Client Cert Fingerprint Matching via iRules

Colin Walker, 2011-31-01

Client cert authentication is not a new concept on DevCentral, it's something that has been covered before in the forums, wikis and Tech Tips. Generally speaking it means that you're receiving a request from a client, and want to authenticate them, as is often the case. Rather than asking for a userID and password, though, you're requesting a certificate that only authorized clients should have access to. In this way you're able to allow seamless access to a resource without forcing a challenge response for the user or application, but still ensuring security is enforced. That's the short version.

So that's cert authentication, but what is a cert fingerprint? A cert fingerprint is exactly what it sounds like, a unique identifier for a particular certificate. In essence it's a shorter way to identify a given certificate without having the entirety of the cert. A fingerprint is created by taking a digest of the entire DER encoded certificate and hashing it in MD5 or SHA-1 format. Fingerprints are often represented as hex strings to be more human readable. The process looks something like this:

> **Command:** `openssl x509 -in cert.pem -noout -fingerprint`
>
> **Output:** `3D:95:34:51:24:66:33:B9:D2:40:99:C0:C1:17:0B:D1`

This can be useful in many cases, especially when wanting to store a list of viable certificates without storing the entirety of the certs themselves. Say, for instance, you want to enable client cert authentication wherein a user connects to your application and both client and server present certificates. The authentication process would happen normally and assuming all checked out access would be granted to the connecting user. What if, however, you only wanted to allow clients with a certain list of certs access? Sure you could store the entire client certificate in a database somewhere and do a full comparison each time a request is made, but that's both a bit of a security issue by having the individual client certificates stored in the auth DB itself, and a hassle.

A simpler method for limiting which certs to allow would be to store the fingerprints instead. Since the fingerprints are unique to the certificate they represent, you can use them to enforce a limitation on which client certificates to allow access to given portions of your application. Why do I bring this up? Obviously there's an iRule for that.

Credit for this example goes to one of our outstanding Field Engineers out of Australia, Cameron Jenkins. Cameron ended up whipping together an iRule to solve this exact problem for a customer and was kind enough to share the info with us here at DevCentral. Below is a sanitized version of said iRule:

```
1: when CLIENTSSL_HANDSHAKE {
2:   set subject_dn [X509::subject [SSL::cert 0]]
3:   set cert_hash [X509::hash [SSL::cert 0]]
4:   set cSSLSubject [findstr $subject_dn "CN=" 0 ","]
5:
6:   log local0. "Subject = $subject_dn, Hash = $cert_hash and $cSSLSubject"
7:
8:   #Check if the client certificate contains the correct CN and Thumbprint from the list
9:   set Expected_hash [class lookup $cSSLSubject mythumbprints]
10:
11:  if { $Expected_hash != $cert_hash } {
12:    log local0. "Thumbprint presented doesn't match mythumbprints. Expected Hash = $Expected_hash, Hash received = $cert_hash"
13:    reject
14:  }
15: }
```

As you can see the iRule is quite reasonable for performing such a complex task. Effectively what's happening here is we're storing the relevent data, the Cert's subject and fingerprint, or hash, as it's referred to in our X509 commands, in local variables. Then we're performing a class lookup against a data group that's filled with all of the valid fingerprints that we want to have access to our application. We're using the subject to perform the lookup, and the result will be what we expect the fingerprint of that certificate to be, based on the subject supplied. Then, if that expected hash doesn't match the actual hash presented by the client, we reject the connection, thereby enforcing access as desired.

Related Articles

- Clientless FirePass Login via the command line using client ...
- 26 Short Topics about Security: Stats, Stories and Suggestions
- Configuring SSL Communcations with Apache SOAP > DevCentral > F5 ...
- Manipulating Header or Content Data > DevCentral > F5 DevCentral ...
- Add root CA to ca-bundle? - DevCentral - F5 DevCentral > Forums ...

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

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com