Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 6, 2020

R. Van Rein
InternetWide.org

February 3, 2020

HTTP Resources with User Names
draft-vanrein-http-unauth-user-05

Abstract

Many protocols support users under domain names, but HTTP does not.

This specification defines a header for user names, independent of authenticated identities, and a link to userinfo in HTTP URIs. This intergrates naturally with HTTP, and results in a more refined resource authentication model, in support of advanced usage scenarios.

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 https://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 6, 2020.

Table of Contents

1. Introduction 2
2. Definition of the HTTP User Header 3
3. URI with Resource User 3
4. Protocol Handling of URI and User Header 4
5. The Logic of User in HTTP 5
6. Environment Variable with Local User 6
7. Orthogonality of Authentication (Example) 6
8. IANA Considerations 8
9. Security Considerations 8
10. References 8
10.1. Normative References 8
10.2. Informative References 9
Appendix A. Compatibility with Basic Authentication 9
Appendix B. Compatibility with RESTful Patterns 10
Appendix C. Compatibility with Caching 10
Appendix D. Acknowledgements 11
Author's Address 11

1. Introduction

Many protocols support global identities like john@example.com to represent users like john under domains such as example.com. The URI format for HTTP can express [Section 2.7.1 of [RFC7230]] such authority components (with a userid named "resource user" herein), with an intended interpretation of locating user-specific resources. Many online applications publish resources on individual users, but there is no standard in HTTP to express user names to address them. This specification adds that through a header "User", closely paralleling the "Host" header.

Some current URIs for HTTP have used the userinfo field in the URI to express an authentication user (named "client identity" herein), in spite of the intended use to refine the authority information. This conflates the resource user with the client identity. This specification defines client identity and resource user as orthogonal concepts, and specifies a clear relation to the URI format.

Orthogonality yields a generalisation, but voluntary conflation of client identity and resource user remains possible. In fact, software may use it as default behaviour. Servers can be configured with resource users that demand authentication with the same client identity; they may even accept authentication with a client identity as a bypass to the same resource user. Clients may accept

authentication requests and use the resource user from the target URI as a hint to the expected client identity.

Orthogonal concepts can however be untied to support advanced use cases. Servers may use resource users to identify groups and welcome members to authenticate against such a group resource. While authenticating, client identities can be drawn from another domain, so it is possible to "bring your own identity" as long as the server can rely on a mechanism of realm crossover for credentials.

TOBEREMOVED: We have designed two mechanisms for realm crossover in other specifications; SXOVER is a SASL mechanism for realm crossover via a Diameter backend, which can be provisioned to user agents through HTTP SASL; KXOVER is a Kerberos mechanism that is taken care of in the KDC. Both rely on DNSSEC, DANE and TLS.

The purpose of this specification is to define clear meaning for http and https URIs and their userinfo mappings to HTTP.

2. Definition of the HTTP User Header

The "User" header carries a resource user as part of the requested authority, and therefore refines the resource name scope. The value can be explicitly inserted

or be

the user in the userinfo component of the target URI.

The User header value holds precisely one value with the following ABNF grammar:

User = *(unreserved / pct-encoded / sub-delims)

The referenced non-terminals are as for URIs [RFC3986] and can be directly included in the quoted-string header form; a plain token cannot express "(", ")", "=", ";" and "," without escaping [Section 3.2.6 of [RFC7230]]. Note that the header MUST NOT include a ":" colon (U+003a) character.

3. URI with Resource User

This section is informative.

Naming a user in the authority component of a URI is a general idea, already used for addressing users with SMTP, SIP, XMPP and many other protocols. The addition of users in the URIs for HTTP, a refined resource name is provided, and better crossover of identities with

these protocols can be achieved. Unlike server-specific user name mapping conventions, the specified generic meaning of URI userinfo as part of the authority information enables such crossovers to be automated.

There is a current practice of writing a client identity in the userinfo portion of a URI. This is considered useful if it adds Basic authentication to the first request; Basic can do this because it does not incorporate a server-sent challenge.

Having never been standardised, the mechanistic side of this practice is highly diverse, and URIs are far from portable between browsers or even the various places where they occur within one browser. As a result, these URIs cannot be distributed freely and their usage pattern is dedicated to the client software in use. This specification does completely support [Appendix A] this pattern as a special case.

This specification follows the URI's intention of the userinfo field, and prescribes copying its value into the User header. It will however remove anything from a colon onward, to suppress the portion of userinfo that should not be rendered [Section 3.2.1 of [RFC3986]] as well as a colon hinting at an empty password.

4. Protocol Handling of URI and User Header

Compliant user agents MUST pass userinfo from the target URI (up to but excluding the first colon ":" (U+003a) if it contains any) as a User header field if, and only if, the target URI contains a userinfo part. They MUST NOT remove userinfo from the target URI during this process. Empty userinfo MUST be treated as any other userinfo string.

The User header MAY appear in requests and MUST NOT occur in responses.

When sending it, the user agent SHOULD generate User as the next header field after Host. Transparent intermediates such as proxies and caches MUST NOT add, remove or modify the User header. The CONNECT method and Host header both exclude this information, so the User header complements them.

Compliant servers MAY ignore the User header [Section 3.2.1 of [RFC7230]] and they MAY impose a more restrictive grammar (like a NAI's utf8-username [RFC7542]) than the URI syntax before further processing it. When they do use it, the Effective Request URI [Section 5.5 of [RFC7230]] MUST be constructed with the userinfo and the "@" at delimiter (U+0040) prefixed to the host name and optional

port. Realms are specific to an authority section [Section 2.2 of [RFC7235]] and so a realm never spans across different userinfo values.

As a result of a consistent translation of any User header value into the Effective Request URI, the server would map consistently to resources. It is merely enabled to include a User header as an extra input variable to this mapping to resources.

HTTP caches [RFC7234] need to differentiate requests based on the User header. To accommodate that, the Vary header [Section 7.1.4 of [RFC7231]] MUST be generated by the server in the matching response, and the header MUST either be a single "*" star (U+002a) or list the "User" name, for all responses whose processing was influenced by the User header. This requirement has no bearing on software and configurations that ignore the User header.

Compliant user agents MUST NOT change the support of the User header depending on the source of a reference; be it a redirect form a server, a click in a hyperlinked document, a script or a part of a browser interface or an external source. When processing URIs that are relative to the context of a previous URI, compliant user agents MUST replace the userinfo in the target URI when the new URI specifies an authority component, and MUST keep it otherwise.

5. The Logic of User in HTTP

This section is informative.

HTTP structures a number of things around the authority component of its URIs, and the addition of resource users in this position form a logical extension. This leads to improved user experiences.

Realms are identified by a scheme, the authority and a descriptive string passed with authentication challenges. Clients can use this combination to decide about a client identity to present to a server; it is common for people to have roles relative to one another, and the standard definition of realm identity allows the user agent to select an identity to match the role for the remote party. This can even be used for credentials passed in the TLS handshake, such as X.509 certificates.

A similar logic is found for robot exclusion files. They are found at a path /robots.txt for a given scheme and authority. The inclusion of the resource user in the authority enables personal pages to each have their own robot exclusion file.

More futuristic would be a suggestion that a server may relay connections to user-operated web servers on the basis of the resource user; this is once again an intended use of the authority field. It may not be possible under current specifications yet, but HTTP with User header can support it as soon as TLS can.

6. Environment Variable with Local User

The following variable SHOULD be passed to applications that run on top of the HTTP stack in a server:

LOCAL_USER gives the HTTP User header value after syntax checking and percent-decoding. If used at all, it MUST be treated as a resource user. This header does not describe the authenticated client identity, which is usually passed in a variable REMOTE_USER.

7. Orthogonality of Authentication (Example)

This section is informative.

This section provides an example of an advanced use case. Not only does this use the resource user to locate a shared server account, it is also distinct from the client identity used during authentication. Whether the client identity is welcomed by the resource user is determined with an access control list. Furthermore, this example shows the logic of a realm identity that involves the resource user in finding the right client identity to the contacted resource user.

John and Mary are both part of the Sales group of Example, Inc and John has written a document that he wants Mary to review. Mary opens a link to the document name space under the group's shared account "sales" at https://sales@example.com/docs and her user agent sends:

```
GET /docs HTTP/1.1
Host: example.com
User: sales
```

The server redirects to add a slash, and when this is specific to the sales name space, it must inform caches about this with the Vary header:

```
HTTP/1.1 301 Moved Permanently
Location: /docs/
Vary: User
```

Since the new location lacks an authority component, this part is retained from the referring URI, and the user agent redirects to https://sales@example.com/docs/ by sending:

```
GET /docs/ HTTP/1.1
Host: example.com
User: sales
```

By this time, the server runs into access control, and decides that it needs an authenticated client identity. To this end, it responds with a challenge to the "Documents" realm:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: KnockKnock realm="Documents"
Vary: User
```

Mary's user agent needs to collect credentials, and may hint at the user name "sales" from the URI but, this being the name of a shared resource, Mary has no fitting credentials and instead authenticates with client identity "mary":

```
GET /docs/ HTTP/1.1
Host: example.com
User: sales
Authorization: KnockKnock realm="Documents", user="mary", ...
```

At some point, the server accepts Mary's authentication and proceeds to access control. This phase checks if client identity "mary" may access realm "Documents" of "https://sales@example.com" by checking that Mary works for the Sales department. Once this is assured, the server returns the requested document list:

```
HTTP/1.1 200 OK
Vary: User
Content-Type: text/html

...
<a href="/docs/review.cgi?docid=123">Review 123 now</a>
...
```

Mary clicks on the link to /docs/review.cgi?docid=123 and her user agent sees a relative reference with no authority component, so this is again reused from the referring URI. The new URI is https://sales@example.com/docs/review.cgi?docid=123 with same root https://sales@example.com for which Mary has an authenticated client identity, so the same "Documents" realm can be tried. The user agent therefore sends:

```
GET /docs/review.cgi?docid=123 HTTP/1.1
Host: example.com
User: sales
Authorization: KnockKnock realm="Documents", user="mary", ...
```

After access control, the server starts the CGI script with environment variables LOCAL_USER=sales and REMOTE_USER=mary for the resource user and authenticated client identity, respectively. The script interprets the LOCAL_USER as a group account and the REMOTE_USER as the acting group member, and returns a page for review of the document and Mary can get to work.

8. IANA Considerations

Please add the following entry to the Message Headers registry:

Header Field Name	Template	Protocol	Status	Reference
User		http	TBD	TBD:THIS_SPEC

9. Security Considerations

The User header field as defined herein is orthogonal to issues of authentication and authorisation, and adds no security concerns.

10. References

10.1. Normative References

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <https://www.rfc-editor.org/info/rfc7234>.

[RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <https://www.rfc-editor.org/info/rfc7235>.

10.2. Informative References

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <https://www.rfc-editor.org/info/rfc7542>.

Appendix A. Compatibility with Basic Authentication

This appendix is informative.

Basic authentication is regularly used as a quick and easy HTTP authentication technique. Several user agents continue to support it with the "user:password@" URI prefix to a hostname, despite its deprecation [Section 3.2.1 of [RFC3986]]. This specification imposes no new constraints on this practice; it merely prescribes sending the User header field, and leaves it to client software whether to also sends Basic authentication.

The mapping from HTTP requests to resources is the prerogative of the server. A server supportive of resource selection through Basic authentication could ignore the User header field and still comply with this specification. A server that does recognise the User header field would use it to locate a resource, before deciding about access control to that resource; it may subsequently require authentication, and select schemes that could be supported. At this time, it may or may not welcome an added Basic authentication attempt. All this depends on server configuration.

This flexibility can support a transition from Basic authentication to User headers on the server, and allows client software to also migrate by first adding the User header, and later supporting the advanced uses by allowing differentiation between resource user and client identity. Server administrators have a free choice whether to gradually phase out older clients or to continue to support them.

Sending both the User header and Basic authentication is only to be expected from user agents who conflate resource user with client identity. Such user agents will be less flexible, and will not be able to support more advanced usage patterns that separate these concepts, such as shared/group resources addressed with the User header field and, when desired, authentication through a set of other headers.

Appendix B. Compatibility with RESTful Patterns

This appendix is informative.

Whether and how the User header is interpreted is the prerogative of the server. The server will translate resources in the same manner when provided with the same User header, and may do so without regard for the HTTP method. The main concern is now if it will be addressed in the same manner in every case. This depends on the user agents.

Development environments make sending the User header field simple, so application support is as easy as the applications are flexible. Binary user agents and ones that may lag behind in updates do however call for backward compatibility support of consistent translation to resources.

Backward compatibility can be guaranteed for host names that always require a User header; all resources would be described with URIs having a (possibly empty) userinfo field. Failure to send a User header to such resources when the URI holds userinfo indicates that the user agent fails to support the User header. When an offer for Basic authentication is presented, it may be interpreted as the conflated approach to userinfo, and treated as a substitute for the User header. If neither is offered, then an error may be reported or control redirected to another means of selecting a resource user, perhaps through an alternate local naming convention.

This indicates that the server is able to detect inconsistent tranlation risks, and avoid accidentally binding a request to an unintended resource as a result of a missing User header.

Appendix C. Compatibility with Caching

This appendix is informative.

Whether and how the User header is used to find resources is the prerogative of their server. A conservative cache design might insert the User header value in request URIs, but lose the capability of seeing the equivalence of a resource as perceived by the server. The inclusion of the name "User" in the Vary header of the response adds explicit non-equivalence information, and thereby provides a more accurate cache controlling instruction.

Whether a result is "private" is independent of the User header, as that only signifies a refinement of the resource name space on the server. The rules that signify authentication as default indicator of privacy is orthogonal to the User header. Independent inclusion

of Basic authentication may still invalidate caching, but not as a result of this specification.

User agents that send Basic authentication will invalidate intermediate caching. When an empty password is used to select a resource user, it would improve caching performance to switch from Basic authentication to the User header.

Appendix D. Acknowledgements

This specification could be improved thanks to input from Daniel Stenberg, Amos Jeffries, Paul Vixie, James Fuller and Henri Manson.

Author's Address

Rick van Rein
InternetWide.org
Haarlebrink 5
Enschede, Overijssel 7544 WP
The Netherlands

Email: rick@openfortress.nl