How to Set Up 2-Factor Authentication in Horizon View with Google Authenticator

Author: Eric Monjoin, emonjoin@ vmware. com
February 2015
Introduction

VMware Horizon View enables you to access a virtual desktop from anywhere, anytime. Horizon offers you the possibility to move from one place to another: to work from your office or from a cybercafé, or from any other place, when you have a network connection that lets you connect to the Horizon View infrastructure.

It’s a wonderful solution. But the challenge is with the external connection: how to protect and secure? How to authorize only some users or groups of users to connect from an external network or Internet?

You can use View Connection Server “Tag” but that implies creating dedicated pools for outside connection and others for inside connection. The other method is to use 2-Factor authentication using proprietary solutions based on software or hardware tokens but for a small company that could be expensive.

This document describes how to secure your external connections and authorize only specific users or groups of users connecting to Horizon View from outside for free, using 2-Factor authentication base on Google Authenticator.

The following method was given to me by one of my oldest customers (and a good friend) so all merit goes to Xavier Montaron CTO of “Mayor of Drancy”.

Prerequisites

Prerequisites are the following:

- vSphere Infrastructure correctly configured for Horizon View
- Horizon View correctly configured (Connection Server, Security Server and Composer)
- Active Directory and DNS (don’t forget to add your RADIUS server(s) in your DNS zone.
- Create a group “GG_S_G)..GOOGLE_..AUTH_DISABLED” (or any other name you want, it will be used to temporarily disable access to specific users) in your Active Directory.
- One or two (for HA) Ubuntu 12.04.04 LTS VMs

Initial setup

The first step is to install and configure at least one Ubuntu VM as a radius server, a second one could be installed to provide redundancy (not described in this document).

1. Using vSphere Web Client or Web Console, create a default “Ubuntu Linux (64-bit)” VM
2. Install a fresh Ubuntu Server 12.04.04  LTS and select OpenSSL package during installation
3. Install VMware Tools and configure networking
4. Update your freshly installed Ubuntu server by running as root
   ```
   # apt-get update
   # apt-get dist-upgrade
   ```

Note: If you don’t want to type “sudo” every time, set a password to the “root” account and after that login as root directly (not recommended for production).

   ```
   # sudo su –
   # passwd
   ```
Likewise setup

Now that the VM is ready, we can start installing and configuring Likewise to add the server in the Active Directory domain and permit domain user authentication.

First, we need to be sure that our server is correctly synchronized in term of date and time with the Active Directory Domain Controller by installing NTP.

```
# apt-get install ntp
```

And then installing Likewise.

```
# apt-get install likewise-open
```

Before joining the server in Active Directory we need to check at least two things:

1. Check that your host files contain the FQDN associated with the VM’s IP.
   
   ```
   # cat /etc/hosts
   ```

   In this example we check that “radius.vmlabs.local” exists, if not, edit the file and modify it as required.
2. Check that you resolve.conf files point to a DNS that manage the domain Zone

```bash
# cat /etc/resolv.conf
```

In this example 192.168.98.10 is a Domain Controller acting also as DNS

```
root@radius:~# cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
# DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
search vmlabs.local
nameserver 192.168.98.10
```

Join the server to the Active Directory domain, by typing the following command:

```bash
# domainjoin-cli join <your domain name> <account _ name>
```

<account name> is an account with “Add Computers Account” Active Directory right.

```
root@radius:~# domainjoin-cli join vmlabs.local adm.eric
Joining to AD Domain: vmlabs.local
With Computer DNS Name: radius.vmlabs.local
adm.eric@VMLABS.LOCAL's password:
SUCCESS
You should reboot this system before attempting GUI logins as a domain user.
root@radius:~#
```

Check that your server is added in Active Directory:

![Active Directory Users and Computers](image)

If you have only one domain, you can change the Likewise configuration to login without specifying your AD domain name at the prompt but just your login, for that type the following command:

```bash
# lwconfig AssumeDefaultDomain true
```

And reboot

```
root@radius:~# lwconfig AssumeDefaultDomain true
root@radius:~# reboot
```
How to Set Up 2-Factor Authentication in
Horizon View with Google Authenticator

Test to login on your server using a domain account (without the "AssumeDefaultDomain" command, login would have been VMLABS\adm_eric)

```
radius login: adm_eric
Password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic x86_64)
  * Documentation: https://help.ubuntu.com/
  System information as of Mon Apr 21 21:36:09 CEST 2014
  System load:  0.0  Processes:  271
  Usage of /:  5.3% of 10.32GB  Users logged in:  0
  Memory usage: 9%  IP address for eth0: 192.168.38.12
  Swap usage:  0%

  Graph this data and manage this system at:
      https://landscape.canonical.com/

The programs included with the Ubuntu system are free software; 
the exact distribution terms for each program are described in the 
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by 
applicable law.

adm_eric@radius:~$ _
```
How to Set Up 2-Factor Authentication in Horizon View with Google Authenticator

Google Authenticator setup

As Google Authenticator is delivered as source code and not binaries, additional packages need to be installed. Type the following command to installed required packages:

```
# apt-get install build-essential git libpam0g-dev libqrencode3
```

Type the following command to download Google Authenticator source code from Google

```
# cd /root
# git clone https://code.google.com/p/google-authenticator
```

Note: The command is executed from the “root” home directory (/root) and all source code is downloaded in /root/google-authenticator

```
root@radius:~# git clone https://code.google.com/p/google-authenticator
Cloning into 'google-authenticator'...
remote: Counting objects: 1056, done.
remote: Total 1056 ( reused 0 ), remote 1056.
remote: Receiving objects: 100% (1056/1056), 2.27 MiB | 100 KiB/s, done.
remote: Resolving deltas: 100% (509/509), done.
root@radius:~# ls
google-authenticator
root@radius:~#
```

Build and install Google-Authenticator binaries:

```
# cd /root/google-authenticator/libpam
# make && make install
```

```
root@radius02:/root/google-authenticator/libpam# make && make install
gcc --std=gnu99 -Wall -O2 -g -fpic -c -fvisibility=hidden -o google-authenticator.o google-authenticator.c
gcc --std=gnu99 -Wall -O2 -g -fpic -c -fvisibility=hidden -o base32.o base32.c
gcc --std=gnu99 -Wall -O2 -g -fpic -c -fvisibility=hidden -o hmac.o hmac.c
gcc --std=gnu99 -Wall -O2 -g -fpic -c -fvisibility=hidden -o sha1.o sha1.c
gcc -o google-authenticator google-authenticator.o base32.o hmac.o sha1.o -ldl

gcc --std=gnu99 -Wall -O2 -g -fpic -c -fvisibility=hidden -o google-authenticator.o google-authenticator.c
gcc -shared -g -o google-authenticator.so google-authenticator.o base32.o hmac.o sha1.o -lpam

gcc --std=gnu99 -Wall -O2 -g -fpic -c -fvisibility=hidden -o demo.o demo.c
gcc -DEMO --std=gnu99 -Wall -O2 -g -fpic -c -fvisibility=hidden -o demo.o demo.c
googles2 --std=gnu99 -Wall -O2 -g -fpic -c -fvisibility=hidden -o demo.o demo.c

gcc -lDynamic -o demo demo.o google-authenticator_demo.o base32.o hmac.o sha1.o -ldl

gcc -DEDEBUG --std=gnu99 -Wall -O2 -g -fpic -c -fvisibility=hidden -o google-authenticator.o google-authenticator.c
gcc -shared -g -o google-authenticator.so google-authenticator.o base32.o hmac.o sha1.o -lpam

gcc --std=gnu99 -Wall -O2 -g -fpic -c -fvisibility=hidden -o google-authenticator.o google-authenticator.c
gcc -shared -g -o google-authenticator.so google-authenticator.o base32.o hmac.o sha1.o -lpam

gcc --std=gnu99 -Wall -O2 -g -fpic -c -fvisibility=hidden -o google-authenticator.o google-authenticator.c
gcc -shared -g -o google-authenticator.so google-authenticator.o base32.o hmac.o sha1.o -lpam

cp google-authenticator/usr/local/bin
root@radius02:/root/google-authenticator/libpam#
```
RADIUS setup

Next step: installation of Radius component using freeradius, for this type:

```
    # apt-get install freeradius
```

Once radius is installed, edit /etc/freeradius/radiusd.conf and modify it to read

```
    user = root  (instead of user = freerad)
    group = root  (instead of user = freerad)
```

Before:

```
    # The server will also try to use "initgroups" to read /etc/groups.
    # It will join all groups where "user" is a member. This can allow
    # for some finer-grained access controls.
    #
    user = freerad
    group = freerad
    # max_request_time: The maximum time (in seconds) to handle a request.
    # Requests which take more time than this to process may be killed, and
    # a REJECT message is returned.
```

After:

```
    # The server will also try to use "initgroups" to read /etc/groups.
    # It will join all groups where "user" is a member. This can allow
    # for some finer-grained access controls.
    #
    user = root
    group = root
    # max_request_time: The maximum time (in seconds) to handle a request.
    # Requests which take more time than this to process may be killed, and
    # a REJECT message is returned.
```

To temporarily disable an account from login using RADIUS, we configure RADIUS to search for a specific group in Active Directory and prevent login from any users present in this group.

Edit the file /etc/freeradius/users

And modify the section "Deny access for a group of users"

```
    # Deny access for a group of users.
    # Note that there is NO 'Fall-Through' attribute, so the user will not
    # be given any additional resources.
    #
    #DEFAULT  Group == "disabled", Auth-Type == Reject
    #          Reply-Message = "Your account has been disabled."
    #
    # This is a complete entry for "Steve". Note that there is no Fall-Through
    # entry so that no DEFAULT entry will be used, and the user will NOT
    # get any attributes in addition to the ones listed here.
```
How to Set Up 2-Factor Authentication in Horizon View with Google Authenticator

To read:

\[
\text{DEFAULT Group} = \"GG_S_GOOGLE_AUTH_DISABLED\", \text{Auth-Type} = \text{Reject} \\
\text{Reply-Message} = \"Your account has been disabled.\"
\]

\[
\text{DEFAULT Auth-Type} = \text{PAM}
\]

**Note:** if ever you create a group with a different name, change it accordingly

Now edit the file `/etc/freeradius/sites-enabled/default`

```
root@radius:/etc/freeradius# vim /etc/freeradius/sites-enabled/default
```

Search for “pam” by typing `/pam`
How to Set Up 2-Factor Authentication in Horizon View with Google Authenticator

And uncomment the line it to read:

```
auth require pam_google_authenticator.so forward_pass
auth required pam_lsass.so use_first_pass
```

Now edit the file `/etc/pam.d/radiusd`

```bash
root@radius:/etc/freeradius:
```
Restart freeradius to activate new settings

```
# service freeradius restart
```

Now edit `/etc/freeradius/clients.conf`

```
# vim /etc/freeradius/client.conf
```

Search for `# client some.host.org`, uncomment the section and modify the file to read:

```
client <fqdn_view_connection_servers> {
    secret = <password you want to use>
    shortname = <hostname of your View Connection Server>
}
```

```
client vcs01.vmlabs.local {
    secret = VMware4all_
    shortname = vcs01
}
```

Again, restart RADIUS:

```
# service freeradius restart
```

Everything is configured on the server at this time, now we have to setup Horizon View to use RADIUS and Google-Authenticator:
View Connection Server setup

Connect to your Horizon View Connection Server as Administrator

Select the Horizon View Connection Server you want to use:

On Authentication tab, select RADIUS as “Advanced Authentication”
1. Check: “Enforce 2-Factor and Windows user name matching”
2. Select: “Create New Authenticator”

3. Specify the Label: example: Google Auth
4. Specify the Hostname/Address: fqdn of your radius server
   1. Specify the Shared Secret: the secret you specified in /etc/freeradius/client.conf
Note: For production usage, you can install and configure a secondary radius server:

Add RADIUS Authenticator

Secondary Authentication Server

- Use a secondary server if primary is unavailable

- Hostname/Address:
- Authentication port: 1812
- Accounting port: 1813
- Authentication type: PAP
- Shared secret:
- Server timeout: 3 seconds
- Max attempts: 5
- Realm prefix:
- Realm suffix:
Adding users to RADIUS

At this point, Horizon View is configured 2-Factors authentication using your Radius server, now you have to add users to RADIUS and authorize them.

On your Radius server, as root, impersonate as the user you want to add and execute google-authenticator

```
# su <username>
# google-authenticator
```

Note: To see all options, type:

```
# google-authenticator -h
```

![QR Code Image]
Answer “Y” to all questions

We can pass all questions as arguments to google-authenticator as this:

```
# google-authenticator -tdf -r 3 -R 30 -w 17 -Q UTF8
```

**Note:** If you encounter issue with “-Q UTF8”, use “-Q ANSI” instead.

As you can see there’s emergency scratch code that can be used in case you don’t have your Token generator with you. This is “one time” codes that are deleted as soon as used.

You can also configure your devices (iOS, Android) using the generated QR code or the secret key.

To simplify the operation, you can create an alias for all users by modifying the file `/etc/skel/.bashrc`

```
root@radius:$ vim /etc/skel/.bashrc
```

Add the following line, so you’ll only have to execute “google-authenticator” instead of using arguments or hitting “enter” or “Y” keys.
Hardening the RADIUS server

At this time, any domain users can login to the RADIUS server using SSH, this certainly not what you want, so to prevent this you need to modify the file /etc/ssh/sshd_config and add at the bottom the following line:

```
DenyGroups domain^users
```

Google Authenticator client setup

From the server side, everything is up and running, now we need to install and configure the Google Authenticator client.

It's available for many OS and device flavor: Android, iPhone/iPod/iPad, Blackberry, Windows
Configuration is relatively simple:
Start “Authenticator” and choose the method that meet your needs:

1. Choose “Scan barcode” and then scan the generated QR Code

![Scan barcode](image)

2. Or choose “Manual Entry” and type the “Secret Key” shown just below the QR Code

![Manual entry](image)

Your token is immediately available

![Authenticator](image)
Testing connection

Now you can test and make a connection on your View Connection Server by providing your login name and password in the form of <AD_PASSWORD><Generated Code>

If everything works fine, the second login screen appears and you have to type your AD password again (this is “As design” by Horizon View whatever 2-Factor authentication method you want to use: RADIUS or RSA).
Appendix A – Setting alias for google-authenticator

Typing the whole line to generate token can be painful if you have a huge number of users, the best solution is to create an alias for it so next time you’ll have to type “google-authenticator” and all options will become implicit.

To do that, edit /etc/skel/.bashrc and add an alias for “google-authenticator” as show below:

```bash
# add these lines to your /etc/skel/.bashrc
alias la='la -all'
alias la='la -a'
alias la='la -U'
alias google-authenticator='google-authenticator -tofo -I "Horizon View" -r 3 -R 30 -w 37 -Q UTF8'
```
Appendix B – Google Authenticator Patch

DISCLAIMER: The following patch is provided “As Is”, without any guarantee or technical support. Use it at your own risk.

By default when you scan the QR Code, the Authenticator client shows the account name as user@servername (eg. eric@radius1). This is not really helpful for the user and can be misunderstood. The following patch lets you add a title for the generated token.

Patching Google Authenticator

To patch Google Authenticator

1. Login as “root”
2. Copy and paste the patch below into a file named add-options.patch (or whatever you want)
3. Clean the previous build
   ```
   # cd ~/google-authenticator/libpam/
   # make clean
   ```
4. Patch Google Authenticator with the provided patch
   ```
   # patch google-authenticator.c < /root/add-options.patch
   ```
5. Rebuild and install our modified Google Authenticator
   ```
   # make && make install
   ```
How to Set Up 2-Factor Authentication in Horizon View with Google Authenticator

Using patched Google Authenticator

If you run "google-authenticator --h" to display available help, you'll notice some new options and the one that keep our interest is “-i”.

```
# google-authenticator -tdfo -i "Horizon View - Home Labs" -l "Eric (Admin)" -r 3 -R 30 -w 17 -Q ANSI
```

You'll obtain a more user friendly display:
How to Set Up 2-Factor Authentication in Horizon View with Google Authenticator

Patch - source code

```c
--- google-authenticator.c 2014-04-03 18:07:40.922741501 +0200
+++ google-authenticator-patch.c 2014-04-03 18:37:44.922720724 +0200
@@ -144,14 +144,18 @@
 static const char *getURL(const char *secret, const char *label,
 -                         char **encoderURL, const int use_totp)
 +                         const char* issuer, char **encoderURL,
 +                         const int use_totp) {
 const char *encodedLabel = urlEncode(label);
 - char *url = malloc(strlen(encodedLabel) + strlen(secret) + 80);
 + const char *encodedIssuer = urlEncode(issuer);
 + char *url = malloc(strlen(encodedLabel) + strlen(encodedIssuer)
 +     + strlen(secret) + 88);
 char totp = 'h';
 if (use_totp) {
     totp = 't';
 } -
 +  sprintf(url, "otpauth://%cotp/%s?secret=%s", totp, encodedLabel,
 +            secret);
 +  sprintf(url, "otpauth://%cotp/%s?secret=%s&issuer=%s", totp,
 +            encodedLabel, secret, encodedIssuer);
 if (encoderURL) {
     const char *encoder = "https://www.google.com/chart?chs=200x200&"
     "chld=M|0&cht=qr&chl=";
@@ -175,12 +179,12 @@
 #define UTF8 BOTTOMHALF   "\xE2\x96\x84"
 static void displayQRCode(const char *secret, const char *label,
 -                        const int use_totp) {
 +                        const char* issuer, const int use_totp)
 {
     if (qr_mode == QR_NONE) {
         return;
     }
     char *encoderURL;
 -    const char *url = getURL(secret, label, &encoderURL, use_totp);
 +    const char *url = getURL(secret, label, issuer, &encoderURL,
 +                           use_totp);
     puts(encoderURL);

     // Only newer systems have support for libqrencode. So, instead
@@ -337,7 +341,9 @@
 " -D, --allow-reuse        Allow reuse of previously used TOTP
 tokens\n"
 " -f, --force              Write file without first confirming
with user\n"
 " -l, --label=<label>      Override the default label in
 \"otpauth://\" URL\n" + " -i, --issuer=<label>     Override the default issuer in
```

PROFESSIONAL SERVICES TECHNICAL GUIDE / 23
How to Set Up 2-Factor Authentication in Horizon View with Google Authenticator

enum { ASK_MODE, HOTP_MODE, TOTP_MODE } mode = ASK_MODE;
enum { ASK_REUSE, DISALLOW_REUSE, ALLOW_REUSE } reuse = ASK_REUSE;

int force = 0, quiet = 0;
int force = 0, quiet = 0, qronly = 0;
int r_limit = 0, r_time = 0;
char *secret_fn = NULL;
char *label = NULL;
char *issuer = NULL;
int window_size = 0;
for (;;) {
    static const char optstring[] = "+hctdfl:qQ:r:R:us:w:W";
    static const char optstring[] = "+hctdfl:i:qQ:r:R:us:w:Wo";
    static struct option options[] = {
        { "help",             0, 0, 'h' },
        { "counter-based",    0, 0, 'c' },
        { "allow-reuse",      0, 0, 'D' },
        { "force",            0, 0, 'f' },
        { "label",            1, 0, 'l' },
        { "issuer",           1, 0, 'i' },
        { "quiet",            0, 0, 'q' },
        { "qronly",           0, 0, 'o' },
        { "qr-mode",          1, 0, 'Q' },
        { "rate-limit",       1, 0, 'r' },
        { "rate-time",        1, 0, 'R' },
    };
    if (optind < argc)
        _exit(EXIT_FAILURE);
    for (;; ) {
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
            _exit(EXIT_FAILURE);
        if (idx-- < argc)
How to Set Up 2-Factor Authentication in Horizon View with Google Authenticator

```c
+ fprintf(stderr, "Duplicate -o option detected or used with -q\n");
+ _exit(1);
+ }
+ quiet = 1;
+ qronly = 1;
+ } else if (!idx--) {
    // qr-mode
    if (qr_mode != QR_UNSET) {
        fprintf(stderr, "Duplicate -Q option detected\n");
@@ -590,13 +614,15 @@
    if (!label) {
        uid_t uid = getuid();
        const char *user = getUserName(uid);
    + label = strdup((char *)user);
    + free((char *)user);
    + }
    + if (!issuer) {
        char hostname[128] = { 0 };
        if (gethostname(hostname, sizeof(hostname)-1)) {
            strcpy(hostname, "unix");
    - label = strcat(strcat(strcpy(malloc(strlen(user) +
            strlen(hostname) + 2),
            user), "]@"), hostname);
    - free((char *)user);
    + issuer = strdup(hostname);
    }
    int fd = open("/dev/urandom", O_RDONLY);
    if (fd < 0) {
@@ -617,12 +643,16 @@
        use_totp = mode == TOTP_MODE;
    } if (!quiet) {
    - displayQRCode(secret, label, use_totp);
    + displayQRCode(secret, label, issuer, use_totp);
        printf("Your new secret key is: %s
", secret);
        printf("Your verification code is %06d
", generateCode(secret,
0));
    printf("Your emergency scratch codes are:\n\n");
    + if (qronly) {
    +     displayQRCode(secret, label, issuer, use_totp);
    + } free(label);
    + free(issuer);
        strcat(secret, "\n");
    if (use_totp) {
            strcat(secret, totp);
```
About the Authors and Contributors

Eric Monjoin, Technical Solution Architect, VMware End User Computing Solutions
Xavier Montaron, CIO, Town Hall of Drancy, France