Using a TPM2 HMAC as a shadow password hash

David Safford

7/24/2023

Background:

Linux keeps a shadow password file with

{salt, HASH(salt | password)}

At authentication time, the system takes the passphrase and salt, calculates the hash, and compares it to the stored hash, accepting the authentication if they match. If an attacker can take a copy of the shadow file to another system, they can often reverse the hash in offline attacks, such as rainbow tables, dictionary/cracking attacks, and even brute force VLSI. Offline attacks are estimated to be able to reverse up to 30% of existing password hashes.

This has led to an arms race, with increasingly complex hashes to try to frustrate the offline attacks. The current default on Fedora is “yescrypt”, which attempts to make offline attacks harder, by using a large table in the hash – an attacker has to copy the table, and the table makes dedicated hardware more expensive.

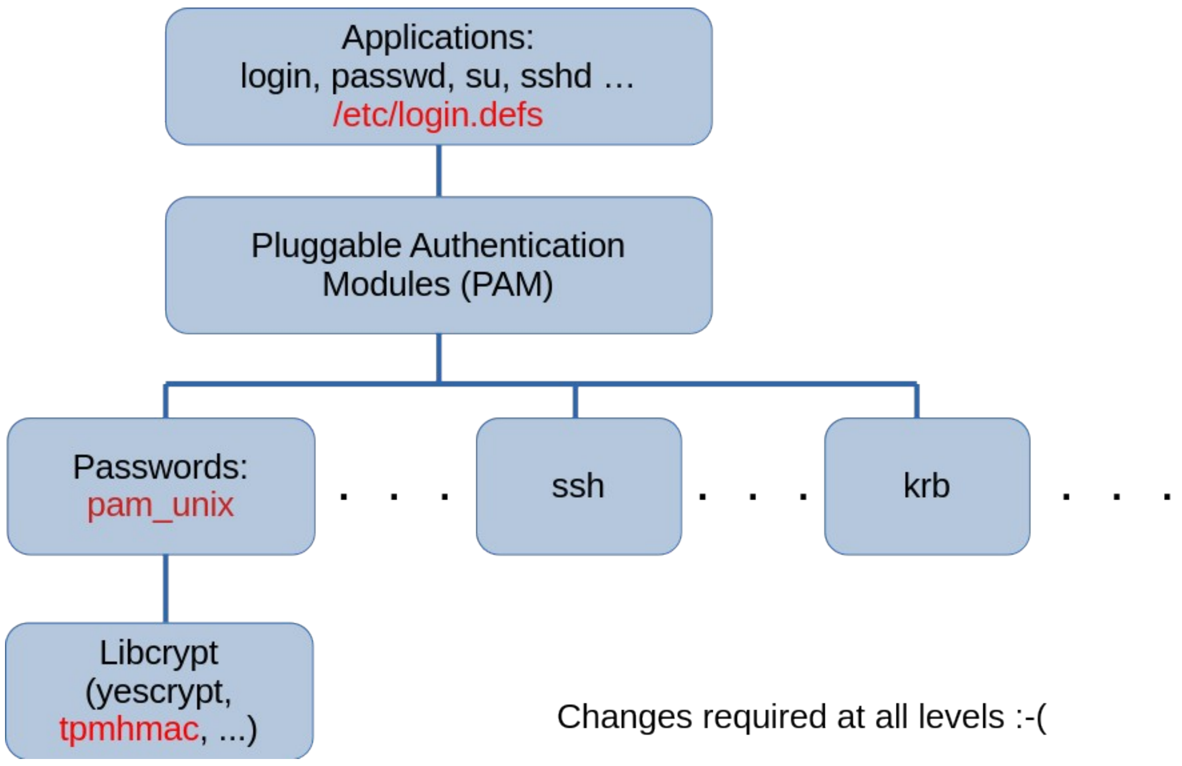
A better approach is to use a TPM, a hardware device built into most PC class machines. The TPM can create an HMAC key inside the chip, and use this key to HMAC the salt and passphrase. Since no one, not even the machine’s owner can get the HMAC key in plaintext, no one can mount offline attacks to try to reverse the hashes. An online attacker could use the TPM as an oracle to mount attacks, but this would inherently be online, detectable, and very slow.

Proposal:

In this proposal, Linux keeps a shadow password file with

{salt, TPM2_HMAC(hmac_key, (salt | password))}

This is simple in concept, but a bit tricky to implement, as Linux authentication is quite complex. Most Linux systems use PAM (Pluggable Authentication Modules) for authentication. This provides flexibility to support passwords, ssh keys, kerberos, tokens and other authentication methods, and to provide this service to all applications needing authentication. The following diagram gives a simplified view of the overall system.


Applications:
login, passwd, su, sshd …
/etc/login.defs
Pluggable Authentication Modules (PAM)
Passwords:
pam_unix
. . .
ssh
. . .
krb
. . .
Libcrypt (yescrypt, tpmhmac, ...)
Changes required at all levels :-(

The top level has applications, including login, passwd, and su, that want to authenticate a user. PAM then provides a common framework. For password authentication, the pam_unix module is used. It in turn uses a password hashing library call libcrypt. (Fedora currently uses the libxcrypt package, but still names the library libcrypt.)

Adding a new hash method requires adding support for the hash to libcrypt, modifying pam_unix to know of the availability of the new hash, and modifying /etc/login.defs to select the new hash.

Some details on the shadow password and libcrypt formats:

A shadow password entry is a line starting with a user name followed with colon delimited fields:

```
    username:hash:last:min:max:warn:inactive:expire:
          last – days since epoch that password was last changed
          min – minimum days between password changes
          max – maximum days between password changes
```

```
        warn – number of days of warning before expiration
        inactive – days after expiration before account disabled
        expire – days since epoch for account expiration
```

The hash field has subfields delimited by ‘$’

```
   $ type $ parameters $ salt $ hash
        The type field is typically a single character:
           ‘1’  MD5
           ‘5’  sha-256
           ‘6’  sha-512
           ‘y’  yescrypt
           ‘t’  tpmhmac (new)

        The parameters are specific to each hash.
        For tpmhmac, they are:
            parent handle ‘$’ base key file path
        e.g.
            0x81000004$/etc/hmac.
        Here, the key files would be /etc/hmac.pub
        and /etc/hmac.priv

        The salt and hash fields depend on the hash.
```

For tpmhmac, they are 16 bytes encoded as 22 b64 characters and 32 bytes encoded as 43 b64 characters.

Installation:

1. Apply the supplied patches to libxcrypt and pam source, compile and install. (On fedora I used the respective source rpms, adding the patches to rpmbuild/SOURCES, and spec files.)

2. copy tpmhmac.conf to /etc

3. Provision your TPM as with an SRK and optionally a DRSK as described in the TPM_KEYS repository.

4. create the HMAC keys. For example, with a DRSK at 0x81000004
   tpm2_create -C 0x81000004 -G hmac -a "sensitivedataorigin|userwithauth|sign" -u hmac.pub -r hmac.priv
   and place them in /etc. (The parent key handle and hmac key filenames can be tailored in /etc/tpmhmac.conf.)

5. in /etc/login.defs change ENCRYPT_METHOD to ENCRYPT_METHOD TPMHMAC

6. update the selinux policy to allow passwd, unix_chkpwd, and su to access the TPM. The easiest way to do this is run each command, and when they fail, check journalctl, Which will have detailed instructions on how to add the needed rules to the current policy. This need only be done once for each command.This will look something like:

   ```
   ausearch -c 'passwd' --raw | audit2allow -M my-passwd
   semodule -X 300 -i my-passwd.pp
   ```

All new password hashes will now use the TPM. Existing hashes continue to use their existing hash even when updated with passwd. To force en existing hash to change to tpmhmac, you have to delete the hash for that account in /etc/shadow (everything between the first and second ':'.)