecosystem well by allowing a few errors in implementations without obstructing participation in the game. The proverbial bar was set low. However, as we have evolved into the current era, some of these lenient stances have begun to expose opportunities that can be exploited by malefactors. Various email-based applications rely on strong application of these standards for simple security checks, while the very basic building blocks of that infrastructure, intending to be robust, fail utterly to assert those standards.

This memo presents some areas in which the more lenient stances can provide vectors for attack, and then presents the collected wisdom of numerous applications in and around the email ecosystem for dealing with them to mitigate their impact.

1.2. Not The Purpose Of This Work

It is important to understand that this work is not an effort to endorse or standardize certain common malformations. The code and culture that introduces such messages into the mail stream needs to be repaired, as the security penalty now being paid for this lax processing arguably outweighs the reduction in support costs to end users who are not expected to understand the standards. However, the reality is that this will not be fixed quickly.

Given this, it is beneficial to provide implementers with guidance about the safest or most effective way to handle malformed messages when they arrive, taking into consideration the tradeoffs of the choices available especially with respect to how various actors in the email ecosystem respond to such messages in terms of handling, parsing, or rendering to end users.

2. Keywords

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 [KEYWORDS].

3. Background

The reader would benefit from reading [EMAIL-ARCH] for some general background about the overall email architecture. Of particular interest is the Internet Message Format, detailed in [MAIL]. Throughout this document, the use of the term "messsage" should be assumed to mean a block of text conforming to the Internet Message Format.

4. Internal Representations

Any agent handling a message could have one or two (or more) distinct representations of a message it is handling. One is an internal representation, such as a block of storage used for the header and a block for the body. These may be sorted, encoded, decoded, etc. as per the needs of that particular module. The other is the representation that is output to the next agent in the handling chain. This might be identical to the version that is input to the module, or it might have some changes such as added or reordered header fields, body modifications to remove malicious content, etc.

In some cases, advice is provided only for internal representations. However, there is often occasion to mandate changes to the output as well.

5. Mail Submission Agents

Within the email context, the single most influential component that can reduce the presence of malformed items in the email system is the Mail Submission Agent (MSA). This is the component that is essentially the interface between end users that create content and the mail stream.

The lax processing described earlier in the document creates a high support and security cost overall. Thus, MSAs MUST evolve to become more strict about enforcement of all relevant email standards, especially [MAIL] and the [MIME] family of documents.

Relay Mail Transport Agents (MTAs) SHOULD also be more strict; although preventing the dissemination of malformed messages is desirable, the rejection of such mail already in transit also has a support cost, namely the creation of a [DSN] that many end users might not understand.

6. Header Anomalies

This section covers common syntactical and semantic anomalies found in headers of messages, and presents preferred mitigations.

6.1. Non-Header Lines

It has been observed that some messages contain a line of text in the header that is not a valid message header field of any kind. For example:

```
From: user@example.com
To: userpal@example.net
Subject: This is your reminder
about the football game tonight
Date: Wed, 20 Oct 2010 20:53:35 -0400

Don’t forget to meet us for the tailgate party!
```

The cause of this is typically a bug in a message generator of some kind. If the fourth line was intended to be a continuation of the third, it should be indented by whitespace as set out in Section 2.2.3 of [MAIL].

This anomaly has varying impacts on processing software, depending on the implementation:

1. some agents choose to separate the header of the message from the body only at the first empty line (i.e. a CRLF immediately followed by another CRLF);

2. some agents assume this anomaly should be interpreted to mean the body starts at line four, as the end of the header is assumed by encountering something that is not a valid header field or folded portion thereof;

3. some agents assume this should be interpreted as an intended header folding as described above;

4. some agents reject this outright as line four is neither a valid header field nor a folded continuation of a header field prior to an empty line.

This can be exploited if it is known that one message handling agent will take one action while the next agent in the handling chain will take another. For example, a filter trained to detect malicious body anomalies (e.g. references to dangerous web sites) that is fed by a Mail Transfer Agent (MTA) implementing (1) above might not get the opportunity to identify something dangerous in a message if it is unaware of the anomaly and does not itself check for it.

Consensus indicates the preferred implementation is to terminate header processing before the first character in line four, as

described in (2) above. Thus, a module compliant with this specification MUST terminate header processing upon encountering the first line of text that is not a valid header field. That is, all data after that point in the input MUST NOT be considered part of the header of the message. If that line is not an empty line, an empty line MUST be inserted at that point in the emitted version of the message being processed.

It should be noted that a few implementations make choice (4) above since any reputable message generation program will get header folding right, and thus anything so blatant as this malformation is likely an error caused by a malefactor.

6.2. Header Malformations

There are various malformations that exist. A common one is insertion of whitespace at unusual locations, such as:

```
From: user@example.com
To: userpal@example.net
Subject: This is your reminder
MIME-Version : 1.0
Content-Type: text/plain
Date: Wed, 20 Oct 2010 20:53:35 -0400

Don’t forget to meet us for the tailgate party!
```

Note the addition of whitespace in line four after the header field name but before the colon that separates the name from the value.

The acceptance grammar of [MAIL] permits that extra whitespace, so it cannot be considered invalid. However, a consensus of implementations prefers to remove that whitespace. There is no perceived change to the semantics of the header field being altered as the whitespace is itself semantically meaningless. Thus, a module compliant with this memo MUST remove all whitespace after the field name but before the colon, and MUST emit that version of that field on output.

6.3. Header Field Counts

Section 3.6 of [MAIL] prescribes specific header field counts for a valid message. Few agents actually enforce these in the sense that a message whose header contents exceed one or more limits set there are generally allowed to pass; they may add any required fields that are missing, however.

Also, few agents that use messages as input, including Mail User

Agents (MUAs) that actually display messages to users, verify that the input is valid before proceeding. Two popular open source filtering programs and two popular Mailing List Management (MLM) packages examined at the time this memo was drafted select either the first or last instance of a particular field name, such as From, to decide who sent a message. Absent enforcement of [MAIL], an attacker can craft a message with multiple fields if that attacker knows the filter will make a decision based on one but the user will be shown the other.

This situation is exacerbated when a claim of message validity is inferred by something like a valid [DKIM] signature. Such a signature might cover one instance of a constrained field but not another, and a naive consumer of DKIM's output, not realizing which one was covered by a valid signature, presume the wrong one was the "good" one. An MUA, for example could show the first of two From fields as "good" or "safe" while the DKIM signature actually only verified the second.

Thus, an agent compliant with this specification MUST enact one of the following:

1. reject outright or refuse to process further any input message that does not conform to Section 3.6 of [MAIL];

2. remove or, in the case of an MUA, refuse to render any instances of a header field whose presence exceeds a limit prescribed in Section 3.6 of [MAIL] when generating its output;

3. alter the name of any header field whose presence exceeds a limit prescribed in Section 3.6 of [MAIL] when generating its outputso that later agents can produce a consistent result.

7. MIME Anomalies

[MIME], et seq, define a mechanism of message extensions for providing text in character sets other than ASCII, non-text attachments to messages, multi-part message bodies and similar facilities.

Some anomalies with MIME-compliant generation are also common. This section discusses some of those and presents preferred mitigations.

7.1. Missing MIME-Version Field

Any message that uses [MIME] constructs is required to have a MIME-Version header field. Without them, the Content-Type and associated fields have no semantic meaning.

It is often observed that a message has complete MIME structure, yet lacks this header field.

As described at the end of Section 6.1, this is not expected from a reputable content generator and is often an indication of mass-produced spam or other undesirable messages.

Therefore, an agent compliant with this specification MUST internally enact one or more of the following in the absence of a MIME-Version header field:

1. Ignore all other MIME-specific fields, even if they are syntactically valid, thus treating the entire message as a single-part message of type text/plain;

2. Remove all other MIME-specific fields, even if they are syntactically valid, both internally and when emitting the output version of the message;

3. Rename all other MIME-specific fields, even if they are syntactically valid, both internally and when emitting the output version of the message.

8. IANA Considerations

This memo contains no actions for IANA.

9. Security Considerations

The discussions of the anomalies above and their prescribed solutions are themselves security considerations. The practises enumerated in this memo are generally perceived to resolve security considerations that already exist rather than introducing new ones.

10. References

10.1. Normative References

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

[MAIL] Resnick, P., "Internet Message Format", RFC 5322, October 2008.

10.2. Informative References

[DKIM] Allman, E., Callas, J., Delany, M., Libbey, M., Fenton, J., and M. Thomas, "DomainKeys Identified Mail (DKIM)

Signatures", RFC 4871, May 2007.

[DSN] Moore, K. and G. Vaudreuil, "An Extensible Message Format for Delivery Status Notifications", RFC 3464, January 2003.

[EMAIL-ARCH] Crocker, D., "Internet Mail Architecture", RFC 5598, July 2009.

[MIME] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[RFC822] Crocker, D., "Standard for the Format of Internet Text Messages", RFC 822, August 1982.

Appendix A. Examples

Examples, if needed, can go here.

Appendix B. Acknowledgements

The author wishes to acknowledge the following for their review and constructive criticism of this proposal: (names)

Author’s Address

Murray S. Kucherawy
Cloudmark, Inc.
128 King St., 2nd Floor
San Francisco, CA 94107
US

Phone: +1 415 946 3800
EMail: msk@cloudmark.com

Network Working Group
Internet-Draft
Intended status: BCP
Expires: July 31, 2011

E. Lear
Cisco Systems GmbH
P. Eggert
UCLA
January 27, 2011

IANA Procedures for Maintaining the Timezone Database
draft-lear-iana-timezone-database-02

Abstract

ATTENTION: This memo contains a DRAFT proposal for the IANA to assume operational responsibilities relating to the management of the Timezone (TZ) Database. The authors seek comment and review of this proposal. No action will be taken without rough consensus of the TZ community.

Timezone information serves as a basic protocol element in protocols, such as the calendaring suite and DHCP. The Timezone (TZ) Database specifies the indices used in various protocols, as well as their semantic meanings, for all localities throughout the world. This database has been meticulously maintained and distributed free of charge by a group of volunteers, coordinated by a single volunteer who is now planning to retire. This memo specifies IANA procedures involved with maintenance of the TZ database and associated code, including how to submit proposed updates, how decisions for inclusion of those updates are made, and the selection of a designated expert BY AND FOR the timezone community. The intent of this memo is, to the extent possible, document existing practice and provide a means to ease succession.

1. Introduction

ATTENTION: This memo contains a DRAFT proposal for the IANA to assume operational responsibilities relating to the management of the Timezone (TZ) Database. The authors seek comment and review of this proposal. No action will be taken without rough consensus of the TZ community.

The IETF has specified several standards that make use of timezone information. Timezone names are used in DHCP to configure devices with correct local time [RFC4833]. Timezone names can appear in the TZID field of VEVENTs [RFC5545]. The normative reference for these values is the TZ Database [TZDB]. Since the early 1980s, that database, which has been in use on nearly all UNIX systems, Java systems, and other sorts of systems has been hosted at the National Institutes of Health. The database consists of both historic and current entries for geographies throughout the world. Associated with the database is a reference implementation of functions that can be used to convert time values.

The database has been maintained by volunteers who participate in a mailing list that is also hosted at the NIH. The database itself is updated approximately twenty times per year, depending on the year, based on information these experts provide to the maintainer. Arthur David Olson has maintained the database, coordinated the mailing list, and provided a release platform since the database's inception. With his retirement now approaching it is necessary to provide a means for this good work to continue.

The IANA provides registry services to the Internet community. Those registries are coordinated by technical experts who are designated by the Internet Engineering Steering Group (IESG). The IANA is also

well suited as a distribution platform for the TZ Database itself.

The IANA has for quite some time had the capability to maintain designated expert mailing lists. The TZ mailing list would fit nicely just as such a list.

1.1. 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 RFC 2119 [RFC2119].

TZ Database: The TimeZone Database, sometimes referred to as the Olson Database. This database consists of information about offsets from UTC for different localities, including daylight saving time (DST) transition information.

TZ Coordinator: The person or people who maintain and manage release of the TZ Database. The TZ coordinator also has responsibility for maintaining the TZ mailing list. The TZ coordinator is an IANA Designated Expert, as defined in Section 3.2 of [RFC5226]. Roughly speaking, this means that the IESG will choose one or more experts to manage the TZ database, code, and mailing list.

TZ mailing list: The forum where matters relating to the TZ database and supporting code are discussed.

The rest of this document specifies the following:

1. Transferring and maintenance of the TZ mailing list;
2. Procedures for selecting a technical expert for the technical expert who will play the role of coordinator and release manager for the TZ database;
3. Procedures for updating the TZ database;
4. Maintenance and ownership of reference code; and
5. Ownership of the database.

2. The TZ Mailing List

For many years the TZ mailing list at the NIH has been the forum where discussion of changes to the TZ database and support files would take place. In addition, the TZ mailing list is used to announce releases of the database. Currently the TZ mailing list is administered by the TZ coordinator.

This list membership will be transitioned to the IANA mail server. The TZ coordinator will continue to manage the list. While the TZ coordinator may establish other rules of governance for the list, members of that list will be informed that a condition of

participating on the list is that all contributions to the list are released to the public domain, and that by placing their contribution in the public domain, contributors waive forever any intellectual property claims.

The list will be used just as it has been, to learn of, discuss, and confirm TZ definition changes, as well as an announcement list for new versions of the database.

3. Making Updates to the TZ Database

Updates to the TZ database are made by the TZ coordinator in consultation with the TZ mailing list. TZ coordinator is empowered to decide, as the designated expert, appropriate changes, but SHOULD take into account views expressed on the mailing list.

The TZ coordinator will also decide the timing of database releases. The release itself today consists of several archive files that are downloaded from a well known location.

Moving forward, the TZ database and supporting code SHOULD be signed prior to release using a well known key, along with any appropriate supporting information and distributed from a well known location that is advertised by IANA in a manner of its choosing.

The criteria for updates to the database are as follows:

1. New keys are only to be created when the region a key was envisioned to cover is not accurately reflected by an existing key.
2. In order to correct historical inaccuracies, a new key MAY be added when it is necessary to indicate what was the consensus view at given time and location. Such keys are usually not added when the inaccuracy was prior to 1970.
3. Changes to existing entries SHALL reflect the consensus on the ground in the region covered by that entry.

To be clear, the TZ coordinator SHALL NOT set timezone policy policy for a region but use judgment and whatever available sources exist to assess what the average person on street would think the time actually is, or in case of historical corrections, was.

4. Selecting or Replacing a TZ Coordinator

From time to time it will be necessary to appoint a new TZ Coordinator. This could occur for a number of reasons:

- The coordinator is retiring (as Arthur Olson is) or has announced that he or she will be unable to continue to perform the function;
- The coordinator is missing or has died;
- The coordinator is not performing the function in accordance with community wishes.

In any of these cases, members of the community should raise the issue on the TZ list. If a rough consensus can be formed easily, and quickly, then the results should be presented to the IESG for comment and review. The IESG selects the TZ coordinator(s). The IESG will use rough consensus of the TZ mailing list as their primary guide to further action, when it exists, and whatever other means they have at their disposal, when rough consensus cannot be found.

5. Appealing Database Decisions

The TZ coordinator makes decisions based on expertise, as well as with guidance from the TZ mailing lists. While individual decisions MAY be appealed to the IESG, the IESG MUST give great deference to the designated expert in its considerations. In particular, apellants MUST show material harm from the decision, and that the decision is materially in error. The IESG is not a normal avenue for appeals of specific decisions of the coordinator, but rather a last resort when a coordinator is thought not to be functioning in an appropriate way.

N.B., the coordinator is a function, and may be filled by one or more people, as the community sees fit.

6. Maintenance and Distribution of Reference Code

Currently the maintainer of the TZ database also maintains reference code, most of which is public domain. Several files from this software are currently distributed under license. Where they exist, licenses SHALL NOT be changed. IANA SHALL allow for the downloading of this reference code. The reference implementation shall be distributed along with an associated cryptographic signature of an identity that IANA will publish.

7. Database Ownership

The database itself is public domain. Should claims be made and substantiated against the database, the IANA will act in accordance with all competent court orders. No ownership claims will be made by IANA or the IETF Trust on the database or the code. Any person

making a contribution to the database or code waives all rights to future claims.

8. IANA Considerations

The IANA SHALL assist the IESG, as required, in filling of the TZ Coordinator, based on the procedures set forth above. The IANA SHALL act as a repository for the TZ database and associated reference code. The database coordinator SHALL be named by the IESG as described above, and will act as the maintainer of the database and code, as described above. The IANA SHALL provide the TZ coordinator with appropriate access to maintain the database, as well as necessary tooling that may be required, so long as no direct software costs are incurred. Both current and historical versions of the database will be stored and distributed via HTTP/HTTPs. IANA will be operationally responsible for the security of the system upon which the database resides.

The IANA SHALL also maintain a cryptographic identity that is used to sign the database, and that will survive a change of coordinators.

9. Security Considerations

The distribution of the database is currently not secured. This memo states that moving forward the TZ database SHOULD be distributed with a valid cryptographic signature.

10. Acknowledgments

The authors would like to thank the TZ mailing list for their remarkable achievements over the many years. Thanks also to Marshall Eubanks, S. Moonesamy, Peter Saint-Andre, Alexey Melenkov, Tony Finch, Elwin Davies, Alfred Hoenes, and Ted Hardie for the improvements they made to this document. A special acknowledgment should be given to Arthur David Olson for his excellent stewardship.

11. Normative References

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

[RFC4833] Lear, E. and P. Eggert, "Timezone Options for DHCP", RFC 4833, April 2007.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5545] Desruisseaux, B., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, September 2009.

[TZDB] Eggert, P. and A. Olson, "Sources for Time Zone and Daylight Saving Time Data", <http://www.twinsun.com/tz/tz-link.htm>.

Appendix A. Changes

- 02: Separate out from RFC5226 a bit; Simplify language around submissions; host list to IANA; spelling corrections; clarify here and there.
- 01: Proper reference to RFC5226, add acknowledgments, several rewordings.
- Initial Revision

Authors' Addresses

Eliot Lear
Cisco Systems GmbH
Richtistrasse 7
Wallisellen, ZH CH-8304
Switzerland

Phone: +41 1 878 9200
Email: lear@cisco.com

Paul Eggert
UCLA
Computer Science Department
4532J Boelter Hall
Los Angeles, CA 90095
USA

Phone: +1 310 267 2254
Email: eggert@cs.ucla.edu

Cloud Apps
Internet-Draft
Intended status: Informational
Expires: August 1, 2011

S. Ma, Ed.
L. Liang
J. Wang
ZTE Corporation
Jan 28, 2011

Survey of Virtual Desktop Infrastructure System
draft-ma-clouds-vdi-survey-00

Abstract

This document presents a survey of VDI (Virtual Desktop Infrastructure) system. Current popular VDI solutions are focused on, such as Microsoft RDS, Citrix XenDesktop, Redhat enterprise virtualization for desktops and VMware View. By analyzing the architecture and protocol flows of these solutions, the common features of VDI architecture and protocol are summarized.

Table of Contents

1. Introduction

VDI provides services for users to access their desktops remotely and users can choose PC, tablet computer or mobile phone as the client device. The OS (Operating System) and applications of the desktop are running on virtual machines which are deployed in Data Center. All desktops are managed under a central management system.

The benefits of VDI include reduction of IT cost and improvement of security. Centralized desktop management makes IT maintenance tasks more effective, for example backup and upgrade can be done on Data Center side. Since enterprise's data is kept in Data Center, it will not be lost even when client devices are broken. Users can access the data but can't copy it out of the datacenter, so the data security is guaranteed.

The mainstream VDI products include Microsoft's RDS, Citrix's XenDesktop, Redhat's Enterprise Virtualization for Desktops and VMware's VMware View. The architecture, features and VDI protocols of these products are analyzed in this document.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2. Terminology and Abbreviation

- Device redirection: In VDI system, remote desktop must provide the same user experience as local desktop, so the physical interfaces of client device, such as USB, audio/video interface and serial/ parallel ports should be accessed by Guest OS on remote desktop. So client device's interfaces must be redirected to remote virtual machine.
- GCC: Generic Conference Control
- GPU: Graphic Processor Unit
- Guest OS: An OS runs on top of a virtual machine manager.
- HDX: High-Definition user eXperience
- ICA: Independent Computing Architecture
- MCS: Multipoint Communication Service

- OS: Operation System
- PDU: Protocol Data Unit
- RD: Remote Desktop
- RDP: Remote Desktop Protocol
- RDS: Remote Desktop Service
- Resource pool: Resource pool is based on a large quantity of physical server. The computing and storage resources of the servers are managed as a whole, and it is a logical resource pool to users. Users need not to know which server the resource is on when he use it. Resource pool can improve resource's utilization and scalability, and resource will be managed more effectively.
- SPICE: Simple Protocol for Independent Computing Environment
- TPKT: ISO transport services on top of the TCP
- VDA: Virtual Desktop Agent
- VDI: Virtual Desktop Infrastructure
- VM: Virtual machine is a logical machine. Several virtual machines can share one physical machine's resource. The virtual CPU, memory and storage are allocated from physical resource by virtualization layer software, and they can be scheduled under certain policy.
- VMM: Virtual Machine Manager

2. Survey of VDI System

Remote terminal technology was brought into market very early, such as Microsoft's Terminal Services and RealVNC's VNC. Early products are used for server sharing, remote collaboration, and remote control. The messages transferred between server and client contain screen images and the keyboard/mouse events. VDI inherits from the remote terminal technology, with exception that the desktops are not running on physical servers, but in virtual machines. Virtual machines can be allocated on demand, so VDI is more flexible and has higher resource utilization. VDI also adopts variety of technologies to improve user experience, such as enhanced desktop display quality, supporting for USB/audio/video device redirection and etc.

2.1. Citrix XenDesktop

XenDesktop is Citrix's VDI solution. For Guest OS in XenDesktop, it could only be Microsoft's Windows. For Client device, it could support different platforms, like Windows, Linux and Mac OS. Virtualization platform can be Citrix's XenServer, or 3rd party's products, such as Microsoft Hyper-V and VMware ESX. The protocol between client and server is ICA (Independent Computing Architecture).

The architecture of XenDesktop is shown in Figure 1:

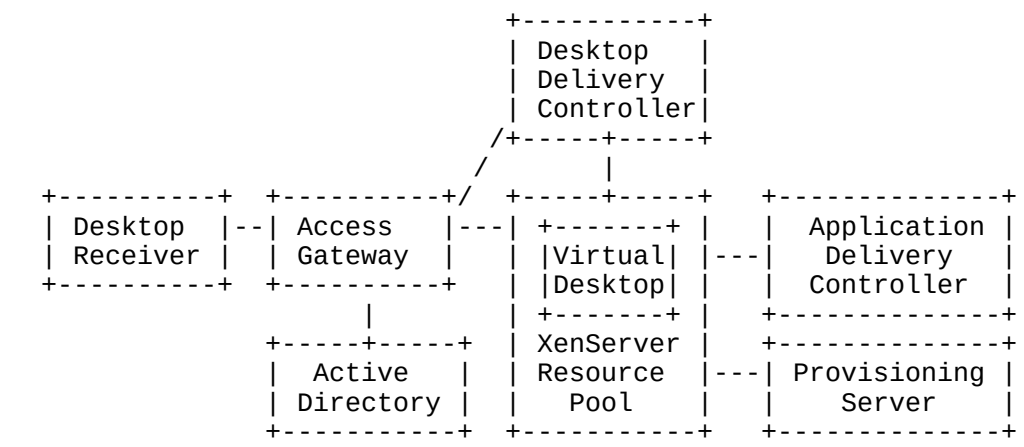

Figure 1 XenDesktop architecture

The key components of XenDesktop:

1. Desktop Receiver: The client-side software, which is running on client devices, connects to remote desktop through ICA protocol.

2. Access Gateway: Responsible for external user authentication. Before the external user accesses the desktop, he must connect to the Access Gateway and pass the authentication. External users's connection to Access Gateway is secured by SSL.

3. Desktop Delivery Controller: Responsible for management of virtual desktop and client connection. When a virtual desktop boots up, it will register with Desktop Delivery Controller. Desktop Delivery Controller will maintain the state of the desktop. When a client initiates a connection, Desktop Delivery Controller chooses an appropriate virtual desktop to connect. Finally it forwards the client's connection to the desktop.

4. XenServer Resource Pool: XenServer is Citrix's virtualization platform, provides virtual machine for virtual desktop and manages all of the virtual machines.

5. Provisioning Server: Responsible for providing OS images. OS images contain pre-configured profile, but applications are not included. When virtual desktop boots up, the OS image will be sent to virtual machine over network.

6. Application Delivery Controller: Responsible for application delivery. User's profile contains assigned application list, and user can choose which application to load.

7. Virtual Desktop Agent: Running on virtual desktop, which accepts the ICA connection from client.

8. Active Directory: Responsible for user authentication.

Besides above specific components, additional general components could be deployed, such as License server, DNS Server, DHCP Server and so on.

Before users access their virtual desktops, they must pass the authentication. Following is the procedure:

1. User launches the browser, and connects to Access Gateway. Access Gateway generates the login page;

2. User inputs credentials, and Access Gateway authenticates the user's credentials against Active Directory;

3. Upon successful authentication, Access Gateway connects to Desktop Delivery Controller;

4. Desktop Delivery Controller retrieves the credentials from Access Gateway, and authenticates the user against Active Directory;

5. After success authentication, Desktop Delivery Controller determines which virtual desktops are available for the user;

6. Access Gateway creates a web page containing a list of virtual desktops for the user, and sends the web page to user's browser.

After successful authentication, user can access virtual desktop by clicking the icon on the web page. Following is the procedure:

1. User clicks the virtual desktop’s icon on the web page, Access Gateway sends the information to Desktop Delivery Controller;

2. Desktop Delivery Controller checks the status of user’s virtual desktop. If there is no available desktop, Desktop Delivery Controller will request XenServer to launch one. The desktop then opens the ICA port to wait for client connections;

3. Desktop Delivery Controller creates an ICA file for the virtual desktop, and transfers the file to Access Gateway. Then Access Gateway transfers the ICA file to VDI client;

4. After receiving the ICA file, VDI client executes it and sends a connection request to Access Gateway;

5. Access Gateway forwards the connection request to virtual desktop;

6. When detecting the connection request, the virtual desktop sends the logon information to Desktop Delivery Controller for validation;

7. For valid validation, Desktop Delivery Controller fetches a license from the license server and sends credentials&license to virtual desktop. The connection is created after virtual desktop logon against Active Directory with the credentials.

To optimize user experience, XenDesktop provides HDX (High Definition eXperience) technology. HDX contains several techniques as following:

1. HDX MediaStream: Optimized for multimedia stream, the compressed multimedia packages are transferred in raw format to client device, and the client device plays them locally;

2. HDX RealTime: Enhances real-time voice and video using advanced encoding and streaming to ensure a no compromise end-user experience;

3. HDX Broadcast: Optimized for bandwidth to ensure high-performance of virtual desktops and applications over any network, including high-latency and low-bandwidth environments. User can set bandwidth limits and close some features to save bandwidth;

4. HDX Plug-n-Play: Enables devices redirection for USB, printers and peripherals;

5. HDX RichGraphics: Optimizes for 2D/3D applications by using the rending ability of server and client;

6. HDX WAN Optimization: Adopts compression and cache technologies to reduce bandwidth on the WAN.

2.2. Microsoft RDS

RDS (Remote Desktop Service) is next-generation product of Microsoft TS (Terminal Service). In Windows server 2008 R2, it is named as RDS. The protocol used in RDS between desktop client and server is RDP (Remote Desktop Protocol).

The architecture of RDS is presented in Figure 2:

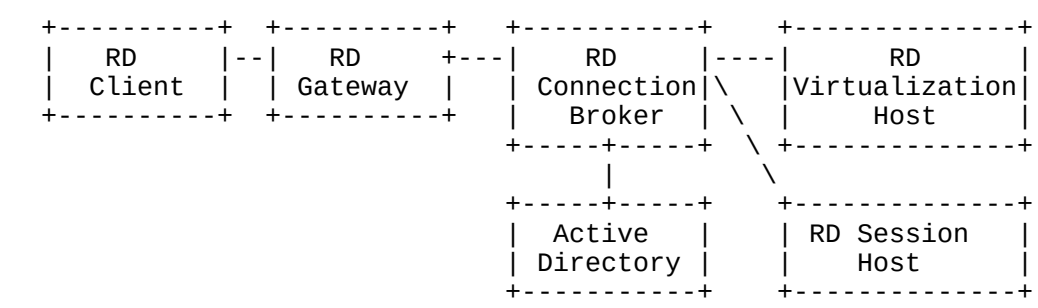

Figure 2 RDS architecture

The key components in RDS:

1. RD Client: Remote desktop client software running on client devices, which connects to virtual desktop using RDP protocol;

2. RD Gateway(RDG): RD Gateway authorizes external RD clients and provides secure connection for external RD client to access virtual desktop;

3. RD Connection Broker (RDCB): Responsible for assigning correct resource to RD clients and session load balancing. User can have a personal desktop, which means that the RDCB assigns the same virtual machine to the user whenever he logins. RDCB also can assign virtual machine dynamically. When user logs on, RDCB selects one from virtual machine pool and assign it to user. Every user shares the same virtual machine image;

4. Remote Desktop Session Host (RDSH): Responsible for hosting windows-based programs or the full Windows desktop for RD

clients. Users can connect to an RD Session Host server to run programs, save files, and use network resources on that server;

5. RD Virtualization Host(RDVH): RDVH integrates with Hyper-V to provide virtual machines for virtual desktops;

6. Active Directory: Responsible for user access authentication.

The connection procedure for RD Client is shown as below:

1. A RD Session Host works as a RD Redirector. When a client connects to RD Redirector, RD Redirector forwards the connection to RD Connection Broker;

2. RD Connection Broker queries personal desktop configuration from Active Directory;

3. RD Connection Broker retrieves user’s virtual machine from RD Virtualization Host. If user’s virtual machine is not running, RD Virtualization Host will launch it;

4. RD Connection Broker returns the information of virtual machine to RD Redirector;

5. RD Redirector forwards the information to RD Client,and redirects RD Client to virtual machine;

6. RD Client connects to virtual machine.

To improve user experience, RDS provides RemoteFX technology in RDP 7.1. RemoteFX contains a set of features shown as following:

1. Host side rendering: Allows graphics to be rendered on the host device instead of the client device. This enables support for all graphics types by sending highly compressed bitmap images to the endpoint device in an adaptive manner. This also allows the applications to run at full speed on the host computer by taking advantage of the GPU and the CPU, providing an experience which is similar to that of a local computer;

2. GPU virtualization: Supports several virtual GPU with one physical GPU, and each virtual GPU is used by one virtual machine;

3. Intelligent Screen Capture: Only the screen change will be encoded and transferred to client. Intelligent Screen Capture also tracks network speed and then dynamically adjusts according to the available bandwidth;

4. RemoteFX Encoder: Allows encoding to be done either on the processor, GPU, or dedicated hardware;

5. RemoteFX Decoder: Decodes bitmaps on the client computer which have been remoted from the virtual desktop to the client computer. The client computer can decode by using software in the GPU, processor, or by using a hardware decoder;

6. USB device redirection: Supports all usb devices redirection over RDP, no client side drivers needed.

2.3. Redhat Enterprise Virtualization for Desktops

Redhat Enterprise Virtualization for Desktops is Redhat's VDI solution,which supports both Windows and Linux desktop. The protocol between client and desktop is SPICE (Simple Protocol for Independent Computing Environment).

The architecture of Redhat Enterprise Virtualization for Desktops is shown in Figure 3:

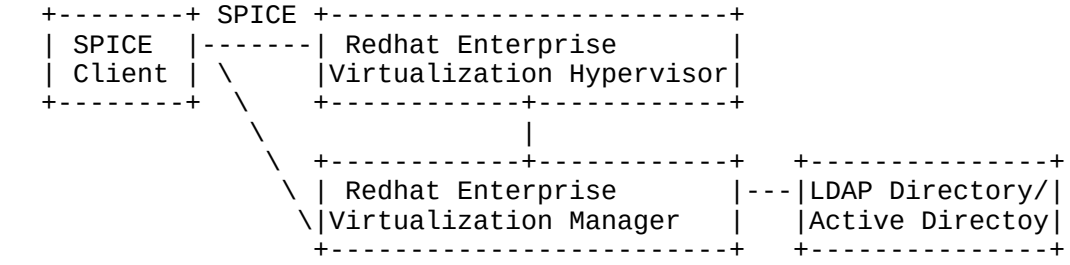

Figure 3 Redhat Enterprise Virtualization for Desktops architecture

The key components of Redhat Enterprise Virtualization for Desktops:

1. SPICE client: The client side software installed on client devices, which connects to virtual desktop using SPICE protocol;

2. SPICE Protocol: An open source VDI protocol, used between SPICE client and virtual desktop;

3. Red Hat Enterprise Virtualization Hypervisor(RHEV-H): Redhat virtualization platform to provide virtual machine for virtual desktops;

4. Red Hat Enterprise Virtualization Manager(RHEV-M): Management System for all components in Redhat's VDI solution;

5. LDAP Directory/Active Directory: Responsible for authentication.

The connection procedure for SPICE client is described as below:

1. User launches browser and connects to RHEV-M, and RHEV-M returns a logon page. Then user inputs corresponding ID and password information;

2. RHEV-M authenticates user against Active Directory;

3. After successful authentication, RHEV-M returns a web page containing a list of available virtual desktops for the user;

4. User selects the desktop by clicking the icon on the web page. Then RHEV-M requests RHEV-H to run the specified virtual desktop;

5. The SPICE client connects to the virtual desktop with SPICE protocol.

2.4. VMware View

VMware View is VMware's virtual desktop solution, which uses VMware vSphere as the virtualization platform. VMware View supports several VDI protocols, like Microsoft RDP, PCoIP and HP RGS protocols. For HP RGS, it can only be used for connecting HP Blades.

The architecture of VMware View is shown in Figure 4.

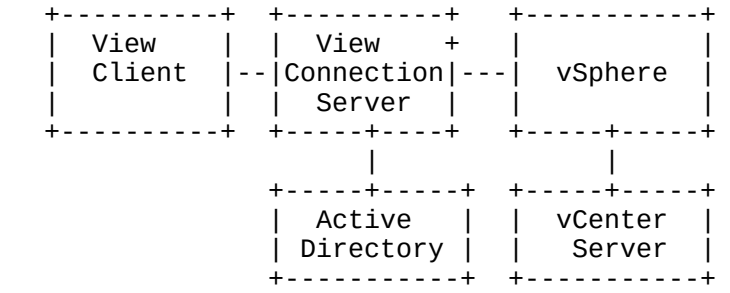

Figure 4 VMware View architecture

The key components of VMware View:

1. View client: The client software for accessing View desktops, which runs either on a Windows or Mac PC as a native application or on a thin client;

2. View Connection Server: View Connection Server acts as a broker for client connections. It authenticates users through Active Directory and directs the request to the appropriate virtual machine;

3. vSphere: The virtualization platform, which provides virtual machines for virtual desktops;

4. vCenter Server: Management System for configuration, deployment and virtual machine management;

5. View Agent: View Agent is installed on all virtual machines. This agent communicates with View Client to provide features such as connection monitoring, virtual printing, and access to locally connected USB devices;

VMware View has following features:

1. VMware View supports USB, printers and other peripherals redirection;

2. Users can configure the amount of bandwidth used by Adobe Flash content to improve the overall Web browsing experience and make other applications more responsive;

3. VMware View supports multimedia redirection, and the media file is send to client to be decoded locally;

4. VMware View supports single-sign-on, which means that users need to log in only once to access virtual desktop;

5. Besides Active Directory authentication, VMware also supports RSA SecurID and Smart Card authentication.

3. VDI Related Protocols

In this section, current hot-discussed VDI protocols, like RDP, SPICE and ICA, are investigated. For closed protocol, such as ICA, the information is collected from published documents on Internet. For each protocol, they are organized from perspective of: Basic architecture, Bandwidth saving and User experience improvement. Bandwidth saving and User experience improvement are two key factors for VDI protocol success.

3.1. T.120

T.120 is listed here for better understanding of Microsoft's RDP, which is described in next section.

3.1.1. Introduction

The T.120-series Recommendations, which are issued by ITU, collectively define a multipoint data communication service for use in multimedia conferencing environments. Depending on the type of T.120 implementations, the result product can make connections, transmit and receive data, and collaborate using compatible data conferencing features, such as program sharing, whiteboard conferencing, and file transfer.

The key functionalities of T.120 are:

1. Establish and maintain conferences without any platform dependence;

2. Manage multiple participants and programs;

3. Send and receive data accurately and securely over a variety of supported networking connections.

3.1.2. T.120 protocol suite

3.1.2.1. Basic architecture

The T.120 system model is comprised of a communications infrastructure and the application protocols that make use of it. Figure 5 shows the full model with both standardized and non-standardized applications.

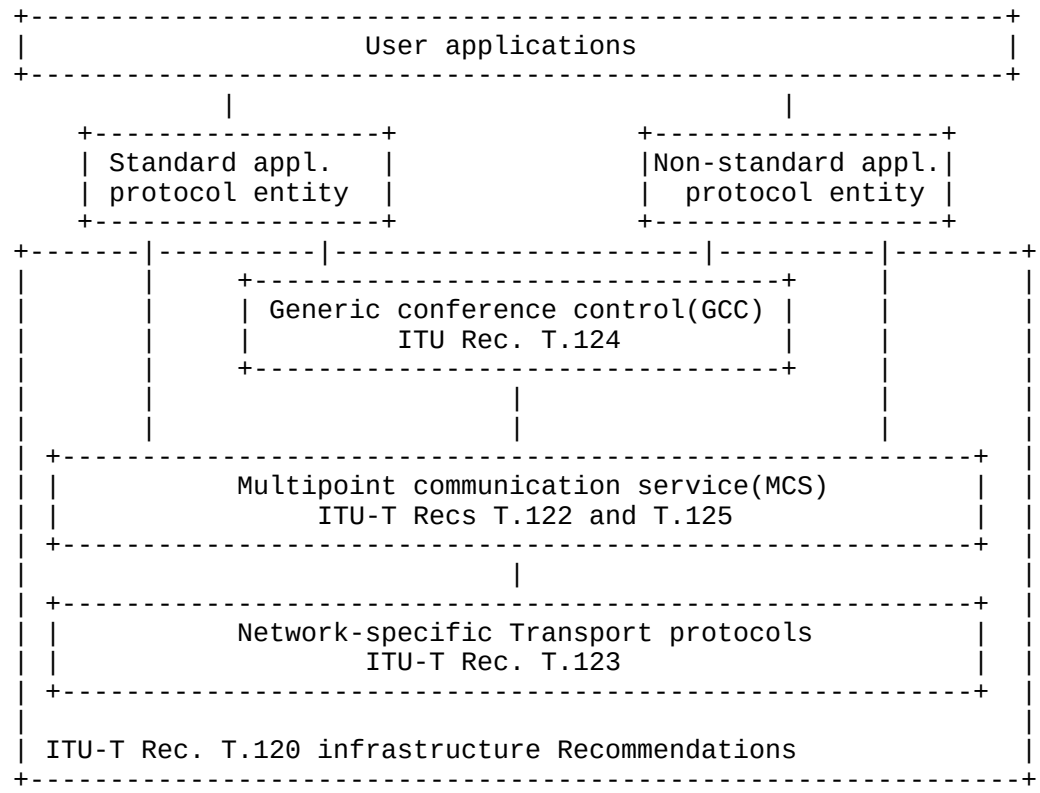

Figure 5 T.120 protocol

Generally, each layer provides services to the layer above and communicates to its peer(s) by sending Protocol Data Units (PDUs) via services provided by the layer below. The lower level layers (T.122, T.123, T.124, and T.125) specify an application-independent mechanism for providing multipoint data communication services to any application that can use these facilities. The upper level application layers define protocols, such as shared white boarding and multipoint file transfer. Applications using these standardized protocols can co-exist in the same conference with applications using proprietary protocols.

Following is one brief description of each protocol:

1. T.121: T.121 provides a template for T.120 resource management that developers should use as a guide for building application protocols. It also provides guidance to user application developers on how to utilize the T.120 infrastructure in a coherent and consistent way;

2. T.122:T.122 defines the multi-point services available to the developer. Together with T.125, they form MCS, the multi-point "engine" of the T.120 conference. MCS relies on T.123 to actually deliver the data. MCS is a powerful tool that can be used to solve virtually any multi-point application design requirement;

3. T.123: T.123 specifies transport profiles for different transport networks like, Public Switched Telephone Networks (PSTN), Integrated Switched Digital Networks (ISDN), TCP/IP and etc. From perspective of MCS layer, it presents a uniform OSI transport interface and services (X.214/X.224). Built-in error correction facilities are also included in this layer;

4. T.124:The T.124 specifies the Generic Conference Control (GCC), which provides a comprehensive set of facilities for establishing and managing the multi-point conference;

5. T.125: T.125 defines: 1) Procedures for a single protocol for the transfer of data and control information from one MCS provider to a peer MCS provider. 2) The structure and encoding of the MCS protocol data units used for the transfer of data and control information.

3.2. RDP

3.2.1. Introduction

Remote Desktop Protocol, which is issued by Microsoft, provides remote display and input capabilities over network connections for Windows-based applications running on a server. It is based on, and is an extension of, the T-120 family of protocol standards. So it inherits corresponding capabilities and benefits from T.120, such as the architectural features necessary to support multipoint and multipoint data delivery, which allows data from an application to be delivered in "real-time" to multiple parties without having to send the same data to each session individually.

3.2.2. RDP protocol stack

3.2.2.1. Basic architecture

As RDP is an extension of the core T.120 protocol, it re-uses services from T.123, T.122, T.125 and T.124 of T.120 protocol suite. In T.120 protocol architecture, RDP is at the same place of application protocols. It defines application PDU for information exchange between server and client, like capability set, bitmap update, keyboard/mouse event and so on.

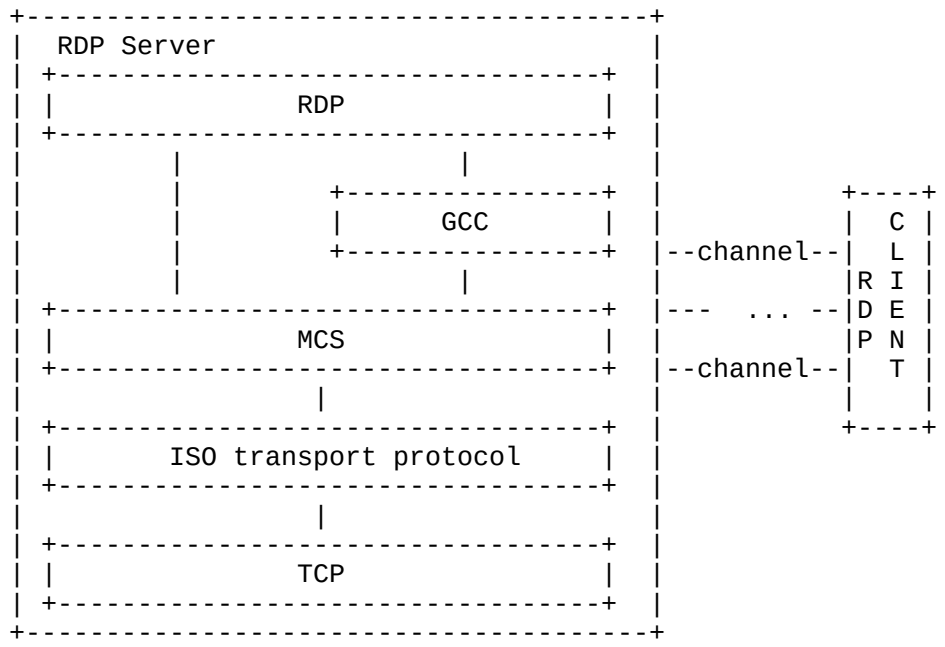

Figure 6 RDP protocol

RDP uses channel service provided by MCS layer for lossless communication between client and server components over the main RDP data connection. Each channel is targeted for one specific application data, such as printer redirection, clipboard redirection and etc.

There are two types of virtual channels in RDP: static virtual channel and dynamic virtual channel. A maximum of 31 static virtual channels can be created at connection time. The list of desired virtual channels is requested and confirmed during the Basic Settings Exchange phase of the server-client connection sequence and the endpoints are joined during the Channel Connection phase. Virtual channel data is application-specific and opaque to RDP. Each virtual channel acts as an independent data stream. The client and server examine the data received on each virtual channel and route the data stream to the appropriate endpoint for further processing. Dynamic virtual channel is put forward to solve the limitation of static virtual channel number. It is implemented on top of a static virtual channel named DRDYNVC. In the DRDYNVC static virtual channel, different services could be allocated and deleted dynamically.

RDP supports two security classifications: standard RDP security and enhanced RDP security:

1. For standard RDP security, client and server use matched encryption and decryption keys for data security. The encryption and decryption keys are generated during connection sequence, based on random number generated on server and client. The client and server both generate a 32-byte random value using a cryptographically-safe pseudorandom number generator. Then the server sends the random value which it generated (along with its public key embedded in a certificate) to the client in the Server Security Data during the Basic Settings Exchange phase of the connection sequence. After receiving that, the client sends its random value to the server (encrypted with the server's public key) in the Security Exchange PDU as part of the RDP Security Commencement phase of the connection sequence. As the client doesn't authenticate server credential, this security method could not protect data from man in-the-middle. It still exists in RDP for backward compatibility;

2. For enhanced security, RDP traffic is no longer protected by using the techniques described in standard RDP security. Instead, all security operations (such as encryption and decryption, data integrity checks, and server authentication) are implemented by external security protocols, like TLS. In this way, there is no longer need to manually implement protocol security mechanisms, which is replaced with well-known and proven security protocol packages to provide end-to-end security.

Server redirection procedure is defined in RDP for load-balancing scenarios. A client connection can be redirected to a specific session on another server by using the Server Redirection PDU. In the Server Redirection PDU, necessary information is include, such as session ID, redirection information flag, user credential information, load balance information, target server address and etc. When the client receives this redirection indication, it will disconnect from current server and try to connect target server, which is specified in Server Redirection PDU.

In RDP framework, it's flexible to define more service using separate static/dynamic channel.

3.2.2.2. Bandwidth saving

RDP uses a bulk compressor to compress virtual channel data and some data in PDUs sent from server to client. During information exchange phase of the connection sequence, server and client negotiate the bulk compressor. One version of the bulk compressor (the RDP 4.0 bulk compressor) is based on the Microsoft Point-To-Point Compression (MPPC) Protocol and uses an 8 kilobyte history buffer. A more advanced version of the compressor (the RDP 5.0 bulk compressor) is

derived from the RDP 4.0 bulk compressor, but uses a 64 kilobyte history buffer and modified Huffman-style encoding rules. Besides employing bulk compression for generic data, RDP also uses variations of run length encoding (RLE) rules to implement compression of bitmap data sent from server to client. "NSCodec Extension" and "RemoteFX Codec Extension" (lossy image codec) specify the image codec that can be used to encode screen images by utilizing efficient and effective compression.

As RDP is one extension of T.120 protocols, each RDP packet will be wrapped with corresponding headers defined in T.120 protocols, like TPKT header, X.224 Class 0 Data TPDU and etc. In some situations, for example point-to-point connection, these additional overhead is not necessary. In RDP protocol, it defined two choices: slow-path which follows strictly T.120 protocols and fast-path with the goal of improving bandwidth. For fast-path, the TPKT Header, X.224 Class 0 Data TPDU, and MCS Send Data Indication are replaced. The Security Header is collapsed into the fast-path output header, and the Share Data Header is replaced by a new fast-path format. The contents of the graphics and pointer updates are also changed to reduce their size, particularly by removing or reducing headers.

The real-time display interaction is accomplished by continuously sending updated bitmap images from server to client. Even though these bitmaps may be compressed, it is still not a bandwidth efficient mechanism to employ, especially when dealing with graphics-intensive applications that refresh regularly. "GDI Acceleration Extension" aims to reduce the bandwidth associated with graphics remoting by encoding the drawing operations that produce an image instead of encoding the actual image. For example, instead of sending the bitmap image of a filled rectangle from server to client, an order to render a rectangle at coordinate (X, Y) with a given width, height, and fill color is sent to the client. The client then executes the drawing order to produce the intended graphics result. In addition to defining how to encode common drawing operations, "GDI Acceleration Extension" also facilitates the use of caches to store drawing primitives such as bitmaps, color tables, and characters. The effective use of caching techniques helps to reduce wire traffic by ensuring that items used in multiple drawing operations are sent only once from server to client (retransmission of these items for use in conjunction with future drawing operations is not required after the item has been cached on the client).

For video and audio application, "Video Redirection Virtual Channel Extension" and "Audio Output Virtual Channel Extension" define video& audio delivery message and procedure between server and client. Before stream traffic starts, server and client will exchange their own capabilities, also for media format. After successful

negotiation, media stream data will be sent between server and client with both agreed format, for example MPEG-2, instead of transferring raw data.

3.2.2.3. User experience improvement

For VDI applications, user experience is very important for more widely acceptance of this technology. When using client component to access desktop on remote side, the user hopes to get experience of using local computer, such as real-time interactive, plug-n-play service, bi-directional multimedia services and etc.

In RDP protocol suite, it uses techniques mentioned in previous "bandwidth saving" section to improve interactive capability and multi-media services.

Following extension protocols are defined to mitigate environment experience of server and client:

1. Clipboard Virtual Channel Extension: specifies how to keep two distinct system clipboards in sync so that at any given time, the data available to an application on one computer (via its local clipboard) is identical to the data available to another application on a remote computer (via its local clipboard);

2. Remote Programs Virtual Channel Extension: provides support to present a remote application (running remotely on the server) as a local user application (running on the client machine). It extends the core RDP protocol to deliver this seamless windows experience;

3. File System Virtual Channel Extension: provides support to redirect access from the server to the client file system. In a typical terminal server scenario, many of the nonvolatile resources used by the server (such as hard drives, flash drives, and floppy disks) are located on the client. The server exposes a file system driver that is visible to server-based applications as a hard drive, which allows the applications to access the client file systems;

4. Serial Port Virtual Channel Extension: provides support to redirect serial and parallel ports from client to the server. This allows the server to access client ports as if the connected devices were local to the server;

5. Print Virtual Channel Extension: provides support to redirect printers from client to the server. This allows the server access to printers physically connected to the client as if the

devices were local to the server;

6. Smart Card Virtual Channel Extension: is designed to remotely execute requests on a client’s Smart Cards for Windows. When Smart Card Redirection is in effect, server application smart card subsystem calls are automatically remapped to the client"Cside Smart Cards for Windows, which will then receive the corresponding request;

7. Plug and Play Devices Virtual Channel Extension: This protocol is used to redirect Plug and Play (PNP) devices from client to the server, for example USB devices. This allows the server access to devices that are physically connected to the client as if the device were local to the server.

3.3. SPICE

3.3.1. Introduction

SPICE was originally developed by Qumranet, which was acquired by Red Hat in 2008. Later Redhat open sourced its SPICE hosted virtual desktop protocol. Now open-source SPICE project aims to provide a complete open source solution for interaction with virtualized desktop devices. The SPICE protocol, acting similar functions as Microsoft RDP, provides client and server communication to support virtual desktop infrastructure.

3.3.2. SPICE Protocol

3.3.2.1. Basic architecture

SPICE protocol defines a set of protocol messages for accessing, controlling, and receiving inputs from remote computing devices (e.g., keyboard, video, and mouse) across networks, and sending output to them. SPICE uses simple messaging and does not depend on any RPC standard or a specific transport layer.

From architecture perspective, SPICE is architected a bit different than RDP and ICA. While RDP and ICA are made of up two components (a remote software component that runs in the OS of the Windows host you’re connecting to, and a client), SPICE is actually made up of three components:

1. Remote guest component: A virtual graphics adapter running in the VM, just like RDP/ICA

2. Client component; The SPICE client software, just like RDP/ICA;

3. Remote host component: A virtual graphics device which the hypervisor makes available to the VM.

In other words, because SPICE has a hypervisor component, it will only work when your remote hosts are VMs.

Similar with RDP, SPICE protocol spilt the communication session into multiple communication channels. Currently the following communication channels are defined in the protocol: a) the main channel serves as the main spice session connection; b) display channel for receiving remote display updates; c) inputs channel for sending mouse and keyboard events; d) cursor channel for receiving pointer shape and position; e) Playback channel for receiving audio stream; f) Record channel for sending audio capture. More channel types will be added as the protocol evolves. Since RDP is based on T.120, it could use channel service from lower layer. But for SPICE, it has to do channel management along with application protocols, which may lead to some implementation complexity.

For user authentication in SPICE, the server generates a 1024 bit RSA key and sends the public part to the client (via RedLinkInfo). Client uses this key to encrypt the password and send it back to server (after RedLinkMess). Server decrypt the password, compare it to ticket.

SPICE doesn’t define encryption mechanism in application layer, which means that it depends on external security protocols for security guarantee so that having maximum flexibility in choosing an encryption method.

SPICE defines full server redirection primitives to support live migration. RDP supports server redirection for load balance purpose, which means server redirection only occurs during 1st server access. SPICE could support server redirection during run-time and corresponding migration process control.

3.3.2.2. Bandwidth saving

SPICE uses different compression strategy for different media resource. It uses QUIC/LZSS/GLZ for lossless image compression. QUIC (based on SFALIC) and GLZ (based on LZSS with a history-based global dictionary) are Redhat proprietary algorithms. As the video/ audio is played using different media player in the virtual machine, it’s not easy to send raw media data to client directly. In current SPICE architecture, the video/audio media file firstly is decoded in virtual machine, then captured by SPICE and sent to client using algorithm MJPEG for video and CELT for audio.

Similar with RDP, SPICE supports transmission 2D graphic primitives to client to fully re-use client computing resource and save bandwidth. In current SPICE implementation, SPICE server also does optimization for graphic commands by building up one graphic command tree and removing un-necessary commands (for example, it is hidden by other commands) to further saving bandwidth.

Caching for pixmaps, palettes and cursors are also defined in SPICE in order to avoid redundant transmissions to the client. In current implementation, the server is responsible for cache management, such as add or remove from the cache. The client cache size is set by the client and transferred to the server through the display channel initialization message. The server monitors the current cache capacity and when it lacks space it removes the least recently used cache items until there is enough available cache space. The server sends an invalidate command with these items and the client removes them. In rendering-related messages, corresponding cache id will be specified instead of carrying over actual image data.

3.3.2.3. User experience improvement

SPICE defines a framework to support interactive operation between client and server based on techniques described in pervious section. For service like device redirection, SPICE protocol currently has no definition. But SPICE community has plan and is going on for following:

1. Network tunneling: using virtual network interface to enable sharing of network resources. Currently the focus is on printer sharing but is not limited to that;

2. Off-screen surfaces: supports off-screen surfaces as infrastructure for future DirectDraw, video acceleration and 3D acceleration. GDI and X11 will also benefit from this feature. It will also lay foundation for multi-head support;

3. Clipboard sharing: allows clients to share their clipboards with SPICE servers;

4. USB sharing: allows clients to share their USB devices with SPICE servers.

3.4. ICA

3.4.1. Introduction

Independent Computing Architecture is a proprietary protocol for an application server system, designed by Citrix Systems. The protocol

lays down a specification for passing data between server and clients, but is not bound to any one platform. The ICA protocol is used within several of Citrix's products to provide desktop virtualization across many different platforms. The three specific Citrix products utilizing ICA are WinFrame, XenApp and XenDesktop. The main purpose of these products is to allow a centralized set of Microsoft Windows based servers to deliver Windows applications/ virtual machines to users on various platforms such as Linux, UNIX, MacOS and even Windows CE.

3.4.2. ICA Protocol

3.4.2.1. Basic architecture

From perspective of 7-Layer OSI model, ICA is one protocol on Presentation layer, which is defined for services like, data conversion, compression, decompression, encryption, decryption and etc.

Well known ports 1494/UDP/TCP and 2598/UDP/TCP are the two primary ports used for the ICA protocol. Over these ports, the ICA protocol handles the transmission of data between the client and the server. Data transfer from the client to the server generally includes mouse movement, events and other requests. Data transfers from the server to the client generally involve a high-level view of the display rather than a full image of the screen, dramatically saving bandwidth and reducing the dependency on latency.

ICA protocol also uses channel-based architecture. It is comprised of 32 virtual channels and each channel with a default priority. QoS is provided based on prioritization and less important data is sacrificed if necessary.

3.4.2.2. Bandwidth saving and User experience improvement

Since ICA is Citrix's proprietary protocol, more detailed information like how compression is defined is Citrix's secret.

From Citrix's product view, Citrix provides HDX technology to deliver a "high definition" desktop virtualization user experience to end users for any application, device or network. These user experience enhancements balance performance with low bandwidth - anything else becomes impractical to use and scale. HDX technology provides network and performance optimizations to deliver the best user experience over any network, including low bandwidth and high latency WAN connections.

4. Conclusion

Through the survey of several VDI products, we found the common key components of VDI architecture:

1. Desktop client: The software installed on client devices. It connects to virtual desktop over protocols such as RDP, ICA and SPICE;

2. Virtual desktop agent: The agent is installed on virtual machine for accepting the client's connection. Not all solutions have same architecture. For instance, in Redhat's VDI solution, SPICE server , which runs on host machine, plays this role and is shared by all virtual machines on the server;

3. Virtual machine pool: Virtual machine pool provides computing resource for virtual desktops. It launches new virtual machine when new client connects to it. Virtual machine pool could contain huge numbers of virtual desktops,so the pool should be highly scalable and reliable;

4. Desktop connection broker. The broker manages desktop connections. It chooses the best virtual machine for the user according to some policy (such as load balance) and directs the connection to virtual machine;

5. Access Gateway. Responsible for external client's access. It authenticates external users and provides secure connections between client and virtual desktop;

6. Authentication Server: Provides authentication for users. It can be directory server or others.

To optimize user experience, all solutions adopt a set of technologies, such as multimedia compress, image cache, device redirection, client rending and adaptive bandwidth management. These technologies ensure that virtual desktop users feel the desktop is running locally.

There are two modes about how the client connects to virtual desktop:

1.The client connects to agent which is running on virtual machine. In Citrix, Microsoft and VMware's solutions, there is a virtual desktop agent on each virtual machine. The agent opens protocol ports to wait for client's connection, and exchange information with the client. This modes is shown in Figure 7:

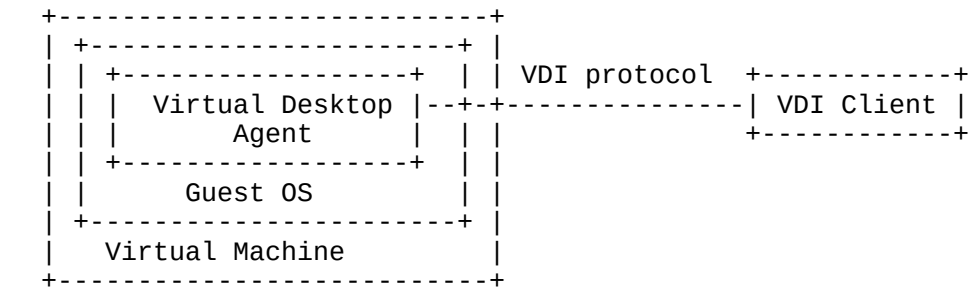

Figure 7 Agent on virtual machine

2.The client connects to agent which is running on host. In Redhat's solution, there is a SPICE server installed on host, instead of installed in every virtual machine. The SPICE server captures the virtual machine's screen information, and transfers it to the client. This mode is shown in Figure 8:

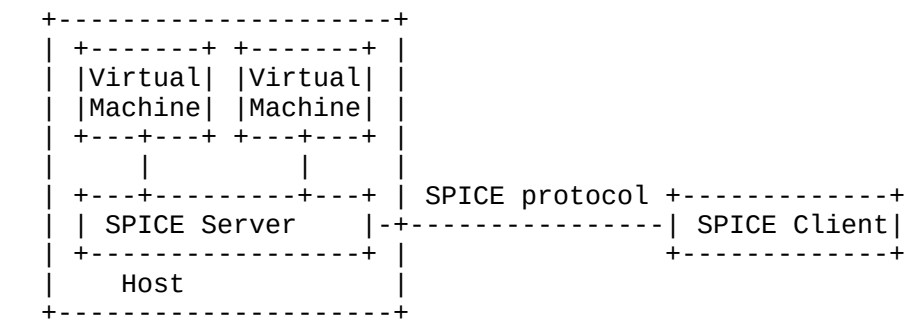

Figure 8 Agent on host

For different VDI protocols, they have similarities from perspective of functionality:

1. Support experience of operating local-computer when accessing remote server, through defining multi-channels and each channel is for one specific data type, like display/audio/input;

2. Enhance user experience through bandwidth saving techniques and more value-added services like Plug-n-Play: 1) define framework to support client and server negotiation of compression method and cache management; 2) define different protocol extensions to support different services, like device redirection; 3) define framework for graphic command transmission for 2D and 3D applications instead of large-size image.

Looking inside each protocol, they are different with each other, especially for:

1. Since they are proposed by different vendors and targeted for different purpose at initial time, detailed message and procedure are totally different, even for the architecture. For example, RDP is based on T.120, while SPICE is self-contained and ICA is proprietary;

2. Compression codec definition in each framework is incompatible. For example, NSCodec and RemoteFX codec are defined in RDP framework for image compression, while SPICE uses GLZ to compress the image;

3. Service definition completeness of each protocol is different. RDP protocol currently contains rich extension for some services, like file-system and device redirection, Plug-n-Play and etc. For SPICE, the community is working hard on that to catch it up.

5. Acknowledgements

6. Security Considerations

Related security issues will be addressed in subsequent draft.

7. References

7.1. Normative References

[ITUT120] ITU, "ITU T.120", 2009, <http://www.itu.int/rec/T-REC-T.120/en>.

[MCPP] Microsoft, "Windows Communication Protocols webpage", 2011, <http://msdn.microsoft.com/zh-cn/library/cc216513>.

[MSRDS] Microsoft, "Microsoft RDS webpage", 2011, <http://www.microsoft.com/windowsserver2008/en/us/rds-product-home.aspx>.

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

[SPICE] Redhat, Spice project., "Spice remote computing protocol definition v1.0", 2009.

[VMWARE] VMware, "VMware View webpage", 2011, <http://www.vmware.com/products/view/>.

[XENDESKTOP] Citrix, "Citrix XenDesktop webpage", 2011, <http://www.citrix.com/xendesktop>.

7.2. Informative References

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

Authors' Addresses

Suan Ma (editor)
ZTE Corporation
No.68 Zijinghua Road
Nanjing,
CN

Phone: +86 15950586953
Email: ma.suan@zte.com.cn

Liang Liang
ZTE Corporation
No.68 Zijinghua Road
Nanjing,
CN

Phone: +86 18913825360
Email: liang.liang12@zte.com.cn

Jun Wang
ZTE Corporation
No.68 Zijinghua Road
Nanjing,
CN

Phone: +86 13770604455
Email: wang.jun17@zte.com.cn

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 24, 2011

A. Melnikov
Isode Ltd

January 20, 2011

Simple Mail Transfer Protocol extension for Alternate Recipient Delivery Option
draft-melnikov-smtp-altrecip-on-error-00

Abstract

This document describes an SMTP extension for allowing the submitter to specify an alternate message recipient to be used in case where there is an error or delay delivering the message to a primary recipient.

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 July 24, 2011.

Copyright Notice

Table of Contents

1. Introduction

This document describes an SMTP extension for allowing the submitter to specify for each primary recipient an alternate message recipient to be used in case where there is an error or delay delivering the message to the corresponding primary recipient. The submitter can also optionally specify a controlling time. If the message is delivered normally to the primary recipient, the alternate recipient is never used. However if there is a problem delivering to the primary recipient, such as a permanent failure (5xx) reply-code from the next hop MTA/MDA, connection timeout to the next hop MTA/MDA, timer expiration or continuous transient failure (4xx) reply-codes, then the message is forwarded to the alternate recipient and the controlling time (if specified) is used as the new timer for the forwarded message.

The motivation behind this extension is to allow automatic handling of critical messages, as bouncing back to the submitter may add too much delay to processing of the message.

This extension might be useful for military and support type environments.

2. Conventions Used in This Document

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].

The formal syntax use the Augmented Backus-Naur Form (ABNF) [RFC5234] notation including the core rules defined in Appendix B of RFC 5234 [RFC5234].

3. Definition

The following service extension is defined:

1. The name of the SMTP service extension is "Alternate Recipient on delivery failure".

2. The EHLO keyword value associated with this extension is "ALTRECIP".

3. No parameter values are defined for this EHLO keyword value. In order to permit future (although unanticipated) extensions, the EHLO response MUST NOT contain any parameters for that keyword. Clients MUST ignore any parameters; that is, clients MUST behave as if the parameters do not appear. If a server includes

ALTRECIP in its EHLO response, it MUST be fully compliant with this version of this specification.

4. No additional SMTP verbs are defined by this extension.

5. One optional parameter ("ABY") is added to the MAIL FROM command. The ABY parameter specifies an alternate BY parameter [RFC2852] value that will be used if the message can't be delivered to one of the recipients (possibly within the time period specified in the BY parameter, if it is also present). The ABY parameter includes the by-time field (see Section 6) and some processing flags. The by-time field specifies the a decimal representation of the number of seconds within which the message should be delivered and has the range -999,999,999 seconds <= by-time <= +999,999,999 seconds.

6. One optional parameter ("ARCPT") is added to the RCPT TO command. The ARCPT parameter specifies an alternate email address to deliver this message to in case the corresponding original recipient is unreachable or unreachable within the specified period of time as specified in the MAIL FROM BY parameter.

7. The maximum length of a MAIL FROM command line is increased by up to 18 octets by the possible addition of the ABY keyword and value. The maximum length of a RCPT TO command line is increased by up to 501 octets by the possible addition of the ARCPT keyword and value.

8. Servers offering this extension MUST provide support for, and announce, the DSN [RFC3461] extension and the DELIVERBY [RFC2852] extension.

9. The ALTRECIP extension is valid for the submission service [RFC4409]. The ALTRECIP extension is not valid for LMTP [RFC2033].

4. SMTP protocol interactions

The following rules apply to SMTP transactions in which any of the ABY or ARCPT keywords are used:

1. If an SMTP client issues a MAIL command containing a valid ABY parameter and associated <esmtp-value> ([RFC5321] Section 4.1.2), a conforming SMTP server MUST return the same reply-code (and Enhanced Status Code) as it would to the same MAIL command without the ABY parameter. A conforming SMTP server MUST NOT refuse a MAIL command based on the absence or presence of a syntactically valid ABY, or on their associated <esmtp-values>.

However, if the associated <esmtp-value> is not valid (i.e., contains illegal characters), or if there is more than one ABY parameter in a particular MAIL command, the server MUST issue the reply-code 501 (with 5.5.2 Enhanced Status Code) with an appropriate message (e.g., "syntax error in parameter").

2. If an SMTP client issues a RCPT command containing any valid ARCPT parameter, a conforming SMTP server MUST return the same response (and Enhanced Status Code) as it would to the same RCPT command without those ARCPT parameter. A conforming SMTP server MUST NOT refuse a RCPT command based on the presence or absence of this parameter. However, if the associated <esmtp-values> is not valid, or if there is more than one ARCPT parameter in a particular RCPT command, the server SHOULD issue the response "501 syntax error in parameter" (with 5.5.2 Enhanced Status Code). The validity check MUST include verification for syntactic validity, and MAY include verification of the validity of the domain part of the address using DNS or verification of the validity of the whole address (e.g. using SMTP VRFY command, etc.).

The entire ARCPT parameter MAY be up to 500 characters in length.

5. Handling of messages received via SMTP

This section describes how a conforming MTA should handle any messages received via SMTP.

5.1. Receipt of a messages by a conforming SMTP servers

Messages received by an MTA/MSA compliant with this specification are processed as specified by [RFC5321]. Additionally, when inserting a Received header field as specified in Section 4.4 of [RFC5321], the compliant MTA/MSA SHOULD include the ALTRECIP clause which syntax is specified in Section 6. ([[anchor7: Open Issue: Should another Received clause be added that contains the ARCPT value? If yes, it would have implications similar to the currently defined FOR clause.]]) After that processing continues as specified is subsequent sections of this document.

5.2. Relay of messages to other conforming SMTP servers

The following rules govern the behavior of a conforming MTA, when relaying a message which was received via the SMTP protocol, to an SMTP server that supports the ALTRECIP SMTP extension. (Note that this section doesn’t apply to Mediators, which are covered in Section 5.4).

1. If the ABY parameter was supplied for a message when the message was received, the MAIL command issued when the message is relayed MUST also contain the ABY parameter along with its associated <esmtp-value>. If the ABY parameter was not supplied when the message was received, the ABY parameter MUST NOT be supplied for that message when the message is relayed.

2. If any ARCPT parameter was present in the RCPT command for a recipient when the message was received, an ARCPT parameter with the identical <alt-recip-address> MUST appear in the RCPT command issued for that recipient when relaying the message. (For example, the MTA acting as a relay therefore MUST NOT change the case of any alphabetic characters in an ARCPT parameter.) If no ARCPT parameter was present in the RCPT command when the message was received, an ARCPT parameter MUST NOT be added to the RCPT command when the message is relayed, unless the receiving MTA is a Mail Submission Agent (MSA).

3. The client MUST act as specified in Section 5.6, if the ARCPT parameter was supplied for a recipient and any of these conditions apply:

   * the SMTP server returns a permanent failure (5xx) reply-code in response to the RCPT command

   * the sending MTA is unable to deliver or relay the message to the recipients for an extended length of time (to be determined by the MTA, see [RFC5321]), e.g. due to continuous transient failures (4xx), or inability to contact the next hop MTA.

   * the sending MTA is unable to deliver or relay the message before deliver-by-time (the BY timer expires) and the by-mode of "R" was specified [RFC2852].

5.3. Relay of messages to non-conforming SMTP servers

The following rules govern the behavior of a conforming MTA (in the role of client), when relaying a message which was received via the SMTP protocol, to an SMTP server that does not support the ALTRECIP SMTP extension:

1. ABY or ARCPT parameters MUST NOT be issued when relaying the message.

2. Parameters to MAIL and RCPT commands specified in [RFC3461] and [RFC2852] cause actions as specified in the respective documents, with the following exception: if the the message is not delivered

or relayed before deliver-by-time (the BY timer expires) and a by-mode of "R" was specified, the client MUST act as specified Section 5.6.

3. If the ARCPT parameter was supplied for a recipient, and the SMTP server returns a permanent failure (5xx) reply-code in response to the RCPT command, the client MUST act as specified Section 5.6.

4. If the ARCPT parameter was supplied for a recipient, and the SMTP server returns a transient failure (4xx) reply-code in response to the RCPT command, and the sending MTA is unable to deliver or relay the message to the recipients for an extended length of time (to be determined by the MTA, see [RFC5321]), the client MUST act as specified Section 5.6.

5. If the ARCPT parameter was supplied for a recipient, and the SMTP server returns a success (2xx) reply-code in response to the RCPT command, the client MUST issue a "relayed" DSN for that recipient, as if the recipient included NOTIFY=SUCCESS value [RFC3461].

Note: relaying to non-conformant MTAs is on the "best effort" basis. While this is not ideal, one of the motivations behind this extension is to avoid the need for the submitter to take explicit action in case a delivery to a primary recipient fails. So this document is recommending an attempt to deliver the message to the primary recipient when ALTRECIP is not supported by the next hop, in favor of bouncing the message or forwarding it to the corresponding alternative recipient.

5.4. Local delivery of messages

Upon successful delivery of a message that was received via the SMTP protocol, to a local recipient's mailbox, a conforming MTA MUST ignore the ARCPT and the ABY parameters. This extension doesn't require the MTA to perform any additional actions.

"Delivery" means that the message has been handed to the recipient's mailbox. For messages which are transmitted to a mailbox for later retrieval via [RFC3501], [RFC1939] or a similar message access protocol, "delivery" occurs when the message is made available to the IMAP (POP, etc.) service, rather than when the message is retrieved by the recipient's user agent.

Similarly, for a recipient address which corresponds to a Mediator (as defined in [RFC5598]), such as a mailing list exploder, an alias or a ReSender, "delivery" occurs when the message is made available

to that mediator, even though the mediator might refuse to deliver that message to its recipient(s).

5.5. Gatewaying a message into a foreign environment

The following rules govern the behavior of a conforming MTA, when gatewaying a message that was received via the SMTP protocol, into a foreign (non-SMTP) environment:

1. If the destination environment is unable to provide an equivalent of ABY and ARCPT parameters, the conforming MTA SHOULD behave as if it is relaying to a non-conformant SMTP (Section 5.3). In particular note the requirement to generate a "relayed" DSN on successful gatewaying for a recipient.

2. If the destination environment is capable of providing an equivalent of ABY and ARCPT parameters, the conforming MTA SHOULD behave as if it is relaying to a conformant SMTP (Section 5.2). Note that gateways to such environments can map ABY/ARCPT parameters as required.

5.6. Delivery to an alternate recipient on delivery failure

When delivery to an original recipient fails (or doesn’t succeed within the specified time period as described by the BY parameter (with by-mode of "R") to the MAIL command), then a delivery is attempted to the alternate recipient specified in the ARCPT parameter.

Because the value of the BY parameter to the MAIL command is decreasing with each hop, it is not suitable for use in the new transaction to alternate recipients. The ABY ("Alternate BY") parameter is used to replace it in the new transaction. The new MAIL command includes all MAIL parameters originally specified (unless a future SMTP extension explicitly excludes one of the parameters), with the exception of ABY (if any) and BY (if any). The ABY value (if specified) becomes the new BY value.

The new RCPT command includes all RCPT parameters originally specified (unless a future SMTP extension explicitly excludes one of the parameters), with the exception of ARPCT and ORCPT (if any). The ARCPT value becomes the new recipient address used by the RCPT command.

The new SMTP transaction proceeds as specified in [RFC3461] and [RFC2852].

[[anchor9: Open Issue: Should an extra Received header field be added

that contains information about the error condition that caused redirect? How does this interact with loop detection (and does it matter)? Alternatively, a new header field can be defined for this purpose.]]

5.7. Interaction with non-conforming SMTP servers which accept a message and then generate a Non Delivery Report

[[anchor11: Open Issue: As some non-conforming SMTP relays are going to accept a message and generate an NDN, instead of rejecting the corresponding recipient over the protocol, the document should specify if some system on the way back to the submitter (e.g. MDA for the Return-path) needs to intercept NDNs that correspond to ALTRECIP SMTP transactions and convert them to new SMTP transactions. Maybe this is just another deployment consideration.]]

5.8. Deployment considerations

Organizations often authorize multiple servers to accept mail addressed to them. For example, the organization may itself operate more than one server, and may also or instead have an agreement with other organizations to accept mail as a backup. Authorized servers are generally listed in MX records as described in [RFC5321]. When more than one server accepts mail for the domain-part of a mailbox, it is strongly advised that either all or none of them support the ALTRECIP extension. Otherwise, unpredictable behavior and mail failures might occur.

6. Syntax

The following syntax specification uses the Augmented Backus-Naur Form (ABNF) as described in [RFC5234]. Terms not defined here are taken from [RFC2852], [RFC3461] and [RFC5321].

```
alt-rcpt-parameter = "ARCPT=" alt-recip-address

alt-recip-address  = addr-type ";" xtext
                     ; <addr-type> and <xtext>
                     ; are defined in RFC 3461.

aby-parameter      = "ABY=" by-value

by-value           = by-time ";" by-mode [by-trace]
                     ; <by-time>, <by-mode> and <by-trace>
                     ; are defined in RFC 2852.
                     ;
                     ; <by-time> is a decimal representation of
                     ; the number of seconds within which the message
```

```
                        ; should be delivered and has the range
                        ; -999,999,999 seconds <= by-time <= +999,999,999 second
s

   Opt-info           = [Via] [With] [ID] [For] [AltRecip]
                        [Additional-Registered-Clauses]
                        ; Updates the Opt-info defined in RFC 5321

   AltRecip  = CFWS "ALTRECIP" FWS AltRecip-Value
               ; Complies with the <Additional-Registered-Clauses>
               ; non-terminal syntax from RFC 5321.

   AltRecip-Value = "yes"


   addr-type = <defined in RFC 3461>

   xtext = <defined in RFC 3461>

   by-time = <defined in RFC 2852>

   by-mode = <defined in RFC 2852>

   by-trace = <defined in RFC 2852>

   Via = <defined in RFC 5321>

   With = <defined in RFC 5321>

   ID = <defined in RFC 5321>

   For = <defined in RFC 5321>

   Additional-Registered-Clauses = <defined in RFC 5321>

   CFWS = <defined in RFC 5321>
```

7. Example

An original SMTP transaction with 2 recipients:

```
     S: 220 example.net SMTP server here
     C: EHLO example.com
     S: 250-example.net
     S: 250-DSN
     S: 250-DELIVERBY
     S: 250-ALTRECIP
     C: MAIL FROM:<eljefe@example.com> BY=120;R ENVID=QQ314159 ABY=60;R
     S: 250 <eljefe@example.com> sender ok
     C: RCPT TO:<topbanana@example.net> ARCPT=rfc822;bottom-apple@loc2.example.o
rg
     S: 250 <topbanana@example.net> recipient ok
     C: RCPT TO:<Dana@Ivory.example.net> NOTIFY=SUCCESS,FAILURE
         ORCPT=rfc822;Dana@Ivory.example.net
     S: 250 <Dana@Ivory.example.net> recipient ok
     C: DATA
     S: 354 okay, send message
     C:  (message goes here)
     C: .
     S: 250 message accepted
     C: QUIT
     S: 221 goodbye
```

The receiving MTA then tries to deliver the message to the next hop. If delivery to the first recipient fails (e.g. due to timer expiration or receipt of a 5XX status code), the message will be forwarded to an alternate recipient for the first message. The new SMTP transaction would look like:

```
     S: 220 loc2.example.org SMTP server here
     C: EHLO example.net
     S: 250-loc2.example.org
     S: 250-DSN
     S: 250-DELIVERBY
     S: 250-ALTRECIP
     C: MAIL FROM:<eljefe@example.com> ENVID=QQ314159 BY=60;R
     S: 250 <eljefe@example.com> sender ok
     C: RCPT TO:<bottom-apple@loc2.example.org>
     S: 250 <bottom-apple@loc2.example.org> is welcomed here
     C: DATA
     S: 354 okay, send message
     C:  (message goes here)
     C: .
     S: 250 message accepted
     C: QUIT
     S: 221 goodbye
```

8. IANA Considerations

This specification requests IANA to add the following SMTP extension to the list of registered SMTP Extensions: ALTRECIP.

This specification also requests IANA to add the following new Received header field clause to help with tracing email messages delivered using the ALTRECIP SMTP extension:

Clause name: ALTRECIP
Description: Signals whether an ARCPT parameter is specified for any of the recipients of the message
Syntax of the value: See Section 6 of RFCXXXX
Reference: [[anchor15: RFCXXXX]]

9. Security Considerations

Use of this extension can advertise to an attacker criticality or urgency of a message. In addition to this, the use of ARCPT parameter to RCPT command can advertise relationship of a primary recipient to the corresponding alternate recipient. If such information is confidential, use of a SMTP extension that can provide connection confidentiality such as [RFC3207] is recommended.

Also see Security Considerations of [RFC3461] and [RFC2852] for information of security considerations related to SMTP DSN and SMTP DELIVERBY extensions respectively.

10. References

10.1. Normative References

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

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, October 2008.

[RFC3461] Moore, K., "Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs)", RFC 3461, January 2003.

[RFC2852] Newman, D., "Deliver By SMTP Service Extension", RFC 2852, June 2000.

[RFC4409] Gellens, R. and J. Klensin, "Message Submission for Mail", RFC 4409, April 2006.

[RFC3207] Hoffman, P., "SMTP Service Extension for Secure SMTP over Transport Layer Security", RFC 3207, February 2002.

10.2. Informative References

[RFC2033] Myers, J., "Local Mail Transfer Protocol", RFC 2033, October 1996.

[RFC1939] Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, May 1996.

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

[RFC5598] Crocker, D., "Internet Mail Architecture", RFC 5598, July 2009.

Appendix A. Acknowledgements

Many thanks for input provided by Steve Kille, John Klensin, Dave Crocker and David Wilson.

The author of this document copied lots of text from RFC 3461 and RFC 2852.

Author’s Address

Alexey Melnikov
Isode Ltd
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

EMail: Alexey.Melnikov@isode.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 22, 2011

M. Nottingham
February 18, 2011

The Network Authentication Required HTTP Status Code
draft-nottingham-http-portal-02

Abstract

"Captive portals" are a commonly-deployed means of obtaining access credentials and/or payment for a network. This memo introduces a new HTTP status code as a means of addressing issues found in these deployments.

This memo should be discussed on the ietf-http-wg@w3.org mailing list, although it is not a work item of the HTTPbis WG.

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 August 22, 2011.

Copyright Notice

Table of Contents

1. Introduction

It has become common for networks to require authentication, payment and/or acceptance of terms of service before granting access. Typically, this occurs when accessing "public" networks such as those in hotels, trains, conference centres and similar networks.

While there are several potential means of providing credentials to a network, these are not yet universally supported, and in some instances the network administrator requires that information (e.g., terms of service, login information) be displayed to end users.

In such cases, it has become widespread practice to use a "captive portal" that diverts HTTP requests to the administrator's web page. Once the user has satisfied requirements (e.g., for payment, acceptance of terms), the diversion is ended and "normal" access to the network is allowed.

Typically, this diversion is accomplished by one of several possible techniques;

- IP interception - all requests on port 80 are intercepted and send to the portal.
- HTTP redirects - all requests on port 80 are intercepted and an HTTP redirect to the portal's URL is returned.
- DNS interception - all DNS lookups return the portal's IP address.

In each case, the intent is that users connecting to the network will open a Web browser and see the portal.

However, because port 80 is used for non-browser traffic, a number of issues (see Appendix B) have been encountered.

This memo introduces a new HTTP status code, 428 Network Authentication Required, as a solution to some of these issues. Appendix A outlines how it might be used in typical deployments.

2. 428 Network Authentication Required

This status code indicates that the client should authenticate to gain network access before resubmitting the request.

The response body SHOULD indicate how to do this; e.g., with an HTML form for submitting credentials.

Responses with the 428 status code MUST NOT be stored by a cache.

3. Security Considerations

In common use, a response carrying the 428 status code will not come from the origin server indicated in the request's URL. This presents many security issues; e.g., an attacking intermediary may be inserting cookies into the original domain's name space, may be observing cookies or HTTP authentication credentials sent from the user agent, and so on.

However, these risks are not unique to the 428 status code; in other words, a captive portal that is not using this status code introduces the same issues.

4. IANA Considerations

The HTTP Status Codes Registry should be updated with the following entry:

- Code: 428
- Description: Network Authentication Required
- Specification: [this document]

Appendix A. Using the 428 Status Code

This appendix demonstrates a typical use of the 428 status code; it is not normative.

A network operator wishing to require some authentication, acceptance of terms or other user interaction before granting access usually does so by identify clients who have not done so ("unknown clients") using their MAC addresses.

Unknown clients then have all traffic blocked, except for that on TCP port 80, which is sent to a HTTP server (the "login server") dedicated to "logging in" unknown clients, and of course traffic to the login server itself.

For example, a user agent might connect to a network and make the following HTTP request on TCP port 80:

```
GET /index.htm HTTP/1.1
Host: www.example.com
User-Agent: ExampleAgent
```

Upon receiving such a request, the login server would generate a 428 response:

```
HTTP/1.1 428 Network Authentication Required
Refresh: 0; url=https://login.example.net/
Content-Type: text/html

<html>
   <head>
   </head>
   <body>
      <h1>You are being redirected to log into the network...</h1>
   </body>
</html>
```

Here, the 428 status code assures that non-browser clients will not interpret the response as being from the origin server, and the Refresh header redirects the user agent to the login server (an HTML META element can be used for this as well).

Note that the 428 response can itself contain the login interface, but it may not be desirable to do so, because browsers would show the login interface as being associated with the originally requested URL, which may cause confusion.

Appendix B. Issues Raised by Captive Portals

Since clients cannot differentiate between a portal's response and that of the HTTP server that they intended to communicate with, a number of issues arise.

One example is the "favicon.ico" <http://en.wikipedia.org/wiki/Favicon> commonly used by browsers to identify the site being accessed. If the favicon for a given site is fetched from a captive portal instead of the intended site (e.g., because the user is unauthenticated), it will often "stick" in the browser's cache (most implementations cache favicons aggressively) beyond the portal session, so that it seems as if the portal's favicon has "taken over" the legitimate site.

Another browser-based issue comes about when P3P <http://www.w3.org/TR/P3P/> is supported. Depending on how it is implemented, it's possible a browser might interpret a portal's response for the p3p.xml file as the server's, resulting in the privacy policy (or lack thereof) advertised by the portal being interpreted as applying to the intended site. Other Web-based protocols such as WebFinger <http://code.google.com/p/webfinger/wiki/WebFingerProtocol>, CORS <http://www.w3.org/TR/cors/> and OAuth <http://tools.ietf.org/html/draft-ietf-oauth-v2> may also be

vulnerable to such issues.

Although HTTP is most widely used with Web browsers, a growing number of non-browsing applications use it as a substrate protocol. For example, WebDAV <http://tools.ietf.org/html/rfc4918> and CalDAV <http://www.ietf.org/rfc/rfc4791.txt> both use HTTP as the basis (for network filesystem access and calendaring, respectively). Using these applications from behind a captive portal can result in spurious errors being presented to the user, and might result in content corruption, in extreme cases.

Similarly, other non-browser applications using HTTP can be affected as well; e.g., widgets <http://www.w3.org/TR/widgets/>, software updates, and other specialised software such as Twitter clients and the iTunes Music Store.

It should be noted that it's sometimes believed that using HTTP redirection to direct traffic to the portal addresses these issues. However, since many of these uses "follow" redirects, this is not a good solution.

Appendix C. Non-HTTP Applications and Techniques

This memo does not address non-HTTP applications, such as IMAP, POP, or even TLS-encapsulated HTTP. Since captive portals almost always target Web browsers (has anyone ever seen one that inserts an e-mail into your inbox asking you to authenticate?), this is appropriate.

Instead, it is anticipated that well-behaved portals will block all non-HTTP ports (especially port 443) until the user has successfully authenticated.

Overall, there may also be an interesting discussion to be had about improving network access methods to the point where a user interface can be presented for the same purposes, without resorting to intercepting HTTP traffic. However, since such a mechanism would by necessity require modifying the network stack and operating system of the client, this memo takes a more modest approach.

Appendix D. Acknowledgements

The author takes all responsibility for errors and omissions.

Author's Address

Mark Nottingham

Email: mnot@mnot.net
URI: http://www.mnot.net/

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 8, 2011

E. Rescorla
RTFM, Inc.
J. Hildebrand
Cisco Systems, Inc.

March 7, 2011

JavaScript Message Security Format
draft-rescorla-jsms-00.txt

Abstract

Many applications require the ability to send cryptographically secured messages. While the IETF has defined a number of formats for such messages (e.g. CMS) those formats use encodings which are not congenial for Web applications. This document describes a new cryptographic message format which is based on JavaScript Object Notation (JSON) and thus is easy for Web applications to generate and parse.

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 September 8, 2011.

Copyright Notice

Table of Contents

1. Introduction

Many applications require the ability to send cryptographically secured (encrypted, digitally signed, etc.) messages. While the IETF has defined a number of formats for such messages, those formats are widely viewed as being excessively complicated for the demands of Web applications, which typically only need the ability to secure simple messages. In addition, existing formats use encoding mechanisms (e.g., ASN.1 BER/DER) which are not congenial for Web applications. This presents an obstacle to the deployment of strong security by such applications.

This document describes a new cryptographic message format, JavaScript Message Security (JSMS) intended to meet the need of the Web environment. While JSMS is modeled on existing formats -- principally CMS [RFC5652] -- it uses JavaScript Object Notation (JSON) rather than ASN.1/BER/DER, making it far easier for Web applications to handle. In the interest of simplicity, JSMS also omits as many as possible of the CMS modes (multiple signatures, password-based encryption).

2. Conventions Used In This Document

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. Overview

The JSMS message format is simply a JSON [RFC4627] dictionary with an appropriate collection of fields. Each operating mode will have a separate set of fields, with a common field to distinguish between the modes.

3.1. Operational Modes

JSMS supports two operational modes:

Encrypted Data
: A block of data encrypted under a random message encryption key (MEK). The MEK is then separately encrypted for each recipient, either via symmetric or asymmetric encryption. The data is always integrity protected, either via a separate Message Authentication Code (MAC) or an Authenticated Encryption with Associated Data (AEAD) algorithm such as AES-GCM or AES-CCM.

Signed Data
: A block of data signed by a single signer using his asymmetric key and optionally carrying his certificate. Multiple signatures are not permitted in order to keep things simple.

Any other desired security functions are provided by composition of these modes. For instance, a signed and encrypted message is produced by first creating a Signed message and then encrypting that data. (See Section 4.6 for more on composition.

3.2. Conventions

In general, JSMS follows the following structural conventions:

Minimize implementation complexity
: Wherever possible, protocol choices have been made such that the time and effort required to implement the protocol in many different programming languages will be minimized. This means that optimizations for bandwidth, CPU, and memory utilization have been explicitly avoided.

Base64 as the only encoding
: Any data that does not have a straightforward string representation (binary values, large integers, etc.) is base64-encoded (see: [RFC4648]). In some cases, hexadecimal encodings might be more convenient, but consistency is even more important to reduce implementation complexity.

No canonicalization
: In many cryptographic message formats, canonical encodings are used to allow the same value to be computed at both sender and recipient (e.g., for digital signatures). This is inconvenient in JSON, which just views messages as a bundle of key/value pairs. Instead, whenever canonicalization would be required, the relevant data is serialized and base64-encoded for transport, allowing both sides to run computations over the same original set of octets.

In-memory processing
: We assume that the entire message can fit in main memory and make no effort to design a wire representation which can be handled in small chunks in a single pass. This means, for instance, that there is no need to have a message digest indicator at the beginning of the message and then the signature at the end, as is done in CMS. Fields are simply serialized in whatever order is most convenient for the JSON implementation. The examples in this document are generally shown in whatever order seems most readable and are not normative.

3.3. Certificate Processing

Experience has shown that certificate handling (path construction) is one of the trickier parts of building a cryptographic system. While JSMS supports PKIX certificates, its certificate processing is far simpler than that of CMS. When a JSMS agent provides its certificate, it must provide an ordered chain (as in TLS [RFC5246]) terminating in its own certificate, thus removing the need to construct certificate paths. The certificates MUST be ordered with the end-entity certificate first and each certificate that follows signing the certificate immediately preceding it. In addition, because many implementations will not want to do any ASN.1/BER processing at all, we will define a Web Service which applications can use for chain validation and translation to an easy-to-parse format. (See [TODO]).

3.4. Certificate Discovery

JSMS will often be used in an online messaging environment with users that have an address of the form user@domain, such as email, XMPP, or SIP. As such, protocols such as WebFinger [I-D.hammer-webfinger] or an end-to-end protocol can be used to retrieve appropriate certificates. Downstream uses of JSMS SHOULD define a discovery mechanism suitable for the intended use.

4. Message Format

All of the field definitions in this section make use of JSON Schema [I-D.zyp-json-schema]. For each of the fields that is designed to hold an enumerated value, a registry will be created allowing other values to be used in addition to the values enumerated in the schema.

4.1. Base64 Handling

As stated in section 3.1 of [RFC4648], Base64 does not require linefeeds after a specific number of characters. Since linefeeds are not valid characters in a JSON string, whenever a field is specified to be Base64-encoded in this document, it MUST NOT include any line breaks. Base64-encoded fields also MUST NOT include JSON-encoded linefeeds such as "\n". Any linebreaks in the middle of Base64-encoded sections of the examples are unintended side-effects of the production process.

Implementation Note: Much existing Base64-encoding code will generate linefeeds every 64 or 76 characters of output. Ensure that these linefeeds are removed before inserting the output into a JSON structure.

4.2. Content Object

JSMS operates by providing transformations on "Content" objects, which are just mime-typed JSON objects. These objects are then wrapped in a signed/encrypted wrapper with the following fields:

ContentType: A MIME [RFC2045] media type that MUST be included indicating the type of the "Data" field.
Type: The constant string "content", to facilitate easy determination that this is the target content. This is useful (for example) in certain operating conditions where you must continue to unwrap layers of signatures until you get to the content. This field MUST be included.
Data: The data value MUST be included as a text encoded as Base64 (See: [RFC4648]).
ID: An OPTIONAL universally unique ID that identifies this message, for use in detecting replay attacks.
Created: An OPTIONAL field describing the UTC date/time that the content was encoded into JSON, formatted according to the "date-time" production of [RFC3339].

Signing and encryption transform a "Content" object into "Signed" and "Encrypted" objects respectively. Verification and decryption transform "Signed" and "Encrypted" objects back into "Content" objects. For example:

```
{
    "ContentType":"text/plain; charset=UTF-8",
    "Type":"content",
    "Data":"SGVsbG8sIFdvcmxkCg==",
    "ID":"746a4c9f-8e84-4313-b669-81590ee2949e",
    "Created":"2011-03-07T16:17Z"
}
```

Figure 1: Content Example

4.3. Common Elements

A JSMS message is a JSON dictionary object containing a set of specific values.

The following fields MUST be present in all messages:

Version: The version number. For this specification this value MUST be set to the string "1.0". See Section 5 for details on version handling.

Type: The type of the message. MUST be either "signed" or "encrypted", to indicate a signed message (Section 4.4) or an encrypted message (Section 4.5) respectively.

4.4. Signed Data

A "signed" message contains a signed data block plus a digital signature over that data. To simplify implementation, only one signer is allowed. In addition to the required fields from Section 4.3, the fields in a signature message are:

SignedData: This field MUST consist of a Base64-encoded "Content" structure (see Section 4.2), which MUST have been encoded into octets as UTF-8 prior to Base64-encoding. The signature is computed over the UTF-8 octet stream before Base64-encoding to ensure that the sender and receiver have the exact same representation.

DigestAlgorithm: The message digest used to compute the signature. This field MUST be present for RSA-based signatures but MAY be omitted for future signatures which do not allow flexible digests. For now, this field MUST have the value "SHA-256", meaning the digest algorithm was SHA-256 [FIPS-180-3].

SignatureAlgorithm: The signature algorithm used to compute the signature. This field MUST be present. For now, this field MUST have the value "RSA-PKCS1-1.5", meaning the signature algorithm was RSASSA-PKCS1-v1_5 as specified in [RFC3447].

Signer: The signer’s identity, expressed as a URI [RFC3986]. This field MUST be present.

CertChain: The signer’s certificate chain, if any (see Section 4.4.2.1).

Signature: The Base64-encoded signature, which MUST be included (see Section 4.4.1).

```
{
    "SignedData":"ewogICAgIkNvbnRlbnRUeXBlIjoidGV4dC9wbGFpbjsgY2hhcn
                  NldD1VVEYtOCIsCiAgICAiVHlwZSI6ImNvbnRlbnQiLAogICAg
                  IkRhdGEiOiJTR1ZzYkc4c0lGZHZjbXhrQ2c9PSIsCiAgICAiSU
                  QiOiI3NDZhNGM5Zi04ZTg0LTQzMTMtYjY2OS04MTU5MGVlMjk0
                  OWUiLAogICAgIkNyZWF0ZWQiOiIyMDExLTAzLTA3VDE2OjE3Wi
                  IKfQ==",
    "DigestAlgorithm":"SHA-256",
    "SignatureAlgorithm":"RSA-PKCS1-1.5",
    "Signer":"xmpp:romeo@example.net",
    "Signature":"sNsxJltUaz4pSzAtJiPZagUMV4SwWugWexGbffK/WJRDi2uq7TxN
                 /V9SwG/kvQ7CaTABbeUuc6cKGO5YxnH5hME3bHB5L9PKPWSjxzxo
                 68RPxQyPli2YJDDHKVPbofEa86CLqYcwTF5qrcL7fQFvlRSOVxpS
                 SJfIdiAJNA+nEnk="
}
```

Figure 2: Signed Message Example

4.4.1. Signature Computation

The signature is computed over the string prior to base64 encoding. I.e., the processing order for encoding is:

1. Serialize the inner "Content" value into a UTF8-encoded octet series X.
2. Compute the signature value over X, and call the result Y. (In the case of signatures which use digests, this means feed the literal octets of the signature into the digest function.)
3. Compute the Base64 representation of X and insert it into the "SignedData" field of the message.
4. Compute the Base64 representation of Y, and insert the result into the "Signature" field.

This procedure removes dependencies on the exact serialization algorithm; variation in spacing, field order, etc. do not affect signature validity since the Base64 representation preserves them on the wire and protects them from modification by intermediaries.

Note: An alternative algorithm would be to compute the signature on the base64 representation itself, but this has two disadvantages: (1) any intermediaries which change spacing/line breaks would break the signature. (2) it is inconsistent with the algorithm for encryption (Section 4.5), which is designed to avoid multiple base64 encoding.

This procedure only specifies the input to the signature computation. The details of the computation depend on the signature algorithm itself. The mapping from code points to algorithms is found in

Section 6.

4.4.2. Signature Verification

In order to verify the signature, the steps of the previous section are reversed.

1. Process the provided "Signer" and "CertChain" fields as described in Section 4.4.2.1 in order to determine the sender's public key.
2. Base64 decode the "SignedData" field in order to recover a string X.
3. Verify the "Signature" field against X using the sender's public key and the "SignatureAlgorithm" and "DigestAlgorithm" fields. If the signature fails, return an error.
4. Deserialize X to recover the inner "Content" value.
5. Check any "ID" or "Created" fields for replay.
6. Using the value of the "ContentType" field to give MIME type context, Base64-decode the "Data" field to retrieve the intended message.

4.4.2.1. Certificate Processing

JSMS uses the "CertChain" element to carry certificate chains. For the moment, each certificate in the chain is expected to be a PKIX certificate BER-encoded then Base64-encoded. Future versions of this document will likely specify other valid certificate formats, since one of the goals of this format is to avoid . The meaning of the fields is described below:

Type: The type of the certificate chain. The only defined value is "PKIX", referring to PKIX [RFC5280] certificates.

Chain: An array of certificate values. In the case of "PKIX" certificates this is a list of base64-encoded DER/BER PKIX certificate values. PKIX certificates MUST be represented in order with each certificate certifying the next and the final certificate representing the end-entity.

```
{
    "Type":"PKIX",
    "Chain":[
        "MIICPjCCAaegAwIBAgIBETANBgkqhkiG9w0BAQUFADBDMRMwEQ
         YKCZImiZPyLGQBGRYDY29tMRcwFQYKCZImiZPyLGQBGRYHZXhh
         bXBsZTETMBEGA1UEAxMKRXhhbXBsZSBDQTAeFw0wNDA0MzAxND
         I1MzRaFw0wNTA0MzAxNDI1MzRaMEMxEzARBgoJkiaJk/IsZAEZ
         FgNjb20xFzAVBgoJkiaJk/IsZAEZFgdleGFtcGxlMRMwEQYDVQ
         QDEwpFeGFtcGxlIENBMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCB
         iQKBgQDC15dtKHCqW88jLoBwOe7bb9Ut1WpPejQt+SJyR3Ad74
         DpyjCMAMSabltFtG6l5myUDfqR6UD8JZ3Ht2gZVo8RcGrX8ckR
         Tzp+P5mNbnaldF9epFVT5cdoNlPHHTsSpoX+vW6hyt81UKwI17
         m0flz+4qMs0SOEqpjAm2YYmmhH6QIDAQABo0IwQDAdBgNVHQ4E
         FgQUCGivhTPIOUp6+IKTjnBqSiCELDIwDgYDVR0PAQH/BAQDAg
         EGMA8GA1UdEwEB/wQFMAMBAf8wDQYJKoZIhvcNAQEFBQADgYEA
         bPgCdKZh4mQEplQMbHITrTxH+/ZlE6mFkDPqdqMm2fzRDhVfKL
         fvk7888+I+fLlS/BZuKarh9Hpv1X/vs5XK82aIg06hNUWEy7yb
         uMitxV5G2QsOjYDhMyvcviuSfkpDqWrvimNhs25HOL7oDaNnXf
         P6kYE8krvFXyUl63zn2KE=",
        "MIICcTCCAdqgAwIBAgIBEjANBgkqhkiG9w0BAQUFADBDMRMwEQ
         YKCZImiZPyLGQBGRYDY29tMRcwFQYKCZImiZPyLGQBGRYHZXhh
         bXBsZTETMBEGA1UEAxMKRXhhbXBsZSBDQTAeFw0wNDA5MTUxMT
         Q4MjFaFw0wNTAzMTUxMTQ4MjFaMEMxEzARBgoJkiaJk/IsZAEZ
         FgNjb20xFzAVBgoJkiaJk/IsZAEZFgdleGFtcGxlMRMwEQYDVQ
         QDEwpFbmQgRW50aXR5MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCB
         iQKBgQDhauQDMJcCPPQQ87UeTX8Ue/b10HjppIrwo3Xs7bZWln
         +ImYWa8j5od4frntGfwLQX3KuJI6QdfhYjTE+oTfUxuHyq4xpJ
         CfRLJtsnZzCCEgFK6Rq2wQxTi2z8L3pD7DM2fjKye9WqzwEUxh
         LsE/ItFHqLIVgUE0xGo5ryFpX/IwIDAQABo3UwczAhBgNVHREE
         GjAYgRZlbmQuZW50aXR5QGV4YW1wbGUuY29tMB0GA1UdDgQWBB
         QXe5Iw/0TWZuGQECJsFk/AjkHdbTAfBgNVHSMEGDAWgBQIaK+F
         M8g5Snr4gpOOcGpKIIQsMjAOBgNVHQ8BAf8EBAMCBsAwDQYJKo
         ZIhvcNAQEFBQADgYEAACAoNFtoMgG7CjYOrXHFlRrhBM+urcdi
         FKQbNjHA4gw92R7AANwQoLqFb0HLYnq3TGOBJl7SgEVeM+dwRT
         s5OyZKnDvyJjZpCHm7+5ZDd0thi6GrkWTg8zdhPBqjpMmKsr9z
         1E3kWORi6rwgdJKGDs6EYHbpc7vHhdORRepiXc0="
    ]
}
```

Figure 3: PKIX CertChain Example

The recipient MUST verify the certificate chain (in the case of PKIX certificates according to [RFC5280]). If any validation failure occurs, the implementation MUST abort processing and return an error.

Once the certificate chain is validated, the end-entity certificate must contain an identity which matches the "Signer" field. In the case of PKIX certificates, the certificate MUST contain a

subjectAltName field of type "uniformResourceIdentifier". This field MUST be equivalent to the URI in the "Signer" field. If not, an error MUST be returned.

4.5. Encrypted Data

An "encrypted" message contains an encrypted "Content" block. All "encrypted" messages contain a symmetric integrity check, either via a MAC or via an AEAD [RFC5116] algorithm such as Galois/Counter Mode (GCM: [GCM]). A message may be encrypted to an arbitrary number of recipients. Each recipient is represented by a "Recipient" block, which contains a copy of the keying material encrypted for that recipient. Both symmetric and asymmetric key establishment is supported. In order to support both integrity and encryption, what is carried in the Recipient block is a Content Master Key (CMK) which is then used with a Key Derivation Function (KDF) to generate the Content Encryption Key (CEK) used to encrypt the message and the Content Integrity Key (CIK) used with the MAC. In addition to the required fields from Section 4.3 the fields in an encrypted message are:

Recipients: The list of recipients. This is an array of Recipient objects, each of which establishes the CMK for that recipient.

KDF: Specifies the key derivation function used to generate the CEK and the CIK from the CMK. This field MAY be absent if an AEAD algorithm is used, in which case the CEK is derived by copying the CMK.

Encryption: Specifies the properties of the encryption. The Algorithm field MUST contain the encryption algorithm and the IV field specifies the initialization vector (if required for the algorithm). This field MUST be present.

Integrity: Specifies the properties of the integrity check. The Algorithm field MUST contain the MAC algorithm and the Value field MUST contain the MAC. This field MAY be absent if no integrity check is used.

Data: Contains the ciphertext.

Each Recipient object provides an encrypted copy of the CMK for a single recipient. The meaning of the fields is described below:

KEKidentifier Describes the key encrypting key (KEK) used to encrypt the CMK. Either a "RecipientName" or a "KeyIdentifier" MUST be provided. If the "RecipientName" is provided, then a "CertificateDigest" SHOULD be provided.

RecipientName: Provides the recipient's name in URI form.
CertificateDigest: For now, the SHA-1 fingerprint of the PKIX certificate associated with the recipient.
KeyIdentifier The name of a shared symmetric key known to both sender and recipient. This need not be globally unique as long as it is unique within the recipient's context.

Algorithm: The algorithm used to encrypt the CMK. For now, one of "RSA-PKCS1-1.5" (meaning RSASSA-PKCS1-v1_5 as specified in [RFC3447]) or "AES-256-CBC" (meaning [FIPS-180-3]). Note the JSMS only supports key transport and not key agreement (since key agreement can always be turned into key transport).

Value: The CMK encrypted under the specified algorithm and key.

4.5.1. Message Encryption

The message encryption process is as follows.

1. Generate a random CMK. The CMK MUST have a length at least equal to that of the larger of the required integrity or encryption keys and MUST be generated randomly. See [RFC4086] for considerations on generating random values. [[TODO - we need a section on generating randomness in browsers - it's easy to screw up]]
2. Encrypt the CMK for each recipient (see Section 4.5.4)
3. Generate a random IV (if required for the algorithm).
4. Run the key derivation algorithm (see Section 4.5.3) to generate the CEK and CIK (if not using an AEAD algorithm).
5. Serialize the content into a bitstring M.
6. Encrypt M using the CEK and IV to form the bitstring C.
7. Set the Value element equal to the base64-encoded representation of C.
8. If not using an AEAD algorithm, compute the function I = MAC(CIK, C) using the chosen integrity algorithm. Note that this is EtA encryption which is considered the best cryptographic choice (See: [krawczyk-ate]). Set the Integrity.Value element equal to the base64-encoded representation of I.

4.5.2. Message Decryption

The message decryption process is the reverse of the encryption process.

1. Identify a Recipient block which appears to reference a key known to the recipient.
2. Decrypt the CMK. If this fails and another Recipient block appears plausible, that MAY be tried.

3. Run the key derivation algorithm (see Section 4.5.3) to generate the CEK and CIK (if not using an AEAD algorithm).
4. If not using an AEAD algorithm, compute the integrity check value I' on the binary representation of the Value element using the indicated integrity check. If the Integrity.Value does not match I', then an error MUST be reported and processing MUST be aborted.
5. Decrypt the binary representation of the Value element and output the result

4.5.3. Key Derivation

The key derivation process converts the CMK into a CEK. It assumes as a primitive a Key Derivation Function (KDF) which notionally takes three arguments:
MasterKey: The master key used to compute the individual use keys
Label: The use key label, used to differentiate individual use keys
Length: The length of the desired use key
The only real KDF specified in this document is the TLS PRF, which is invoked as PRF(MasterKey, Label) with an empty seed and produces an arbitrary length output. The appropriate number of bits (Length) is simply extracted from the beginning of the output. The KDF name "P_XXX" in this document refers the the TLS [RFC5246] PRF using P_XXX as the underlying P_hash function.

To compute the CEK from the CMK, the label "Encryption" is used.

To compute the CIK from the CMK, the label "Integrity" is used.

When AEAD algorithms are used the KDF element MUST NOT be present. When they are not used, it MUST be present.

4.5.4. CMK Encryption

JSMS supports two forms of CMK encryption:

- Asymmetric encryption under the recipient's public key.
- Symmetric encryption under a shared key.

4.5.4.1. Asymmetric Encryption

In the asymmetric encryption mode, the CMK is encrypted under the recipient's public key. The only currently defined asymmetric encryption mode is RSA-PKCS1-1.5, which refers to [RFC3447] RSAES-PKCS1-v1_5.

4.5.4.2. Symmetric Encryption

In the symmetric encryption mode, the CMK is encrypted under a symmetric key shared between the sender and receiver. All such modes MUST provide integrity for the CMK. This document defines four such modes: AES-128-CBC, AES-256-CBC referring to the [RFC5649] AES key wrapping modes and AES-128-GCM, AES-256-GCM, referring to AES encryption with GCM. For GCM the random 64-bit IV is prepended to the ciphertext.

4.6. Composition

This document does not specify a combination signed and encrypted mode. However, because the contents of a message can be arbitrary, and encryption and data origin authentication can be provided by recursively encapsulating multiple JSMS messages. In general, senders SHOULD sign the message and then encrypt the result (thus encrypting the signature). This prevents attacks in which the signature is stripped, leaving just an encrypted message, as well as providing privacy for the signer.

5. Version Processing

For the moment, all version numbers in the protocol MUST be 1.0. Receivers MUST return an error for any other version number. More interesting version processing will be defined in the future.

6. IANA Considerations

[TODO]
- Register MIME types
- Registries for signature, encryption, MAC
- Well known HTTP URLs

7. Security Considerations

Much more to follow here.

8. References

8.1. Normative References

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message

Bodies", RFC 2045, November 1996.

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

[RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, January 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC5649] Housley, R. and M. Dworkin, "Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm", RFC 5649, September 2009.

[I-D.zyp-json-schema]
Zyp, K. and G. Court, "A JSON Media Type for Describing the Structure and Meaning of JSON Documents", draft-zyp-json-schema-03 (work in progress), November 2010.

[FIPS-180-3]
National Institute of Standards and Technology (NIST), "Secure Hash Standard (SHS)", FIPS PUB 180-3, October 2008.

8.2. Informative References

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[I-D.hammer-webfinger] Hammer-Lahav, E., Fitzpatrick, B., and B. Cook, "The WebFinger Protocol", draft-hammer-webfinger-00 (work in progress), October 2009.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[krawczyk-ate] Krawczyk, H., "The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?)", Advances in cryptology--CRYPTO 2001 August 2001.

[GCM] National Institute of Standards and Technology (NIST), "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", SP 800-38D, November 2007.

Appendix A. JSON Schema

The following schemas formally define various namespaces used in this document, in conformance with [I-D.zyp-json-schema]. Because validation of JSON documents is optional, these schemas are not normative and are provided for descriptive purposes only.

A.1. Message Contents Schema

```
{
    "description":"Message Contents",
    "type":"object",
    "properties":{
        "ContentType":{
            "description":"A MIME content type",
            "type":"string",
            "required":true
        },
        "Type":{
            "description":"Dictionary type",
            "type":"string",
            "enum":["content"],
            "required":true
        },
        "Data":{
            "description":"The underlying data",
            "type":"string",
            "required":true
        },
        "ID":{
            "description":"(optional) unique ID for this message",
            "type":"string"
        },
        "Created":{
            "description":"(optional) time the message was created",
            "type":"string",
            "format":"date-time"
        }
    }
}
```

A.2. Common Elements Schema

```
{
    "description":"The basic schema for a JSMS message",
    "type":"object",
    "properties":{
        "Type":{
            "description":"Message type",
            "type":"string",
            "enum":["signed", "encrypted"]
        },
        "Version":{
            "description":"Version number for the message",
            "type":"string",
            "enum":["1.0"]
        }
    }
}
```

A.3. Signed Message Schema

```
{
    "description":"A signed message",
    "type":"object",
    "extends":message_schema,
    "properties":{
        "Signature":{
            "description":"The signature over the SignedData",
            "type":"object",
            "properties":{
                "SignedData":{
                    "description":"content to be signed, Base64",
                    "type":"string",
                    "required":true
                },
                "DigestAlgorithm":{
                    "description":"",
                    "type":"string",
                    "enum":["SHA-256"]
                },
                "SignatureAlgorithm":{
                    "description":"",
                    "type":"string",
                    "enum":["RSA-PKCS1-1.5"]
                },
                "Signer":{
                    "description":"",
                    "type":"string",
                    "format":"uri",
                    "required":true
                },
                "CertChain": {
                    "description":"the signer's cert chain",
                    "type":"PKIXcertchain"
                },
                "Signature":{
                    "description":"the signature",
                    "type":"string",
                    "required":true
                }
            }
        }
    }
}
```

A.4. PKIX Certificate Chain Schema

```
{
    "description":"A chain of PKIX certificates",
    "id":"PKIXcertchain",
    "properties":{
        "Type":{
            "description":"The type of certificate chain",
            "type":"string",
            "enum":["PKIX"] },
        "Chain":{
            "description":"PKIX certs ordered from root to end",
            "type":"array",
            "items":{
                "description":"A base64-encoded BER certificate",
                "type":"string"
            }
        }
    }
}
```

A.5. Encrypted Message Schema

```
{
    "description":"An encrypted object",
    "type":"object",
    "extends":message_schema,
    "properties":{
        "Recipients":{
            "description":"The list of recipient blocks",
            "type":"array",
            "required":true,
            "items":{
                "description":"A single recipient block",
                "type":"Recipient"
            }
        },
        "KDF":{
            "description":
            "The KDF used to derive the MAC and encryption keys",
            "type":"string",
            "enum":["P_SHA256"]
        },
        "Encryption":{
            "description":"Encryption control information",
            "type":"object",
            "required":true,
            "properties":{
```

```
                    "Algorithm":{
                        "description":"The algorithm used to encrypt",
                        "type":"string",
                        "enum":["AES-256-CBC"]
                    },
                    "IV":{
                        "description":"Initialization vector (base64)",
                        "type":"string"
                    }
                }
            },
            "Integrity":{
                "description":"The integrity control information",
                "type":"object",
                "properties":{
                    "Algorithm":{
                        "description":"The MAC algorithm",
                        "type":"string",
                        "enum":["HMAC-SHA-256"]
                    },
                    "Value":{
                        "description":"The MAC value (base64-encoded)",
                        "type":"string",
                        "required":true
                    }
                }
            },
            "Data":{
                "description":"The ciphertext (Base64-encoded)",
                "type":"string",
                "required":true
            }
        }
    }
```

A.6. Recipient Schema

```
{
    "description":"The recipient of an encrypted object",
    "type":"object",
    "id":"Recipient",
    "properties":{
        "KEKidentifier":{
            "type":"object",
            "description":"Identifies the key encrypting key",
            "properties":{
                "RecipientName":{
                    "type":"string",
                    "description":"The recipient's name",
                    "format":"uri"
                },
                "CertificateDigest":{
                    "type":"string",
                    "description":"Recipient's cert fingerprint"
                },
                "KeyIdentifier":{
                    "type":"string",
                    "description": "Shared symmetric key (opaque)"
                }
            }
        },
        "Algorithm":{
            "description":"The algorithm used to protect the CMK",
            "type":"string",
            "enum":["RSA-PKCS1-1.5", "AES-256-CBC"]
        },
        "Value":{
            "description": "Base64 of the encrypted CMK",
            "type":"string"
        }
    }
}
```

Appendix B. Acknowledgments

[TODO]

Authors' Addresses

Eric Rescorla
RTFM, Inc.
2064 Edgewood Drive
Palo Alto, CA 94303
USA

Email: ekr@rtfm.com

Joe Hildebrand
Cisco Systems, Inc.
1899 Wyknoop Street, Suite 600
Denver, CO 80202
USA

Email: jhildebr@cisco.com

Cloud Apps
Internet-Draft
Intended status: Informational
Expires: July 24, 2011

J. Wang, Ed.
S. Ma
L. Liang
ZTE Corporation
Jan 20, 2011

Virtual Desktop Infrastructure Problem Statement
draft-wang-clouds-vdi-problem-statement-00

Abstract

The Virtual Desktop Infrastructure is a technology to separate local desktop and remote computing and storage resources, which was initially derived from the remote desktop administration but with new business models and very different use cases. Most of existing VDI systems are based on proprietary implementation, and positioning different market with different features. Since virtual desktop technology is believed that it will become a mainstream application delivery method similar to the http protocol with regard to web applications, so it's important to make the virtual desktop access protocol open and standard. This draft summarizes the limitations of existing virtual desktop systems, and proposes the intent standardization work in IETF.

Table of Contents

1. Introduction

The Virtual Desktop technology is a combination of remote desktop control and virtualization. Client can access a remote virtual desktop or a remote application through VDI protocol. Currently, there are a lot of VDI system vendors, including Citrix, Microsoft, VMware etc., each of them provides bundle of components based on proprietary implementation, it's difficult to interwork between different vendors. In addition, the existing solutions focus on enterprise scale application, we haven't found any one designed for public virtual desktop service offering.

Since the VDI will be an alternative application delivery method and even more powerful than web, it can benefit a lot from a unified open and standard protocol, which ensures any qualified client can access any qualified service.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2. Terminology and Abbreviation

- OS: Operation System
- Guest OS: An OS runs on top of a virtual machine manager.
- VDI: Virtual Desktop Infrastructure.
- VM: Virtual Machine, contains at least one guest OS, usually several additional application processes.
- VMM: Virtual Machine Manager.
- VDA: Virtual Desktop Agent.

2. Virtual Desktop System Architecture

2.1. Common Framework

A VDI system includes at least one client and one server. VDI adopts the client-server model, in which the client runs a VDI viewer software, and connects to the server through virtual desktop access protocol. Unlike the remote administration, VDI client connects to a dedicated virtual machine and doesn't share the desktop with others.

The service provider or the enterprise's IT department must prepare unique VM image and virtual storage space for each client, and keep track of the modification on desktop and storage of its client.

Depend on the VDI system implementation, the VDI server module lies in either the VMM or the guest OS. Accordingly, the VDI systems can be divided into two catagories.

2.1.1. VDI server in the VMM

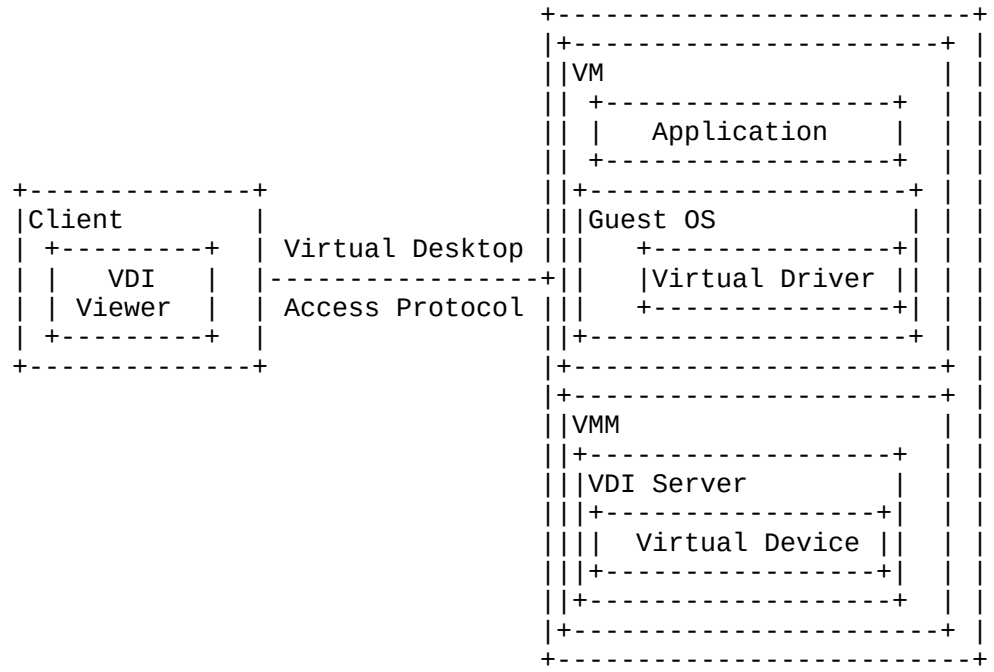

Figure 1 virtual desktop system framework: VDI server in the VMM

A VDI backend system makes up of a VDI server module running inside the VMM and some virtual devices drivers. The VDI server is the virtual desktop access protocol processing endpoint on the server side, and also simulates several necessary virtual devices such as graphic card, usb devices and sound card. The corresponding drivers are installed into the guest OS, so the I/O operation can be pass-through along the application - guest OS - virtual driver-virtual device-VDI server - virtual desktop access protocol - VDI client path.

The different I/O information encapsulates into respective transport

channel of the protocol, normally, including control, screen, mouse/ keyboard input, audio input, audio output and usb devices channel. Any screen change of the server will be sent back to the client which is in charge of re-rendering the information to the client screen; similarly, the mouse, keyboard, audio input will be sent from client to the server and audio output, usb writing operation at the server sent to the client. All of these VDI system behaviors make sure that the user gets the same experience as operating his local computers while accessing remote virtual desktop.

2.1.2. VDI Server In The Guest OS

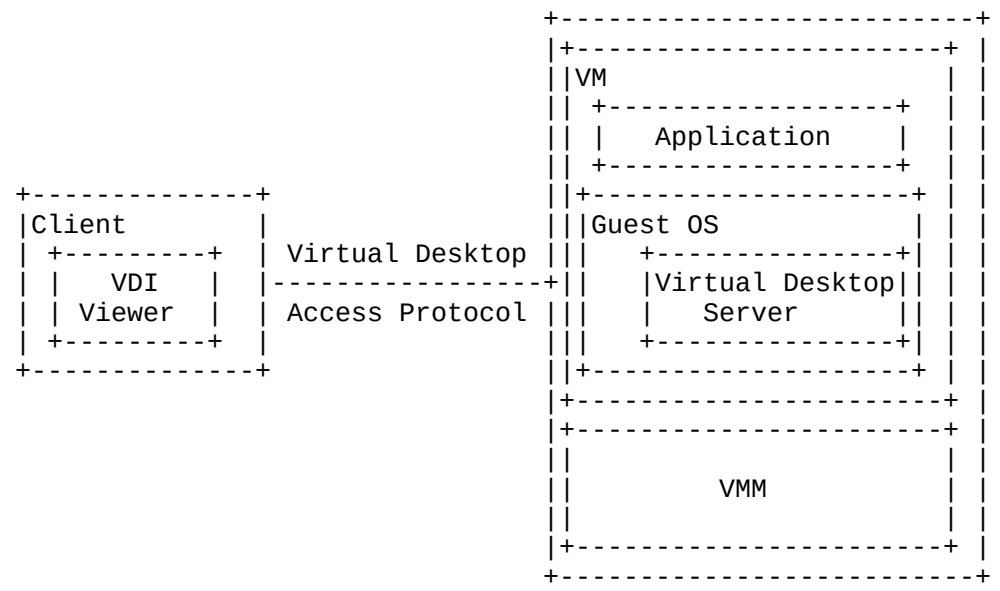

Figure 2 virtual desktop system framework- VDI server in the Guest OS

Compared to the first architecture, the only difference is virtual desktop server be installed in the guest OS. As a result, the connection terminates in the guest OS rather than in the VMM, so the client can’t communicate with its virtual desktop until the guest OS finished startup.

Sometimes, The virtual desktop is also called ’virtual desktop agent’, which performance the functions of both the virtual device and virtual driver modules, plus other necessary function.

2.2. Deployment System Architecture

A typical virtual desktop system comprises of clients, access gateways or connection brokers, authentication server, virtual desktop server pool and virtual application server, some additional component will also be deployed if extra features be required.

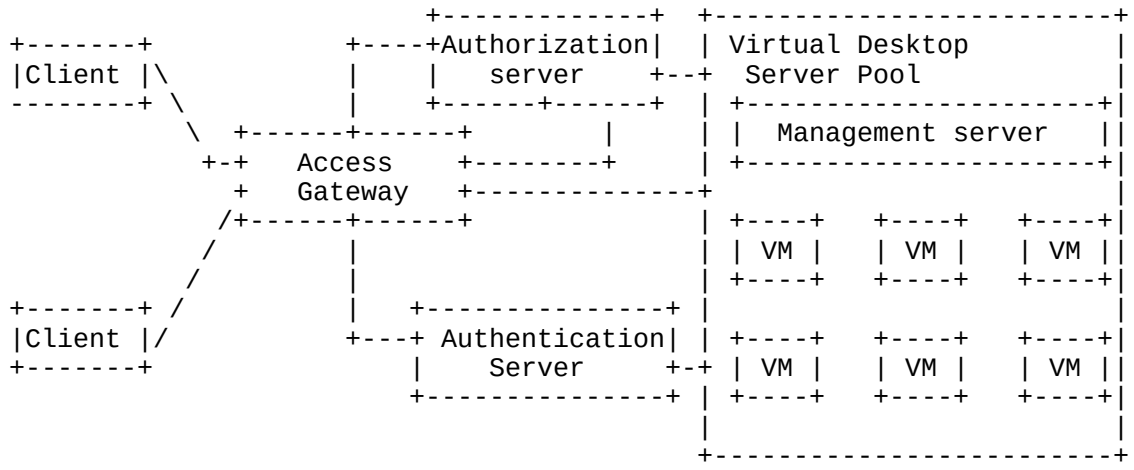

Figure 3 virtual desktop system deployment architecture

- VDI Clients, usually run a virtual desktop client software, sometimes they're called viewer or receiver. The client software performances service discovery, resource connection and virtual desktop access through virtual desktop access protocol.

- Access Gateway, which is a virtual desktop infrastructure entry point.If the client lies in the public network, the access gateway is used to authenticate the client and perform other security related function. In some specific VDI system, the access gateway is replaced with redirector, which directs the traffic to a virtual desktop resource instead of forcing all of the traffic going through a designated gateway.

- Authentication server, which stores required security material, responsible for the client authenticating processing. Most of the VDI systems reuse existing technology such the windows active directory.

- o Authorization server, it contains the VDI subscriber's profile information, including the virtual desktop cpu number, memory and disk quota, application preference, etc. Usually, the authorization server is merged into authentication server, e.g., the active directory or Home subscriber server defined in 3GPP. In certain case, the authorization server may also act as resources allocating server, accepting the authorization request from the user or access gateways and checking if the user has the rights to access certain services. If the result is positive, the server sends a virtual resource allocating request the resource pool management server for the user. After the virtual desktop resource successfully allocated, the user will be directed to connect the designated resource.

- o virtual desktop server pool, they're a bunch of computing and storage resources. Depending on VDI business model, the resources pool's scale may varies from hundreds to millions virtual machine.

Not every vendor provides aforementioned components. Some of them ,doesn't implement access gateway, e.g., Redhat provides a web portal to redirect the authenticated client to the designated VM. Most of the vendors support application virtualization and application streaming, in which the application software are installed on some dedicated servers instead of keeping a copy on every virtual desktop machine, reducing the disk space requirement significantly and making the virtual machine mobility simpler.

Since the VDI systems intend to realize a 'virtual real desktop' for the client, besides the screen, mouse and keyboard input/output information transfer, most of them support the I/O redirection features, which means the remote virtual desktop can access the client's I/O devices, including audio and usb equipment, just like the devices have been installed at the remote servers.

3. Application scenarios

3.1. scenario 1: Enterprise IT Application

The enterprise built a centralized virtual desktop resource pool, where each of the office staffs has been designated at least a VM. The employees run the office software and access the enterprise application through their virtual desktop. Any devices, e.g., thin clients, laptops, tablets and smartphones, can be used as virtual desktop clients without extra cross-platform software porting costs, and, the confidential data, documents stores in the server so that information leakage risk can be controled.

3.2. scenario 2: Hosted Virtual Computer

Virtual Computer service providers provide VM to their customers who access their VM via virtual desktop protocol. The subscription fees depend on the contracted computing resources, such as CPU number, memory capacity, disk quota, and their usage statistics.

The subscribers can access their virtual computer via any devices and any media, the benefit for them includes:1)Reduction of the devices hardware upgrading period, thus hardware cost can be cut down.2) Keeping data, documents synchronized among their devices. 3)Installing software once and using them anywhere and on any devices.

3.3. scenario 3: Hosted Telecommunication Terminals

The carriers supply the virtual desktop host service to their customers, which allows the clients to use the telecommunication application similar to local phones, such as voice/video calls, messages and address book, and furthermore, keep the same phone number as their real phones.

The benefits of this kind of business model are: 1)Multi-screen service convergence, which means the users can access the same application and content via different screens and get the same experience, and can migrate the services among different devices in realtime. 2)Get rid of the terminals' hardware limitations, one can even play a complex 3D game on a ordinary smartphone. Of cause, the smart phone only act as an I/O device while game is running on the remote VM. 3) Accelerating the time-to-market period of new telecom services. Formerly, if the new application impacts on the terminals' software, the operator must persuade the users upgrading their devices, this would spend very long time. But if the terminals were hosted in carrier's network, the upgrading can be done centralized in a short time.

3.4. scenario 4: Alternative SaaS delivery method

Till now, most of the SaaS delivered by WEB technology, but it's still a hard task to realize some sophisticated applications, such as graphical design and 3D game, in web pages. These cons of WEB are just the pros of virtual desktop. By using virtual desktop, SaaS can be extended to any software products.

4. Known Issues

At the very beginning, the virtual desktop solution targeted the

enterprise market, so the office software support is a priority. Therefore, most VDI product, especially the Microsoft's RDS/RDP product family, are highly optimized for the office software. Our test results show that RDP session only produces tens of kilobytes per second traffic while running Microsoft's word or excel software with regular document reading or editing operation, as well as the internet surfing.

Some of the VDI vendors claimed that their systems have been optimized for multimedia applications, however, the test results reveal that they are not fully optimized as their assertion especially under low-speed access environment. We have tested the streaming application performance at the platform based on Redhat's spice and Microsoft's RDP protocol. The spice protocol chooses M-JPEG as the streaming data compression algorithm, and the test results disclose that it has obvious better performance compared to Microsoft's RDP+terminal services combination. Despite the optimization efforts done by Redhat, the spice still consumes 12 times access bandwidth compared to the original bitrates of a standard definition video, that's totally not acceptable while accessing the services via carrier's wireless network.

M-JPEG does not use inter-frame encoding, which reduces the processing overhead but results in lower compression ratio. MPEG-4/ h.264's compression ratio is about 5-10 times compared better than M-JPEG, but requires more powerful CPU or GPU hardware. Recently, even the low end desktop or laptop systems have been equipped with integrated GPU capable of HD MPEG-4 video decoding. At the mobile sector, most of the 3G mobile phones have the h.264 decoding chipset installed, and the high-end handsets have shipped with an integrated GPU, e.g., iPhone 4/iPad.

Secondly, the audio streaming overhead also need to be cut down. The widely adopted audio codec in mobile network is AMR or it's wideband version - the AMR-WB, cost bandwidth about 6.6 to 23.85 kilobits per second, if encoded within rtp, the bitrate ranges between 23kbps and 40kbps with 20ms ptime parameter. In contrast, the regular audio streaming in the VDI system consumes 128kps or higher bandwidth. the excess overhead can not be neglected in the mobile network.

Thirdly, ssuming that the providers use VDI technology to deliver services, they purchase server hardware and sotware from different vendors for some commercial considerations, but still hope to provide the service to any potential customers whatever the device they used to access the network. So all of the participants would benefit from the fact that any VDI client device running any vendor's software can connect to the VDI backend services built by any vendor's solution, that is, a unified open and standard protocol should be widely

adopted. Furthermore, due to the VDI backend system variance, the access security mechanism may totally different, e.g., Active Directory vs. 3gpp sim based access security, so it'd better to separate the security framework from the basic VDI protocol design.

Fourthly, there're more and more mobile devices connecting to the network via wireless technology, the mobility management and service continuity should be a fundamental capability of the VDI protocol. Someone may argue that the mobility is the network's business, but actually, network based mobility can't provide inter-network ip mobility at most cases, we still need keep eye on the mobility issues while VDI client switches between a 3G network to another carrier's WiFi hotspot.

5. Intended Work In IETF

- Designing a fundamental virtual desktop access protocol, which enables the interworking between different VDI clients and backends. The protocol should support screen, mouse and keyboard mapping between clients and remote systems as well as audio and usb devices redirection.

- Defining a framework to accommodate advanced video and audio compression algorithm, especially allow the client negotiates intended algorithm with backend system.

- Defining an access security framework, in which the client and server can negotiate preferred security suite. To meet the minimal interworking requirement, some access security mechanism must be defined as a mandatory component, e.g., Pre-shared Key or PKI method, some other authentication mechanism also can be introduced, e.g., the 3gpp SIM based security. If the integrated security also need to be achieved, the protocol should support transfer credential information from clients to servers, e.g., restricted access to the client's certificates or SIM card from virtual desktop server side.

- Integrating the mobility and service continuity capabilities. To facilitate the user roaming among different network providers, mobility and service continuity should not rely on the underlying network facilities. Another service continuity related issue is caused by the VM live migration in the backend system, the protocol design should also take it into account.

6. Acknowledgements

7. Security Considerations

Related security issues will be addressed in subsequent draft.

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.

[RFC4867] Sjoberg, J., Westerlund, M., Lakaniemi, A., and Q. Xie, "RTP Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs", RFC 4867, April 2007.

[spice] redhat, spice project., "Spice remote computing protocol definition v1.0", 2009.

8.2. Informative References

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

Authors' Addresses

Jun Wang (editor)
ZTE Corporation
No.68 Zijinghua Road
Nanjing,
CN

Phone: +86 13770604455
Email: wang.jun17@zte.com.cn

Suan Ma
ZTE Corporation
No.68 Zijinghua Road
Nanjing,
CN

Phone:
Email: ma.suan@zte.com.cn

Liang Liang
ZTE Corporation
No.68 Zijinghua Road
Nanjing,
CN

Phone:
Email: liang.liang12@zte.com.cn