Basic vs Digest

Adrian Kosmaczewski

2008-07-07

In the series of highly boring posts ;) here's another one; in this case, a simple explanation of two different authentication protocols available in the HTTP standard.

HTTP Basic Authentication Protocol

This is the simplest HTTP Authentication protocol available:

- The browser sends a request to a protected resource: `GET /index.html`
- The server looks for the "Authenticated" header in the request; since it does not find it, it replies back with a response with the 401 code ("Unauthorized"). The response contains a "WWW-Authenticate" header, with the value "Basic". This response is called a "challenge", and it also contains a "realm" text, which describes the protected resource in clear text (the "realm" is shown in the pop-up window that usually appears for you to type your password when this protocol is used).
- The browser creates a new request `GET /index.html` that contains an HTTP_AUTHORIZATION header, whose value is the word "Basic" followed by the 'username:password' string encoded in base 64. This algorithm is a two-way algorithm: you can retrieve the username and password from the base 64-encoded string.
- The server receives this response, and since base 64 is a two-way algorithm, it compares the MD5 (or SHA1) password hash to the one stored in the database. If they are the same, the request is processed. Otherwise, the user gets a 401 again.

The advantage of this protocol is that it is simpler to implement, but the trade-off is that any malicious user sniffing on the network can retrieve the username:password combo and use the base 64 algorithm to get the original values. All of this makes this protocol relatively insecure.

HTTP Digest Authentication Protocol

Simply put, the HTTP Digest Authentication protocol goes like this:

- The browser sends a request to a protected resource: `GET /index.html`

- The server looks for the "Authenticated" header in the request; since it does not find it, it replies back with a response with the 401 code ("Unauthorized"). The response contains a "WWW-Authenticate" header, with the value "Digest" followed by two long MD5 strings ("nonce" and "opaque") generated specifically for this request, and that the browser will use to answer back to the server. This response is called a "challenge", and it also contains a "realm" text, which describes the protected resource in clear text (the "realm" is shown in the pop-up window that usually appears for you to type your password when this protocol is used).
- The browser creates a new request GET /index.html but this time it will not send the username/password combo in clear text (like in the case of HTTP Basic Authentication): it will "hash" this information, together with the "nonce" and "opaque" values received from the server, together with the "realm", using the MD5 hashing algorithm. This is a username/realm/password hash, so to speak.
- The server receives this response, and since MD5 is a one-way algorithm (you cannot, theoretically, find the original password from all the blah-blah sent in step 3), it compares the username/realm/password hash to the one stored in the database. If they are the same, the request is processed. Otherwise, the user gets a 401 again.

The advantage of the HTTP Digest Authentication protocol is that the key exchange between client and server is done in encrypted hashes. Even better, the server does not store the password per se, but a complex mix of strings all hashed together. This makes the protocol a very secure option, but at the same time, it requires more processing time and code to execute.

However, the HTTP Digest Authentication protocol implies several design issues:

- The username/realm/password combo is encoded and stored using the "realm" string. So this means that if the realm changes, all the passwords become invalid. And since MD5 is a "one-way" algorithm, you cannot retrieve the passwords, and your users must retype them. Be careful!
- Since the default user classes in most web programming toolkits (Django, Rails, etc) just store an MD5 (or SHA1) hash of the password, to use HTTP Digest Authentication you must have a separate database field, somewhere, to store the username/realm/password combo hash.