HTTP Authorization Implementation for APRS-IS Servers

Peter S. Loveall

ARRL, TAPR, $(ISC)^2$

Author Note

This paper describes the implementation of Base64 encoding in javAPRSSrvr.

Abstract

Implementation of HTTP Authorization using type APRS-IS and Base64 encoding of the login line in the HTTP Authorization header similar to Basic authorization. Base64 encoding is also supported in any authenticated login mechanism on javAPRSSrvr.

Keywords: APRS,Base64,HTTP

HTTP Authorization Implementation for APRS-IS Servers

Requirements

HTTP 1.1 specifies an authorization mechanism which must be supported anytime the 401 response code is used. In the past, javAPRSSrvr and aprsc have not properly provided a WWW-Authenticate header as part of the response because they did not recognize the Authorization header in the request. To remedy this situation, I focused on the most straightforward way to implement authorization in the Authorization header while being compliant yet unique to the HTTP specification. In the process, the same authorization mechanism can be used for non-HTTP logins if implemented properly. As with all new server features, all current and prior clients must be supported by the server and implementation in the client is not required.

Project Scope

javAPRSSrvr is a full APRS-IS server implemented 100% in Java and has been in use since 2003. Through the ensuing years, it has constantly been maintained and available to amateur radio operators worldwide. It was the cornerstone in stabilizing APRS-IS by implementing the q algorithm and other loop detection. javAPRSFilter, an adjunct to javAPRSSrvr created by Roger Bille SM5NRK, is the basis for all server filter commands allowing a client to connect to a server and only receive requested packets and any packets necessary to properly operate an IGate. It has been **critical** that any enhancements to javAPRSSrvr do not require any changes to existing APRS-IS communications and user connectivity.

This project scope specified implementing support for proper use of the 401 response code when responding to HTTP/HTTPS requests with an invalid or missing login line. The HTTP specification states a server MUST provide a WWW-Authenticate header to tell the client what is missing.

If the implementation of the HTTP authorization protocol is done within a broader scope, all login lines could benefit. In fact, by implementing most in the login line parser, this benefit was realized.

HTTP Listener Port

RFC 7235 specifies the HTTP Authentication Framework. It states “The 401 (Unauthorized) status code indicates that the request has not been applied because it lacks valid authentication credentials for the target resource. The server generating a 401 response MUST send a WWW-Authenticate header field (Section 4.1) containing at least one challenge applicable to the target resource.” The format of the WWW-Authenticate header is:

WWW-Authenticate: type realm=”Text”

Since this mechanism is to use existing APRS-IS authentication, I determined the “type” should be APRS-IS. Since the “realm” is simply a textual “hint”, I made the implementation in javAPRSSrvr to be:

WWW-Authenticate: APRS-IS realm="APRS-IS Valid Login"

The next piece, according to the RFC, is for the client to place an Authorization header in the request. For this, I took the implementation for Basic authentication and adapted it for APRS-IS. To do this, the Authorization header is formatted:

Authorization: APRS-IS Base64-encoded-login-line

The Base64 encoding is easy to implement with the myriad of available libraries (I use the java.util.Base64 class), obscures the login line yet accommodates complete encoding without white spaces in the encoded text. The login line that is encoded does not include CR or LF and starts with “user “ (no leading white space). If this is seen within the HTTP headers, it will be used as the login line and any in-stream login lines will be ignored.

I placed the Base64 encoding/decoding in the APRSISLogin class in javAPRSSrvr. By doing so and keying on the mandatory start of the encoded line (dXNlci), I was also able to detect/send a Base64 encoded login line anywhere, on any port.

Upstream Dialer

The upstream dialer and send-only upstream dialer allow for Base64 encoding. On the send-only upstream dialer, if Base64 has been selected and it is an HTTP upstream dialer, the Base64 login line will be sent in an Authorization header. All other dialers will replace the login line with the encoded line in the data stream.

Summary

Because of the modular design of javAPRSSrvr, adding Base64 encoded login line support was relatively simple and easily maintained. RFC 7235 requirements have been met with this implementation as well as adding Base64 encoding/decoding availability to non-HTTP ports and connections.

The implementation meets the requirements of being additive without affecting current software while providing developers another standards-based (and therefore publicly supported) method of logging the client into APRS-IS. It is **NOT** secure but the login line can now be sent encoded instead of in plain text.

I have reached out to the owner of the aprsc software to encourage inclusion of this login method on that software as well.

References

Overview (Java Platform SE 8) (oracle.com)

RFC 7235 - Hypertext Transfer Protocol (HTTP/1.1): Authentication

Connecting to APRS-IS