Network Working Group
Request for Comments: 2243
Category: Standards Track

C. Metz
The Inner Net

November 1997

OTP Extended Responses

Status of this Memo

Copyright Notice

Abstract

This document provides a specification for a type of response to an OTP [RFC 1938] challenge that carries explicit indication of the response's encoding. Codings for the two mandatory OTP data formats using this new type of response are presented.

This document also provides a specification for a response that allows an OTP generator to request that a server re-initialize a sequence and change parameters such as the secret pass phrase.

1. Conventions, Terms, and Notation

This document specifies the data formats and software behaviors needed to use OTP extended responses. The data formats are described three ways: using an ad-hoc UNIX manual page style syntax, using augmented BNF described in sections two and three of RFC 822, and by examples. Should there be any conflict between these descriptions, the augmented BNF takes precedence. The software behaviors are described in words, and specific behavior compliance requirements are itemized using the requirements terminology (specifically, the words MUST, SHOULD, and MAY) defined in RFC 2119.

2. Extended Challenges and Extended Responses

This document builds on the protocol and terminology specified in RFC 1938 and assumes that you have already read this document and understand its contents.

An extended challenge is a single line of printable text terminated by either a new line sequence appropriate for the context of its use (e.g., ASCII CR followed by ASCII LF) or a whitespace character. It contains a standard OTP challenge, a whitespace character, and a list that generators use to determine which extended responses are supported by a server.

An extended response is a single line of printable text terminated by a new line sequence appropriate for the context of its use. It contains two or more tokens that are separated with a single colon (':') character. The first token contains a type specifier that indicates the format of the rest of the response. The tokens that follow are argument data for the OTP extended response. At least one token of data MUST be present.

2.1. Syntax

In UNIX manual page like syntax, the general form of an extended challenge could be described as:

```
<standard OTP challenge> ext[,<extension set id>[, ...]]
```

And the general form of an extended response could be described as:

```
<type-specifier>:<arg1>[:<arg2>[:...]]
```

In augmented BNF syntax, the syntax of the general form of an extended challenge and an extended response is:

```
extended-challenge = otp-challenge 1*LWSP-char capability-list
                     (NL / *LWSP-char)
otp-challenge     = <a standard OTP challenge>
capability-list   = "ext" *("," extension-set-id)
extension-set-id  = *<any CHAR except LWSP, CTLs, or ",">
extended-response = type 1*(":" argument) NL
type              = token
argument          = token
token             = 1*<any CHAR except ":" and CTLs>
NL                = <new line sequence appropriate for the context
                     in which OTP is being used>
```

An example of an extended challenge indicating support for OTP extended responses and for a mythical response set "foo" is:

```
otp-md5 123 mi1234 ext,foo
```

An example of an extended response using a mythical type named "foo" is:

```
foo:some data:some more data:12345
```

2.2. Requirements

A server compliant with this specification:

1. MUST be able to receive and parse the general form of an extended response
2. MUST be able to receive, parse, and correctly process all extended responses specified in this document
3. MUST process the type field in a case-insensitive manner
4. MUST reject any authentication attempt using an extended response if it does not support that type of response
5. SHOULD provide an appropriate indication to the generator if the response was rejected because of (4)
6. MUST limit the length of the input reasonably
7. MUST accept otherwise arbitrary amounts of whitespace wherever a response allows it
8. MUST be able to receive and correctly process standard OTP responses

A generator compliant with this specification:

1. MUST be able to generate standard OTP responses
2. MUST use standard responses unless an extended challenge has been received for the particular server AND seed
3. MUST generate the type field in lower case
4. MUST NOT send a response type for which the server has not indicated support through an extended challenge

Extension set identifiers and extension type identifiers named with the prefix "x-" are reserved for private use among mutually consenting implementations. Implementations that do not recognise a particular "x-" extension MUST ignore that extension. This means that all "x-" extensions are likely to be non-interoperable with other extensions. Careful consideration should be given to the possibility of a server interacting with with a generator implementation which, although it recognizes a given "x-" extension, uses it for a different purpose. All of the remaining extension namespace is reserved to IANA, which will only officially assign the extension

into this namespace after the IESG approves of such an assignment. During the lifetime of the OTP WG, it is recommended that the IESG consult with the OTP WG prior to approving such an assignment.

3. The "hex" and "word" Responses

There exists a very rare case in which a standard OTP response could be a valid coding in both the hexadecimal and six-word formats. An example of this is the response "ABE ACE ADA ADD BAD A." The solution to this problem mandated by the OTP specification is that compliant servers MUST attempt to parse and verify a standard response in both hexadecimal and six-word formats and must consider the authentication successful if either succeeds.

This problem can be solved easily using extended responses. The "hex" response and the "word" response are two response types that encode an OTP in an extended response that explicitly describes the encoding. These responses start with a type label of "hex" for a hexadecimal OTP and "word" for a six-word coded OTP. These responses contain one argument field that contains a standard OTP response coded in the indicated format.

3.1. Syntax

In UNIX manual page like syntax, the format of these responses could be described as:

```
hex:<hexadecimal number>
word:<six dictionary words>
```

In augmented BNF syntax and with the definitions already provided, the syntax of these responses is:

```
hex-response  = "hex:" hex-64bit NL
hex-64bit     = 16(hex-char *LWSP-char)
hex-char      = ("A" / "B" / "C" / "D" / "E" / "F" /
                 "a" / "b" / "c" / "d" / "e" / "f" /
                 "0" / "1" / "2" / "3" / "4" / "5" /
                 "6" / "7" / "8" / "9")

word-response = "word:" word-64bit NL
word-64bit    = 6(otp-word 1*LWSP-char)
otp-word      = <any valid word in the standard OTP coding
                dictionary>
```

Examples of these responses are:

```
hex:8720 33d4 6202 9172
word:VAST SAUL TAKE SODA SUCH BOLT
```

3.2. Requirements

A server compliant with this specification:

1. MUST process all arguments in a case-insensitive manner

A generator compliant with this specification:

1. SHOULD generate otp-word tokens in upper case with single spaces separating them
2. SHOULD generate hexadecimal numbers using only lower case for letters

4. The "init-hex" and "init-word" Responses

The OTP specification requires that implementations provide a means for a client to re-initialize or change its OTP information with a server but does not require any specific protocol for doing it. Implementations that support the OTP extended responses described in this document MUST support the response with the "init-hex" and "init-word" type specifiers, which provide a standard way for a client to re-initialize its OTP information with a server. This response is intended to be used only by automated clients. Because of this, the recommended form of this response uses the hexadecimal encoding for binary data. It is possible for a user to type an "init-hex" or "init-word" response.

4.1. Syntax

In UNIX manual page like syntax, the format of these responses could be described as:

```
init-hex:<current-OTP>:<new-params>:<new-OTP>
init-word:<current-OTP>:<new-params>:<new-OTP>
```

In augmented BNF syntax and with the definitions already provided, the syntax of the "init-hex" response is:

```
init-hex-response = "init-hex:" current-OTP ":" new-params ":"
                     new-OTP NL

current-OTP     = hex-64bit
new-OTP         = hex-64bit
```

```
new-params      = algorithm SPACE sequence-number SPACE seed
algorithm       = "md4" / "md5" / "sha1"
sequence-number = 4*3DIGIT
seed            = 16*1(ALPHA / DIGIT)
```

In augmented BNF syntax and with the definitions already provided, the syntax of the "init-word" response is:

```
init-word-response = "init-word:" current-OTP ":" new-params ":"
                     new-OTP NL

current-OTP     = word-64bit
new-OTP         = word-64bit

new-params      = algorithm SPACE sequence-number SPACE seed
algorithm       = "md4" / "md5" / "sha1"
sequence-number = 4*3DIGIT
seed            = 16*1(ALPHA / DIGIT)
```

Note that all appropriate fields for the "init-hex" response MUST be hexadecimally coded and that all appropriate fields for the "init-word" response MUST be six-word coded.

Examples of these responses are:

```
init-hex:f6bd 6b33 89b8 7203:md5 499 ke6118:23d1 b253 5ae0 2b7e
init-hex:c9b2 12bb 6425 5a0f:md5 499 ke0986:fd17 cef1 b4df 093e

init-word:MOOD SOFT POP COMB BOLO LIFE:md5 499 ke1235:
ARTY WEAR TAD RUG HALO GIVE
init-word:END KERN BALM NICK EROS WAVY:md5 499 ke1235:
BABY FAIN OILY NIL TIDY DADE
```

(Note that all of these responses are one line. Due to their length, they had to be split into multiple lines in order to be included here. These responses MUST NOT span more than one line in actual use)

4.2. Description of Fields

The current-OTP field contains the (RFC 1938) response to the OTP challenge. The new-params field contains the parameters for the client's new requested challenge and the new-OTP field contains a response to that challenge. If the re-initialization is successful, a server MUST store the new OTP in its database as the last successful OTP received and the sequence number in the next challenge presented by the server MUST be one less than the sequence number specified in the new-params field.

The new-params field is hashed as a string the same way that a seed or secret pass phrase would be. All other field values are hashed in their uncoded binary forms, in network byte order and without any padding.

4.3. Requirements

A server compliant with this specification:

1. SHOULD NOT allow a user to use the same value for their seed and secret pass phrase.
2. MUST disable all OTP access to any principal whose sequence number would be less than one
3. MUST decrement the sequence number if a reinitialization response includes a valid current-OTP, but the server is unable to successfully process the new-params or new-OTP for any reason.

A generator compliant with this specification:

1. SHOULD NOT allow a user to use the same value for their seed and secret pass phrase
2. MUST take specific steps to prevent infinite loops of re-initialization attempts in case of failure
3. SHOULD provide the user with some indication that the re-initialization is taking place
4. SHOULD NOT do a re-initialization without the user's permission, either for that specific instance or as a configuration option
5. SHOULD NOT retry a failed re-initialization without a user's permission
6. SHOULD warn the user if the sequence number falls below ten
7. MUST refuse to generate OTPs with a sequence number below one

5. Security Considerations

All of the security considerations for the OTP system also apply to the OTP system with extended responses.

These extended responses, like OTP itself, do not protect the user against active attacks. The IPsec Authentication Header (RFC-1826) (or another technique with at least as much strength as IPsec AH) SHOULD be used to protect against such attacks.

The consequences of a successful active attack on the re-initialization response may be more severe than simply hijacking a single session. An attacker could substitute his own response for

that of a legitimate user. The attacker may then be able to use the OTP system to authenticate himself as the user at will (at least until detected).

Failure to implement server requirement 3 in section 4.3 opens an implementation to an attack based on replay of the current-OTP part of the response.

6. Acknowledgments

Like RFC 1938, the protocol described in this document was created by contributors in the IETF OTP working group. Specific contributions were made by Neil Haller, who provided input on the overall design requirements of a re-initialization protocol, Denis Pinkas, who suggested several modifications to the originally proposed re-initialization protocol, and Phil Servita, who opened the debate with the first real protocol proposal and provided lots of specific input on the design of this and earlier protocols. The extensions to the OTP challenge were suggested by Chris Newman and John Valdes.

Randall Atkinson and Ted T'so also contributed their views to discussions about details of the protocol extensions in this document.

References

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

[RFC 1825] Atkinson, R., "Security Architecture for the Internet Protocol," RFC 1825, August 1995.

[RFC 1938] Haller, N. and C. Metz, "A One-Time Password System," RFC 1938, May 1996.

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

Author's Address

Craig Metz
The Inner Net
Box 10314-1936
Blacksburg, VA 24062-0314
(DSN) 354-8590
cmetz@inner.net

Appendix: Reference Responses

The following responses were generated by a development version of the One-Time Passwords in Everything (OPIE) implementation of this specification.

All of these are responses to the challenge:

```
otp-md5 499 ke1234 ext
```

Note that the re-initialization responses use the same secret pass phrase for new and current and a new seed of "ke1235". Also, these responses have been split for formatting purposes into multiple lines; they MUST NOT be multiple lines in actual use.

The secret pass phrase for these responses is:

```
This is a test.
```

The OTP standard hexadecimal response is:

```
5bf0 75d9 959d 036f
```

The OTP standard six-word response is:

```
BOND FOGY DRAB NE RISE MART
```

The OTP extended "hex" response is:

```
hex:5Bf0 75d9 959d 036f
```

The OTP extended "word" response is:

```
word:BOND FOGY DRAB NE RISE MART
```

The OTP extended "init-hex" response is:

```
init-hex:5bf0 75d9 959d 036f:md5 499 ke1235:3712 dcb4 aa53 16c1
```

The OTP extended "init-word" response is:

```
init-word:BOND FOGY DRAB NE RISE MART:md5 499 ke1235:  RED HERD
NOW BEAN PA BURG
```

Full Copyright Statement