Two-Factor Authentication With Google Authenticator And LDAP

George Watkins, 2011-20-12

Introduction

Earlier this year Google released their time-based one-time password (TOTP) solution named Google Authenticator. A TOTP is a single-use code with a finite lifetime that can be calculated by two parties (client and server) using a shared secret and a synchronized clock (see RFC 4226 for additional information). In the case of Google Authenticator, the TOTP are generated using a software (soft) token on a mobile device. Google currently offers applications for the Apple iPhone, Android-based devices, and Blackberry handsets. A user authenticating with a Google Authenticator-enabled service will require the possession of this software token. In order for the token to be effective, it must not be able to be duplicated and the shared secret should be closely guarded.

Google Authenticator's soft token solution offer a number of advantages over other commercially available solutions. It is free to use (all applications are free to download), the TOTP algorithm is open source, well-known, and well-tested, and finally it does not require a dedicated server for processing tokens. While certain potential weakness in SHA-1 have been identified, none of them can be exploited within the 30-second timeframe of the TOTP's usability. For all intents and purposes, SHA-1 is reasonably secure, well-tested, and purpose-appropriate for this application. The algorithm however is only as secure as the users and administrators are at protecting the shared secret used in token processing.

Calculating The Google Authenticator TOTP

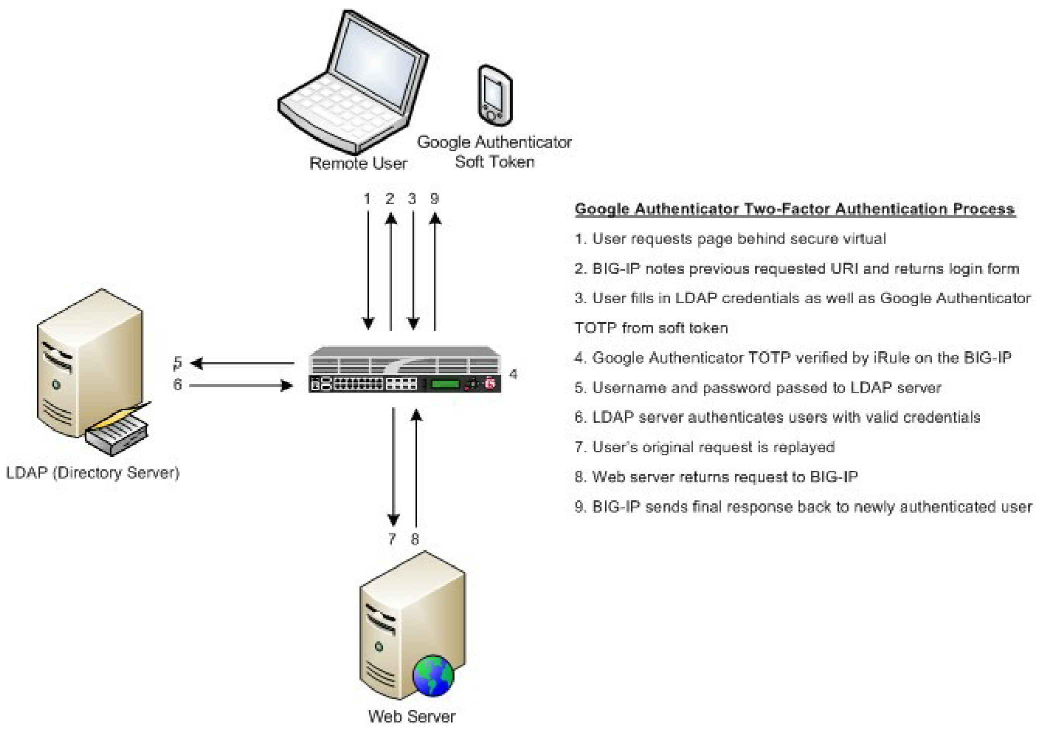
The Google Authenticator TOTP is calculated by generating an HMAC-SHA1 token, which uses a 10-byte base32-encoded shared secret as a key and Unix time (epoch) divided into a 30 second interval as inputs. The resulting 80-byte token is converted to a 40-character hexadecimal string, the least significant (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.

Google Authenticator Two-Factor Authentication Process



Google Authenticator Two-Factor Authentication Process

1. User requests page behind secure virtual
2. BIG-IP notes previous requested URI and returns login form
3. User fills in LDAP credentials as well as Google Authenticator TOTP from soft token
4. Google Authenticator TOTP verified by iRule on the BIG-IP
5. Username and password passed to LDAP server
6. LDAP server authenticates users with valid credentials
7. User's original request is replayed
8. Web server returns request to BIG-IP
9. BIG-IP sends final response back to newly authenticated user

Installing Google Authenticator Two-Factor Authentication

The installation of Google Authenticator two-factor authentication on your BIG-IP is divided into six sections: creating an LDAP authentication configuration, configuring an LDAP (Active Directory) authentication profile, testing your authentication profile, adding the Google Authenticator iRule and “user_to_google_auth” mapping data group, attaching iRule to the authentication profile, and finally generating soft tokens for your users. The process is broken out into steps as trying to complete all the sections in tandem can be difficult to troubleshoot.

Creating An LDAP (Active Directory) Authentication Configuration

The LDAP profile we will configure will be extremely basic: no SSL, no Active Directory, etc. A detailed walkthrough for more advanced deployments can be found in our best practices guide: Configuring LDAP remote authentication for Active Directory.

1. Login to your BIG-IP using administrator credentials

2. Navigate to Local Traffic > Profiles > Authentication > Configurations

3. Click “Create” in the upper right-hand corner

4. Select “LDAP” from the “Type” drop-down menu

5. Now fill in the fields with your environment-specific values:

Name: ldap.f5test.local

Type: LDAP

Remote LDAP Tree: dc=f5test, dc=local

Host(s): <IP address(es) of LDAP server(s)>

Service Port: 389 (default)

LDAP Version: 3 (default)

Bind DN: cn=ldap_bind_acct, dc=f5test, dc=local (if your LDAP server allows anonymous binds you may not need this option)

Bind Password: <admin password>

Confirm Bind Password: <admin password>

6. Click “Finished” to save the configuration

Configuring An LDAP (Active Directory) Authentication Profile

1. Navigate to Local Traffic > Profiles > Authentication > Profiles

2. Click “Create” in the upper right-hand corner

3. Select “LDAP” from the “Type” drop-down menu

4. Fill in fields with appropriate values:

Name: ldap.f5test.local

Type: LDAP

Configuation: ldap.f5test.local (select previously named configuration from drop-down)

Rule: (leave this unchecked and not enabled for now, but this is where we will enable the Google Authenticator iRule shortly)

5. Click “Finished”

Test Your Authentication Profile

1. Create a basic HTTP virtual server with your LDAP authentication profile enabled on the virtual

2. Access your virtual from a web browser and you should be prompted with an HTTP Basic Authentication credential form

3. Test with known-working credentials, if everything works you’re good to go, if not you’ll need to troubleshoot the authentication issue

Adding the Google Authenticator iRule

1. Go to the DevCentral Codeshare and download the Google Authenticator iRule

2. Navigate to Local Traffic > iRules > iRule List

3. Click “Create” in the upper right-hand corner

4. Name your iRule “google_authenticator_plus_ldap_two_factor” and paste the iRule into “Definition” section

5. Click “Finished” when you’re done

Attaching The Google Authenticator iRule To Your Authentication Profile

1. Go back to the “Authentication Profile” section by browsing to Local Traffic > Profiles > Authentication > Profiles

2. Select your LDAP profile from the list

3. Now attach select the “google_authenticator_plus_ldap_two_factor” iRule from the “Rule” drop-down

4. Click “Finished”

Generating Software Tokens For Users

In addition to the Google Authenticator iRule 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 “user_to_google_auth” (data group name can be changed in the RULE_INIT section of the Google Authenticator iRule)

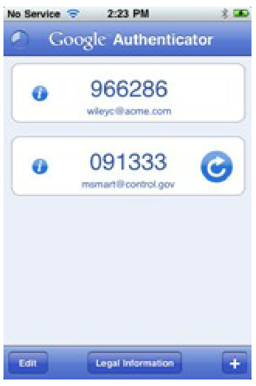
4. The type of data group will be “string”

5. Type the “username” into the “string” field and paste the “Google Authenticator key” into the “value” field

6. Click “Add” and you 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.

Testing and Troubleshooting

There are a lot of moving pieces in this iRule so troubleshooting can be a bit daunting at first glance, but because all of the pieces can be separated into their constituents the problem is usually identified quickly. There are five pieces that make up this solution: the LDAP service, the BIG-IP LDAP profile, the Google Authenticator iRule, the "user_to_google_auth" mapping data group, and finally the soft token. Try to separate them from each other to expedite the troubleshooting process. Here are a few helpful hints in troubleshooting potential issues:

1. Are all the clocks synchronized?

The BIG-IP and LDAP server can be tested from the command line by running 'ntpdate –q pool.ntp.org'. If the clocks are more than a few milliseconds off, they'll need to be adjusted. An NTP server should be configured for all devices. Likewise the user's mobile device must be configured to use network time or else the calculated value will always be wrong. Remember that timezones do not matter when using Unix time.

2. Is basic LDAP working without the iRule attached?

Before ever touching any of the Google Authenticator related iRules, data groups, devices, etc. your LDAP configuration should be in working order. If you're having problems finding the issue, enable "debug logging" at the bottom of the LDAP authentication configuration page on your BIG-IP and tail the logs on your LDAP server. Revisit the best practices guide if you are still unsure about any configuration options.

3. Turn on (or increase) logging for Google Authenticator iRule.

In the RULE_INIT section of the Google Authenticator iRule, there is a debug logging option. Set it to '2' and all actions from the iRule will be logged to /var/log/ltm. If you see one particular area that is consistently hanging, investigate it further.

Conclusion

With every passing day system security becomes a greater concern. Today's attacks are far more sophisticated and costly than those of days past. With all the stories of stolen laptops and other devices in the field, it is a little easier to sleep as a systems administrator knowing that a tech-aware thief has one more hurdle to surpass in an effort to compromise your infrastructure. The implementation costs of deploying two-factor authentication with Google Authenticator in an existing F5 infrastructure are very low assuming your employees have company-issued mobile devices. The cost can be deduced to the man hours required to install this iRule and generate tokens for your users. The cost is almost certainly less than that of a single incident of a compromise account. Until next time, batten down the hatches and get that two-factor project underway that's been on the backburner for two years.

Code and References

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

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

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

SOL11072 - Configuring LDAP remote authentication for Active Directory

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