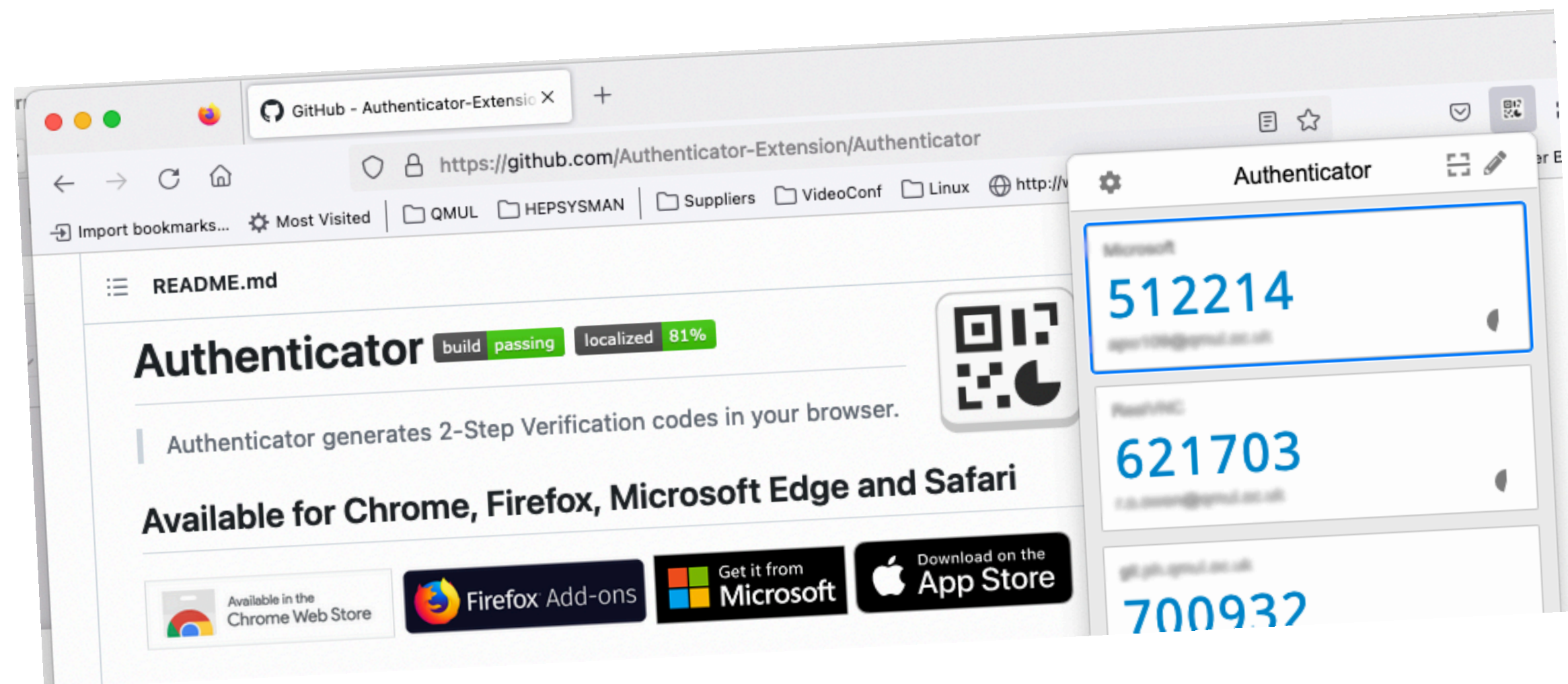


OpenSSH PAM and MFA SSH Testing + Hardening

Alex Owen @ QMUL



```
$ ssh port  
Verification code:  
Password:  
owen@port:~$
```

Passwords just won't do!

- Constant ssh brute force attempts.
- Solutions like fail2ban only go so far.
 - No cluster support (although see a previous talk of mine)
 - IPv6 opens up a larger pool of attack IP's
- Keys are good but can't enforce users use a passphrase
- MFA often mandated by policy

Three Rules of SSH Keys

Rule 1: An ssh key must always have a passphrase.

Rule 2: An ssh key must **ALWAYS** have a passphrase.

Rule 3: You are not ready for Rule 3

When a user/admin is ready...

Rule 3: An ssh key without a passphrase must be

- restricted by IP address
- be bound to a command

AuthenticationMethods


- Key + Password
 - No ability to automate
- Key + Key or Key + Password
 - Now we can automate
- On-site + (Password or key)
 - I'm the admin in a tight spot...
 - let me fix this thing!
- Onetime code + Password
 - Oops I lost my laptop!

`/etc/ssh/sshd_config`

```
AuthenticationMethods publickey,password
```

Or

```
AuthenticationMethods publickey,publickey publickey,password
```

 insert your own ip addresses!

```
Match Address 169.254.0.0/16,fe80::/64,127.0.0.1,::1  
AuthenticationMethods publickey password
```

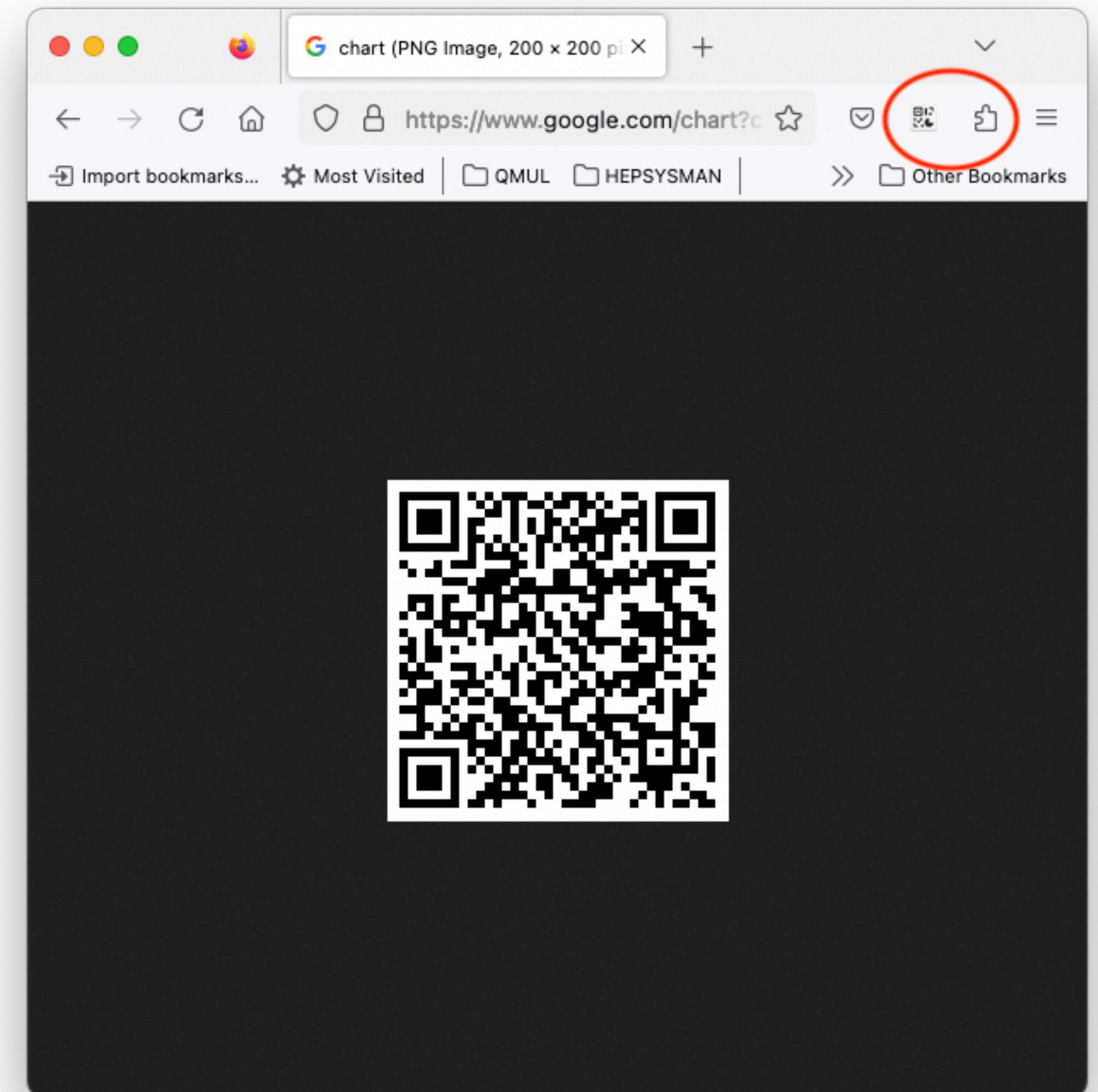
I'm glad you asked... Lets take a look...

MFA - libpam-google-authenticator

- \$pkgmgr install libpam-google-authenticator
- Other MFA options exist
- User setup requires users to run: google-authenticator
- This may or may not scale well to non-admin users



- Say yes to timebased code
- Enter your new code to check
- On mobile scan QR with google-authenticator app or other TOPT app
 - Or use the link in a browser and use Authenticator-Extension (see next slide)
- Or both!



Authenticator-Extension/Authenticator

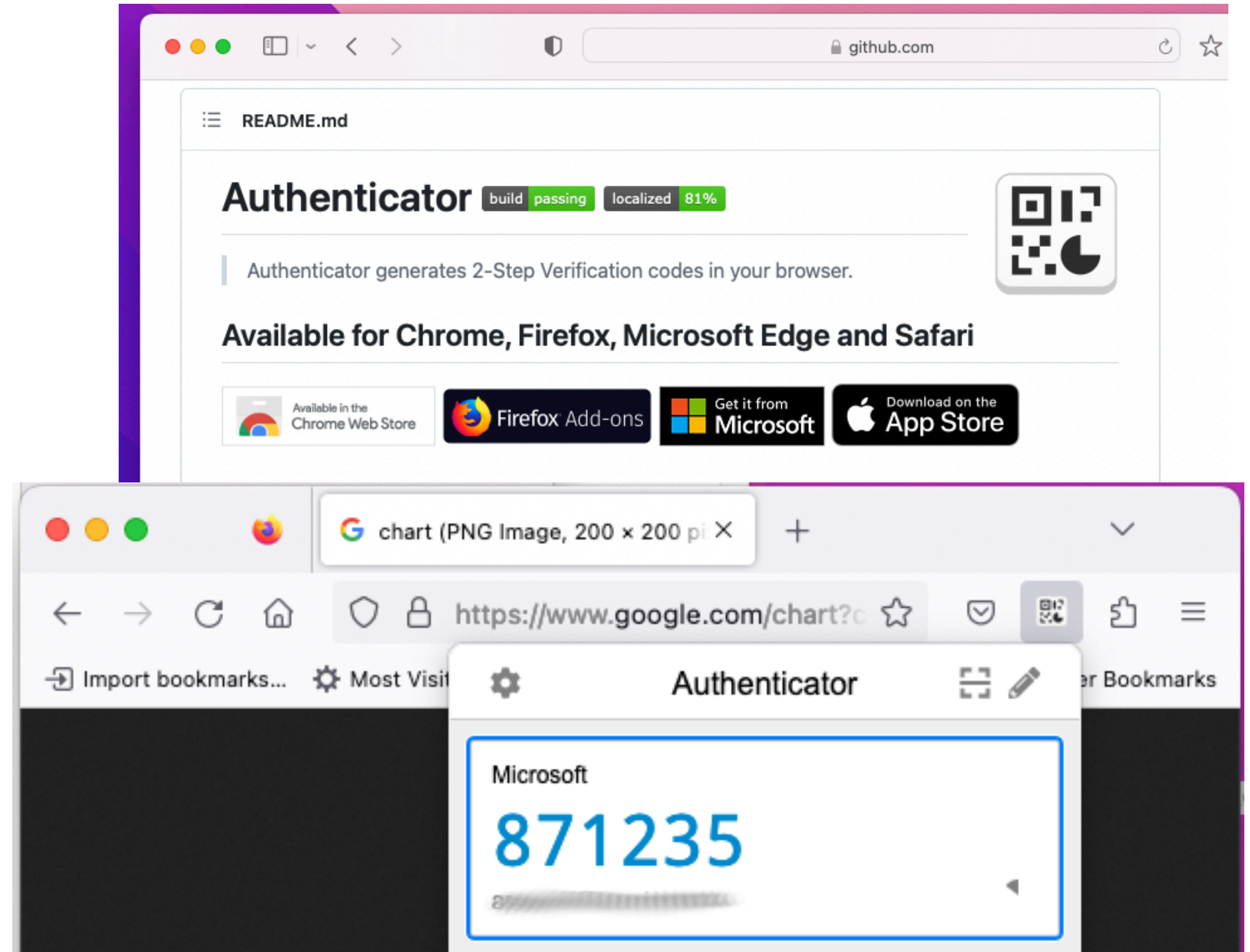
<https://github.com/Authenticator-Extension/Authenticator>

Available for
Chrome
Firefox
Microsoft Edge
Apple Safari

Plugin Icon



Scan Icon



MFA - libpam-google-authenticator

/etc/pam.d/sshd (Debian Syntax)

```
auth required pam_google_authenticator.so no_increment_hotp  
# Standard Un*x authentication.  
@include common-auth
```

/etc/ssh/sshd_config

```
# Change to yes to enable challenge-response passwords  
KbdInteractiveAuthentication yes  
# Now mandate keyboard interactive  
AuthenticationMethods keyboard-interactive
```

Old Standard Login Prompt

```
$ ssh port  
owen@port's password:  
owen@port:~$
```

New keyboard-interactive Login Prompt

```
$ ssh port  
Verification code:  
Password:  
owen@port:~$
```

MFA with only the standard prompt?

```
$ ssh port  
owen@port's password:
```

/etc/pam.d/sshd (Debian Syntax)

```
auth required pam_google_authenticator.so forward_pass no_increment_hotp  
# Standard Un*x authentication.  
@include common-auth
```

/etc/pam.d/common-auth (Debian Syntax)

```
# here are the per-package modules (the "Primary" block)  
auth [success=1 default=ignore] pam_unix.so use_first_pass nullok
```

```
$ ssh port  
owen@port's password: 012345MySecretPassword
```

But I'm never happy!

- On site: Key or password (with standard Prompt)
- Off site: Key+Key or Key+Password or OTP+Password (new prompt)

PAM needs to distinguish between `AuthenticationMethods keyboard-interactive`
and `AuthenticationMethods password`

The answer is in the environment: `SSH_AUTH_INFO_0` **exists for** `keyboard-interactive`

SSH_AUTH_INFO_0

```
publickey ssh-rsa AAAAB3NzaC1yc-blah-blah
publickey ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABA-blah-blah
keyboard-interactive/pam
```


Now I'm happy! Lets Script!

/etc/security/ssh-password.sh

```
#!/bin/bash
####
# Test if ssh is calling PAM for "password" method.
# For keyboard interactive env var SSH_AUTH_INFO_0 will exist
# return 0 if "password" ; return 1 if SSH_AUTH_INFO_0 so keyboard interactive
####
set -e
/usr/bin/printenv | /usr/bin/grep -q ^SSH_AUTH_INFO_ || exit 0
exit 1
```

Now I'm happy! Lets Script!

/etc/security/ssh-used-pubkey.sh

```
#!/bin/bash
####
# Test if ssh already used publickey authentication by parsing env var SSH_AUTH_INFO_0
# return 0 if pubkey used; return 1 otherwise
####
# example of SSH_AUTH_INFO_0 format (multi line env var)
#export SSH_AUTH_INFO_0="publickey ssh-rsa AAAAB3NzaC1yc-blah-blah
#publickey ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABA-blah-blah
#keyboard-interactive/pam"
####
set -e
EXITSTAT=1
while read -r method junk ; do
    if [ "$method" == "publickey" ]; then
        EXITSTAT=0
        break
    fi
done < <(echo "$SSH_AUTH_INFO_0")
exit $EXITSTAT
```

Putting it all together!

AuthenticationMethods **password**

AuthenticationMethods **publickey**

/etc/pam.d/sshd (Debian Syntax)

```
#Skip N rules if exit zero: [success=N default=ignore] pam_exec.so quiet /my/script
auth [success=2 default=ignore] pam_exec.so quiet /etc/security/ssh-password.sh
auth [success=1 default=ignore] pam_exec.so quiet /etc/security/ssh-used-pubkey.sh
auth required pam_google_authenticator.so no_increment_hotp
# Standard Un*x authentication.
@include common-auth
```


Putting it all together!

/etc/ssh/sshd_config

```
# Change to yes to enable challenge-response passwords
```

```
KbdInteractiveAuthentication yes
```

```
#####
```

```
# OTP/Password setup
```

```
#####
```

```
# NB: This is a complex setup... relies on the config in /etc/pam.d/sshd and you following the rules!!!
```

```
# HERE ARE THE RULES - there is only one!:
```

```
# (1) Do NOT mix keyboard-interactive and password in one AuthenticationMethods line - this may be insecure
```

```
# keyboard-interactive should prompt for OTP and password (unless preceded by publicly then just password)
```

```
# password will ONLY prompt for password
```

```
# never have password in an AuthenticationMethods line if you want enforce OTP
```

```
#####
```

```
#NB with correct pam setup in /etc/pam.d/sshd keyboard-interactive is OTP+password
```

```
# while password is only-password
```

```
AuthenticationMethods publickey,publickey publickey,keyboard-interactive keyboard-interactive
```

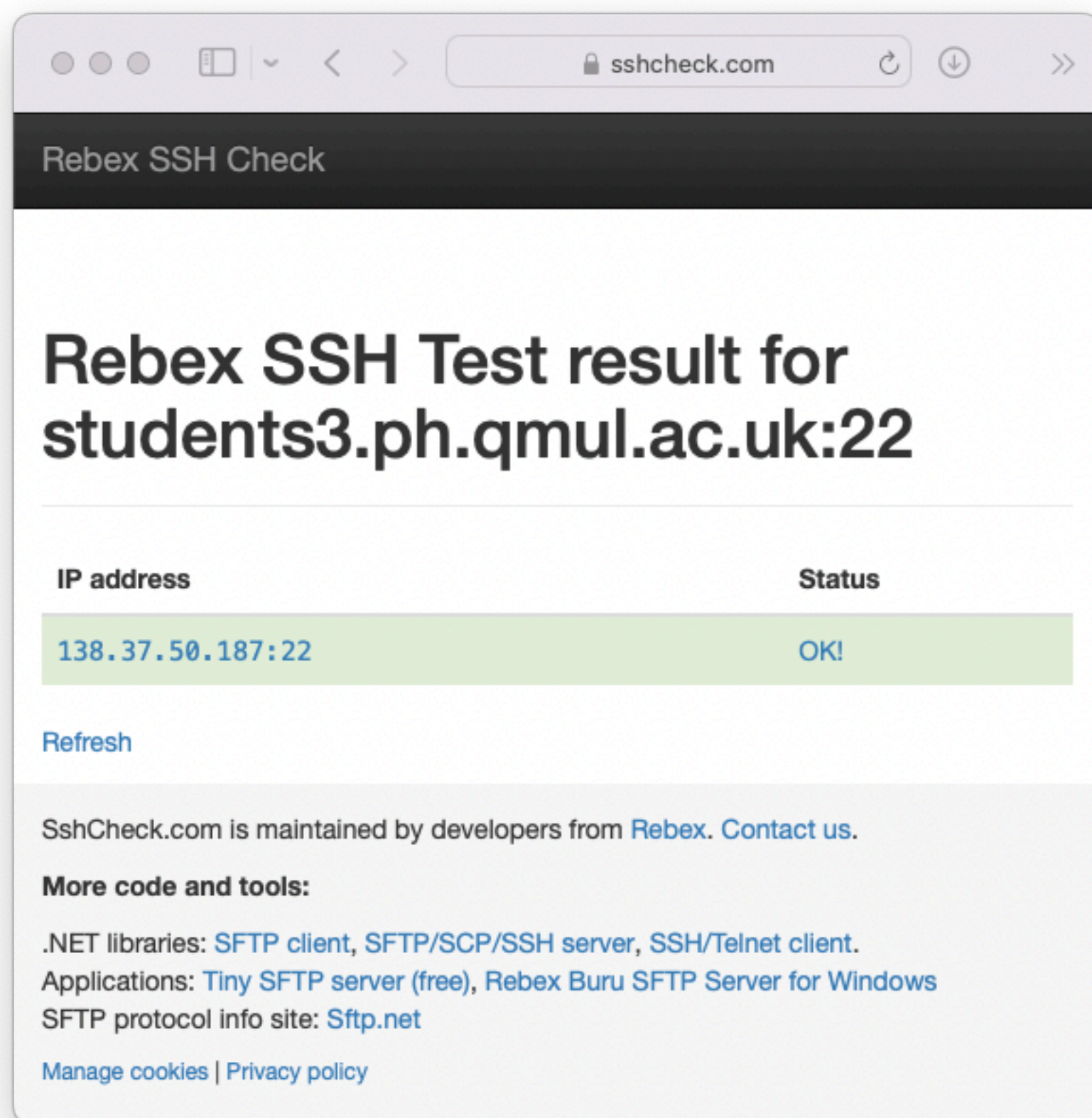
```
#On site exception
```

```
Match Address 169.254.0.0/16,fe80::/64,127.0.0.1,::1
```

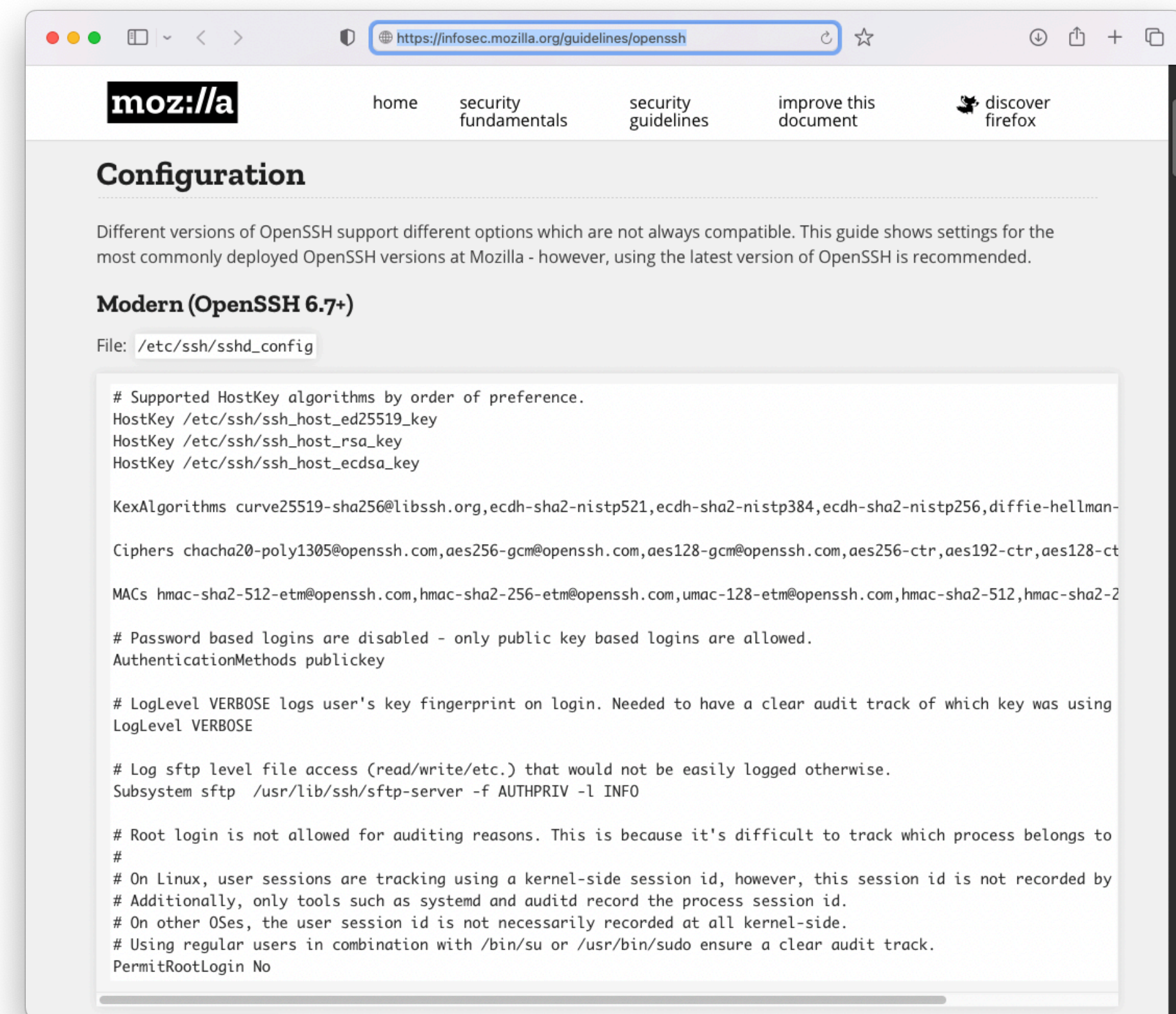
```
AuthenticationMethods publickey password
```

Testing and Hardning

- Test here:
<https://sshcheck.com/>



- Example config here:
<https://infosec.mozilla.org/guidelines/openssh>



/etc/ssh/sshd_config Include /etc/ssh/sshd_config.d/*.conf

Also...

AI Makes Stuff Up!

Green is OK

Red is made up!

Thanks Google's bard!

A

Do you know about openssh and Linux PAM? Can you explain how the environment variable SSH_AUTH_INFO_0 works with ssh and pam?

View other drafts

Yes, I do know about OpenSSH and Linux PAM. SSH_AUTH_INFO_0 is an environment variable that is used to pass authentication information from OpenSSH to PAM. This information can include the username, password, and public key fingerprint of the user who is attempting to authenticate.

When a user attempts to SSH into a server, OpenSSH will first authenticate the user using its own built-in authentication methods. If the user is authenticated successfully, OpenSSH will then set the SSH_AUTH_INFO_0 environment variable with the authentication information.

PAM is a framework for authenticating users on Unix-like operating systems. It allows administrators to configure different authentication methods for different users and groups. When a user attempts to log in to a system that uses PAM, the system will first check the SSH_AUTH_INFO_0 environment variable to see if it contains any authentication information. If it does, PAM will use this information to authenticate the user.

The format of the SSH_AUTH_INFO_0 environment variable is a JSON string. The following is an example of the format:

```
JSON

{
  "username": "user",
  "password": "password",
  "public_key_fingerprint": "ssh-rsa AAAA..."
}
```

Use code with caution. [Learn more](#)

If the PAM module is able to successfully authenticate the user using the information in the SSH_AUTH_INFO_0 environment variable, it will return a PAM_SUCCESS status. Otherwise, it will return an error status.

The SSH_AUTH_INFO_0 environment variable is a powerful tool that can be used to integrate OpenSSH with PAM. It allows administrators to use PAM to authenticate users who are logging in via SSH. This can help to improve the security of a system by providing an additional layer of authentication.