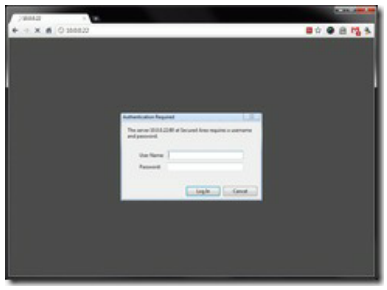
HTTP Basic Access Authentication iRule Style

George Watkins, 2010-30-09

I started working on an administrator control panel for my previous Small URL Generator Tech Tip (part 1 and part 2) and realized that we probably didn't want to make our Small URL statistics and controls viewable by everyone. That led me to make a decision as to how to secure this information. Why not the old, but venerable HTTP basic access authentication? It's simple, albeit somewhat insecure, but it works well and is easy to deploy.

First, a little background on how this mechanism works: First, a user places a GET request for a page without providing authentication. The server then responds with a "401 Authorization Required" status code and the authentication type and realm specified by the WWW-Authenticate header. In our case we will be using basic access authentication and we've arbitrarily named our realm "Secured Area". The user will then provide a username and password. The client will then join the username and password with a colon and apply base-64 encoding to the concatenated string. The client then presents this to the server via the Authorization header. If the credentials are verified by the server, the client will then be granted access to the "Secured Area".



Authentication Required for "Secured Area"

This transaction normally takes place between the client and application server, but we can emulate this functionality using an iRule. The mechanism is rather simple and easy to implement. We need to look for the client Authorization header and if present, verify the credentials. If the credentials are valid, grant access to the virtual server's content, if not, display the authentication box and then repeat the process. On the BIG-IP side, we are storing the username and MD5-digested password in a data group (which we aptly named authorized_users). While this is not as secure as a salted hash, it does provide some security beyond storing the credentials in plain text on our BIG-IP. Once we took these elements into consideration, this is the iRule we developed:

```
1: when HTTP_REQUEST {
2:     binary scan [md5 [HTTP::password]] H* password
3:
4:     if { [class lookup [HTTP::username] $::authorized_users] equals $password } {
5:         log local0. "User [HTTP::username] has been authorized to access virtual server [virtual name]"
6:
7:         # Insert iRule-based application code here if necessary
8:     } else {
9:         if { [string length [HTTP::password]] != 0 } {
10:            log local0. "User [HTTP::username] has been denied access to virtual server [virtual name]"
11:        }
12:
13:        HTTP::respond 401 WWW-Authenticate "Basic realm=\"Secured Area\""
14:    }
15: }
```

There are a couple different ways this iRule could be implemented. It can be applied as-is directly to any HTTP virtual and begin protecting the virtual's contents. It can also be used to secure an iRule-based application. In which case the application code would need to be encapsulated where the comment is located in the above code. I will be publishing a more functional example of this in the near future, but you can start using it now if you have such a necessity.

Earlier we discussed the inherent insecurity of basic access authentication due to the username and password being transmitted in plain text. To combat this, you'll probably want to conduct this transaction over an SSL secured connection. Additionally you'll want to generate the passwords as MD5 hashes before adding them to your data group. Here are a few ways to accomplish this:

Bash

```
% echo -n "password" | md5sum

    => 5f4dcc3b5aa765d61d8327deb882cf99  -
```

Perl

```
% perl -e "use Digest::MD5 'md5_hex'; print md5_hex("password")

    => 5f4dcc3b5aa765d61d8327deb882cf99
```

Ruby (using irb)

```
irb(main):001:0> require "md5"
    => true
irb(main):002:0> MD5::md5("password")
    => #<MD5: 5f4dcc3b5aa765d61d8327deb882cf99>
```

While HTTP basic access authentication may not be the best authentication method for every case, it definitely has its advantages. It is easy to deploy (and even easier via an iRule), provides basic authentication without having to configure or depend on an external authentication service, and is supported by any browser developed in the last 15 years. Through the use of different data groups you can easily separate access to different virtuals or provide a simple SSO (Single Sign-On) solution for a number of virtual servers. We hope you find this tidbit of code to be useful in your environment. Stay tuned for more extensive examples and usages of this tech tip in the near future!

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.	F5 Networks	F5 Networks Ltd.	F5 Networks
Corporate Headquarters	Asia-Pacific	Europe/Middle-East/Africa	Japan K.K.
info@f5.com	apacinfo@f5.com	emeainfo@f5.com	f5j-info@f5.com