Two-Factor Authentication With Google Authenticator And APM

George Watkins, 2012-21-02

Introduction

Two-factor authentication (TFA) has been around for many years and the concept far pre-dates computers. The application of a keyed padlock and a combination lock to secure a single point would technically qualify as two-factor authentication: “something you have,” a key, and “something you know,” a combination. Until the past few years, two-factor authentication in its electronic form has been reserved for high security environments: government, banks, large companies, etc. The most common method for implementing a second authentication factor has been to issue every employee a disconnected time-based one-time password hard token. The term “disconnected” refers to the absence of a connection between the token and a central authentication server. A “hard token” implies that the device is purpose-built for authentication and serves no other purpose. A soft or “software” token on the other hand has other uses beyond providing an authentication mechanism. In the context of this article we will refer to mobile devices as a soft tokens. This fits our definition as the device an be used to make phone calls, check email, surf the Internet, all in addition to providing a time-based one-time password.

A time-based one-time password (TOTP) is a single use code for authenticating a user. It can be used by itself or to supplement another authentication method. It fits the definition of “something you have” as it cannot be easily duplicated and reused elsewhere. This differs from a username and password combination, which is “something you know,” but could be easily duplicated by someone else. The TOTP uses a shared secret and the current time to calculate a code, which is displayed for the user and regenerated at regular intervals. Because the token and the authentication server are disconnected from each other, the clocks of each must be perfectly in sync. This is accomplished by using Network Time Protocol (NTP) to synchronize the clocks of each device with the correct time of central time servers.

Using Google Authenticator as a soft token application makes sense from many angles. It is low cost due to the proliferation of smart phones and is available from the “app store” free of charge on all major platforms. It uses an open standard (defined by RFC 4226), which means that it is well-tested, understood, secure. Calculation as you will later see is well-documented and relatively easy to implement in your language of choice (iRules in our case). This process is explained in the next section.

This Tech Tip is a follow-up to Two-Factor Authentication With Google Authenticator And LDAP. The first article in this series highlighted two-factor authentication with Google Authenticator and LDAP on an LTM. In this follow-up, we will be covering implementation of this solution with Access Policy Manager (APM). APM allows for far more granular control of network resources via access policies. Access policies are rule sets, which are intuitively displayed in the UI as flow charts. After creation, an access policy is applied to a virtual server to provide security, authentication services, client inspection, policy enforcement, etc. This article highlights not only a two-factor authentication solution, but also the usage of iRules within APM policies. By combining the extensibility of iRules with the APM’s access policies, we are able to create virtually any functionality we might need.

Note: A 10-user fully-featured APM license is included with every LTM license. You do not need to purchase an additional module to use this feature if you have less than 10 users.

Calculating The Google Authenticator TOTP

The Google Authenticator TOTP is calculated by generating an HMAC-SHA1 token, which uses a 10-byte base32-encoded shared secret as a key and Unix time (epoch) divided into a 30 second interval as inputs. The resulting 80-byte token is converted to a 40-character hexadecimal string, the least significant (last) hex digit is then used to calculate a 0-15 offset. The offset is then used to read the next 8 hex digits from the offset. The resulting 8 hex digits are then AND'd with 0x7FFFFFFF (2,147,483,647), then the modulo of the resultant integer and 1,000,000 is calculated, which produces the correct code for that 30 seconds period.

Base32 encoding and decoding were covered in my previous Tech Tip titled Base32 Encoding And Decoding With iRules. The Tech Tip details the process for decoding a user's base32-encoded key to binary as well as converting a binary key to base32.

The HMAC-SHA256 token calculation iRule was originally submitted by Nat to the Codeshare on DevCentral. The iRule was slightly modified to support the SHA-1 algorithm, but is otherwise taken directly from the pseudocode outlined in RFC 2104.

These two pieces of code contribute the bulk of the processing of the Google Authenticator code. The rest is done with simple bitwise and arithmetic functions.

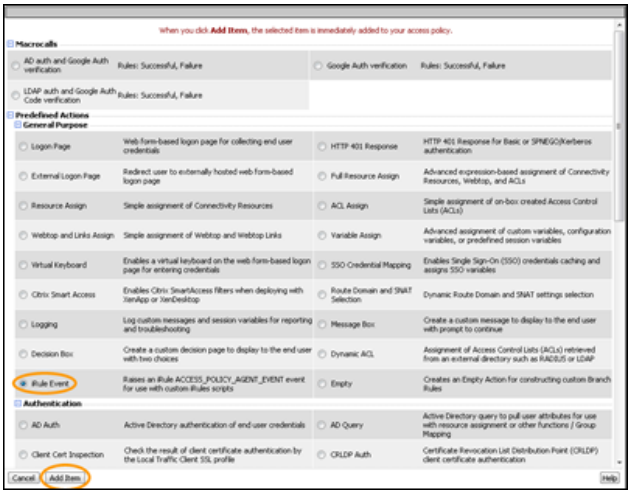
Triggering iRules From An APM Access Policy

Our previously published Google Authenticator iRule combined the functionality of Google Authenticator token verification with LDAP authentication. It was written for a standalone LTM system without the leverage of APM's Visual Policy Editor. The issue with combining these two authentication factors in a single iRule is that their functionality is not mutually exclusive or easily separable. We can greatly reduce the complexity of our iRule by isolating functionality for Google Authenticator token verification and moving the directory server authentication to the APM access policy.

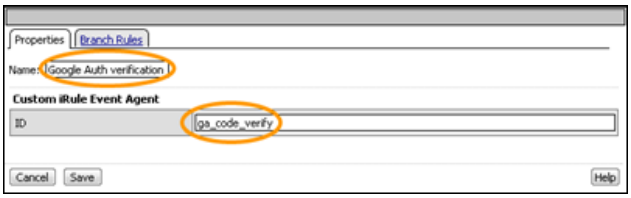
APM iRules differ from those that we typically develop for LTM. iRules assigned to LTM virtual server are triggered by events that occur during connection or payload handling. Many of these events still apply to an LTM virtual server with an APM policy, but do not have perspective into the access policy. This is where we enter the realm of APM iRules. APM iRules are applied to a virtual server exactly like any other iRule, but are triggered by custom iRule event agent IDs within the access policy. When the access policy reaches an iRule event, it will trigger the ACCESS_POLICY_AGENT_EVENT iRule event. Within the iRule we can execute the ACCESS::policy agent_id command to return the iRule event ID that triggered the event. We can then match on this ID string prior to executing any additional code. Within the iRule we can get and set APM session variables with the ACCESS::session command, which will serve as our conduit for transferring variables to and from our access policy. A visual walkthrough of this paragraph is shown below.

iRule Trigger Process

1. Create an iRule Event in the Visual Policy Editor



2. Specify a Name for the object and an ID for the Custom iRule Event Agent

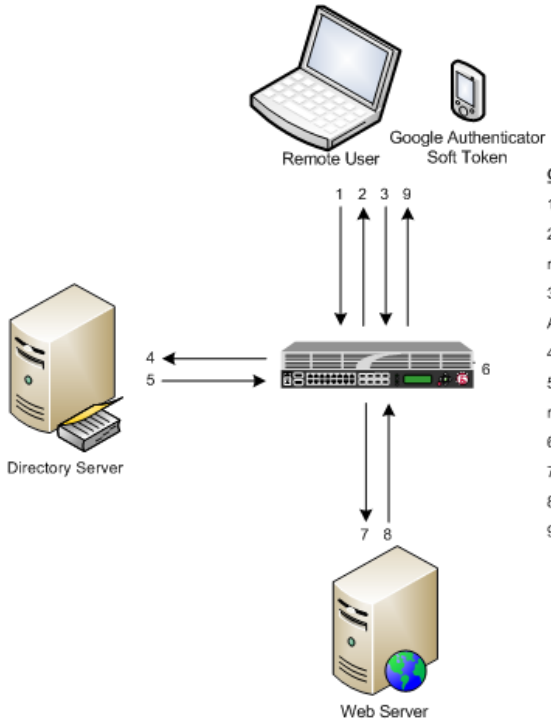


3. Create an iRule with the ID referenced and assign it to the virtual server

```
1: when ACCESS_POLICY_AGENT_EVENT {
2:     if { [ACCESS::policy agent_id] eq "ga_code_verify" } {
3:         # get APM session variables
4:         set username [ACCESS::session data get session.logon.last.username]
5:
6:         ### Google Authenticator token verification (code omitted for brevity) ###
7:
8:         # set APM session variables
9:         ACCESS::session data set session.custom.ga_result $ga_result
10:    }
11: }
```

4. Add branch rules to the iRule Event which read the custom session variable and handle the result

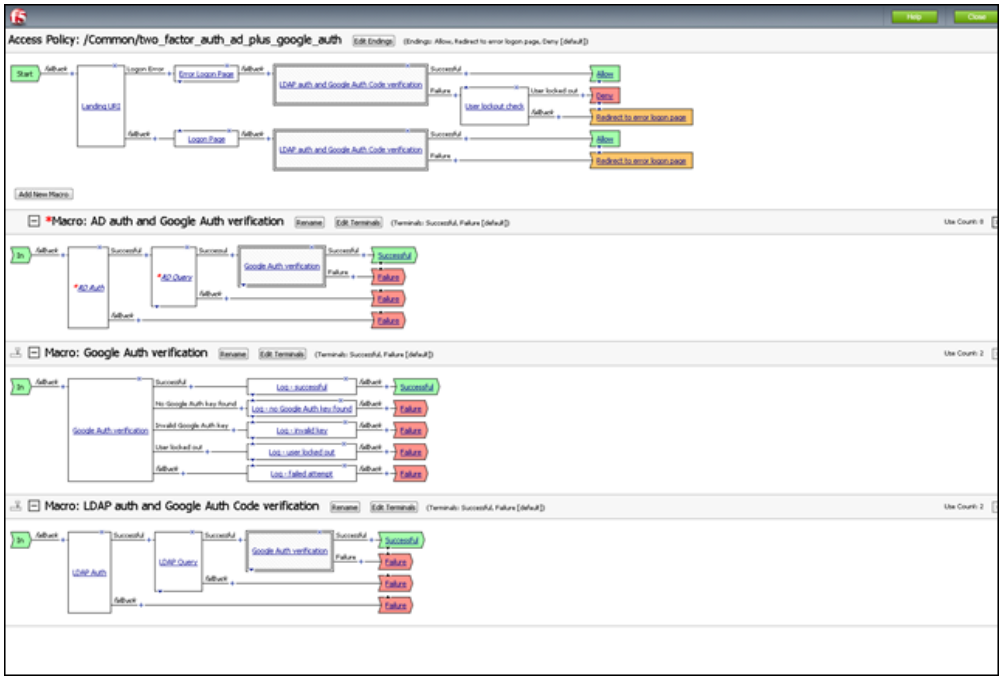
Google Authenticator Two-Factor Authentication Process



Google Authenticator Two-Factor Authentication Process

1. User requests page behind APM-enabled virtual
2. APM does not have an active user session, so a logon page is returned to the user
3. User provides credentials and TOTP provided by their Google Authenticator token
4. Username and password passed to LDAP server
5. Directory server authenticates user with valid credentials and retrieves key from server if using directory server storage
6. Google Authenticator TOTP verified by iRule event
7. User's original request is replayed to origin server
8. Web server returns request to BIG-IP
9. Response returned to newly authenticated user

Two-Factor Authentication Access Policy Overview



Rather than walking through the entire process of configuring the access policy from scratch, we'll look at the policy (available for download at the bottom of this Tech Tip) and discuss the flow. The policy has been simplified by creating macros for the redundant portions of the authentication process: Google Authenticator token verification and the two-factor authentication processes for LDAP and Active Directory.

The "Google Auth verification" macro consists of an iRule event and 5 branch rules. The number of branch rules could be reduced to just two: success and failure. This would however limit our diagnostic capabilities should we hit a snag during our deployment, so we added logging for all of the potential failure scenarios. Remember that these logs are sent to APM reporting (Web UI: Access Policy > Reports) not /var/log/ltm. APM reporting is designed to provide per-session logging in the user interface without requiring grepping of the log files.

The LDAP and Active Directory macros contain the directory server authentication and query mechanisms. Directory server queries are used to retrieve user information from the directory server. In this case we can store our Google Authenticator key (shared secret) in a schema attribute to remove a dependency from our BIG-IP. We do however offer the ability to store the key in a data group as well.

The main portion of the access policy is far simpler and easier to read by using macros. When the user first enters our virtual server we look at the Landing URI they are requesting. A first time request will be sent to the “normal” logon page. The user will then input their credentials along with the one-time password provided by the Google Authenticator token. If the user’s credentials and one-time password match, they are allowed access. If they fail the authentication process, we increment a counter via a table in our iRule and redirect them back to an “error” logon page. The “error” logon page notifies them that their credentials are invalid. The notification makes no reference as to which of the two factors they failed. If the user exceeds the allowed number of failures for a specified period of time, their session will be terminated and they will be unable to login for a short period of time. An authenticated user would be allowed access to secured resources for the duration of their session.

Deploying Google Authenticator Token Verification

This solution requires three components (one optional) for deployment:

1. Sample access policy
2. Google Authenticator token verification iRule
3. Google Authenticator token generation iRule (optional)

The process for deploying this solution has been divided into four sections:

Configuring a AAA server

1. Login to the Web UI of your APM
2. From the side panel select *Access Policy* > *AAA Servers* > *Active Directory*, then the + next to the text to create a new AD server
3. Within the AD creation form you’ll need to provide a *Name*, *Domain Controller*, *Domain Name*, *Admin Username*, and *Admin Password*
4. When you have completed the form click *Finished*

Copy the iRule to BIG-IP and configure options

1. Download a copy of the Google Authenticator Token Verification iRule for APM from the DevCentral CodeShare (hint: this is much easier if you “edit” the wiki page to display the source without the line numbers and formatting)
2. Navigate to *Local Traffic* > *iRules* > *iRule List* and click the + symbol
3. *Name* the iRule '”google_auth_verify_apm,” then copy and paste the iRule from the CodeShare into the *Definition* field
4. At the top of the iRule there are a few options that need to be defined:
    - *lockout_attempts* - number of attempts a user is allowed to make prior to being locked out temporarily (default: 3 attempts)
    - *lockout_period* - duration of lockout period (default: 30 seconds)
    - *ga_code_form_field* - name of HTML form field used in the APM logon page, this field is define in the "Logon Page" access policy object (default: ga_code_attempt)
    - *ga_key_storage* - key storage method for users' Google Authenticator shared keys, valid options include: datagroup, ldap, or ad (default: datagroup)
    - *ga_key_ldap_attr* - name of LDAP schema attribute containing users' key
    - *ga_key_ad_attr* - name of Active Directory schema attribute containing users' key
    - *ga_key_dg* - data group containing user := key mappings
5. Click *Finished* when you’ve configured the iRule options to your liking

Import sample access policy

1. From the Web UI, select *Access Policy* > *Access Profiles* > *Access Profiles List*
2. In the upper right corner, click *Import*
3. Download the sample policy for Two-Factor Authentication With Google Authenticator And APM and extract the .conf from ZIP archive
4. Fill in the *New Profile Name* with a name of your choosing, then select *Choose File*, navigate to the extracted

sample policy and *Open*
5. Click Import to complete the import policy
6. The sample policy's AAA servers will likely not work in your environment, from the *Access Policy List*, click *Edit* next to the imported policy
7. When the *Visual Policy Editor* opens, expand the macro (LDAP or Active Directory auth) that describe your environment
8. Click the *AD Auth* object, select the AD server from the drop-down that was defined earlier in the *AAA Servers* step, then click *Save*
9. Repeat this process for the *AD Query* object

Assign sample policy and iRule to a virtual server

1. From the Web UI, select *Local Traffic* > *Virtual Servers* > *Virtual Server List*, then the create button (+)
2. In the *New Virtual Server* form, fill in the *Name*, *Destination* address, *Service Port* (should be HTTPS/443), next select an *HTTP profile and an SSL Profile (Client).*
3. Next you'll add a *SNAT Profile* if needed, an *Access Profile*, and finally the token verification *iRule*
4. Depending on your deployment you may want to add a pool or other network connectivity resources
5. Finally click *Finished*

At this point you should have a function virtual server that is serving your access policy. You'll now need to add some tokens for your users. This process is another section on its own and is listed below.

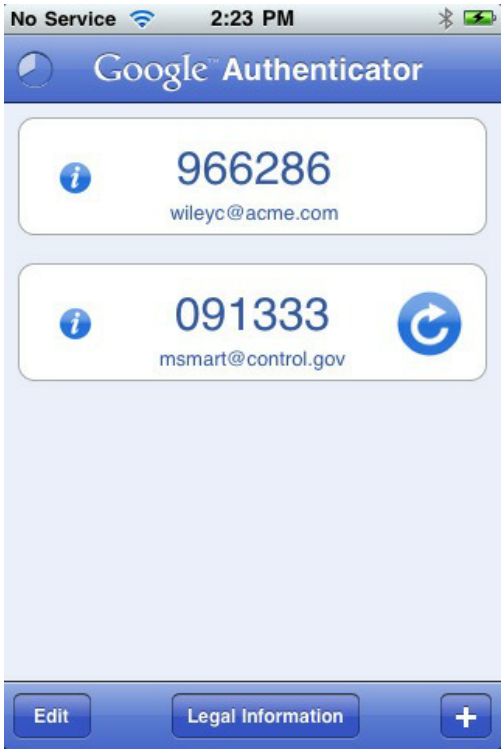
Generating Software Tokens For Users

In addition to the Google Authenticator Token Verification iRule for APM we also wrote a Google Authenticator Soft Token Generator iRule that will generate soft tokens for your users. The iRule can be added directly to an HTTP virtual server without a a pool and accessed directly to create tokens. There are a few available fields in the generator: account, pre-defined secret, and a QR code option. The "account" field defines how to label the soft token within the user's mobile device and can be useful if the user has multiple soft token on the same device (I have 3 and need to label them to keep them straight). A 10-byte string can be used as a pre-defined secret for conversion to a base32-encoded key. We will advise you against using a pre-defined key because a key known to the user is something they know (as opposed to something they have) and could be potentially regenerate out-of-band thereby nullifying the benefits of two-factor authentication. Lastly, there is an option to generate a QR code by sending an HTTPS request to Google and returning the QR code as an image. While this is convenient, this could be seen as insecure since it may wind up in Google's logs somewhere. You'll have to decide if that is a risk you're willing to take for the convenience it provides.

Once the token has been generated, it will need to be added to a data group on the BIG-IP:

1. Navigate to *Local Traffic* > *iRules* > *Data Group Lists*
2. Select *Create* from the upper right-hand corner if the data group does not yet exist. If it exists, just select it from the list.
3. *Name* the data group "google_auth_keys" (data group name can be changed in the beginning section of the iRule)
4. The type of data group will be *String*
5. Type the "username" into the *String* field and paste the "Google Authenticator key" into the *Value* field
6. Click *Add* and the username/key pair should appear in the list as such: user := ONSWG4TFOQYTEMZU
7. Click *Finished* when all your username/key pairs have been added.

Your user can scan the QR code or type it into their device manually. After they scan the QR code, the account name should appear along with the TOTP for the account. The image below is how the soft token appears in the Google Authenticator iPhone application:



Once again, do not let the user leave with a copy of the plain text key. Knowing their key value will negate the value of having the token in the first place. Once the key has been added to the BIG-IP, the user's device, and they've tested their access, destroy any reference to the key outside the BIG-IPs data group.If you're worried about having the keys in plain text on the BIG-IP, they can be encrypted with AES or stored off-box in LDAP and only queried via secure connection. This is beyond the scope of this article, but doable with iRules.

Code

Google Authenticator Token Verification iRule for APM – Documentation and code for the iRule used in this Tech Tip

Google Authenticator Soft Token Generator iRule – iRule for generating soft tokens for users

Sample Access Policy: Two-Factor Authentication With Google Authenticator And APM – APM access policy

Reference Materials

RFC 4226 - HOTP: An HMAC-Based One-Time Password Algorithm

RFC 2104 - HMAC: Keyed-Hashing for Message Authentication

RFC 4648 - The Base16, Base32, and Base64 Data Encodings

SOL3122: Configuring the BIG-IP system to use an NTP server using the Configuration utility – Information on configuring time servers

Configuration Guide for BIG-IP Access Policy Manager – The "big book" on APM configurations

Configuring Authentication Using AAA Servers – Official F5 documentation for configuring AAA servers for APM

Troubleshooting AAA Configurations – Extra help if you hit a snag configuring your AAA server

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