README.md

64-bit PGP Key ID clash

When downloading software you can optionally verify that the file is authentic.

This can be done with PGP.

This attack has been performed with 32-bit keys [1] but I think I can take it a little further and show, that even 64-bit keys are not sufficient.

I'll use Ubuntu as an example because of their use of 64-bit PGP keys in there documentation.

Expected way

Download software

```
wget http://releases.ubuntu.com/bionic/ubuntu-18.04.2-desktop-amd64.iso
```

Download signatures

```
wget http://releases.ubuntu.com/bionic/SHA256sums.gpg
```

Recv keys from the Ubuntu server

```
gpg --keyid-format long --keyserver hkp://keyserver.ubuntu.com --recv-keys 0x46181433FBB75451
```

Which then will produce:

```
gpg: key 46181433FBB75451: 102 signatures not checked due to missing keys
gpg: key 46181433FBB75451: public key "Ubuntu CD Image Automatic Signing Key <cdimage@ubuntu.com>" imported
gpg: no ultimately trusted keys found
gpg: Total number processed: 1
gpg:               imported: 1
```

The software is then downloaded and can be verified using

```
gpg --verify SHA256sums.gpg ubuntu-18.04.2-desktop-amd64.iso
```

Will produce:

```
gpg: Signature made Fri 25 Mar 04:36:20 2016 GMT
                    using DSA key ID 46181433FBB75451
gpg: Good signature from "Ubuntu CD Image Automatic Signing Key <cdimage@ubuntu.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: C598 6B4F 1257 FFA8 6632  CBA7 4618 1433 FBB7 5451
```

All well and good!

Malicious way

Download software (Software replaced with malicious payload)

```
wget http://releases.ubuntu.com/bionic/ubuntu-evil.iso
```

Download signatures (Also spoofed)

```
wget http://releases.ubuntu.com/bionic/SHA256_evil_sums.gpg
```

Now, this is where the main attack occurs. The evil sums have been signed with a key that has the *same* 64-bit ID as the key defined in the documentation.

So when the command is run below it will load the legitimate key *alongside the evil key*

```
gpg --keyid-format long --keyserver hkp://keyserver.ubuntu.com --recv-keys 0x46181433FBB75451
```

Will produce something like:

```
gpg: key 46181433FBB75451: 102 signatures not checked due to missing keys
gpg: key 46181433FBB75451: public key "Ubuntu CD Image Automatic Signing Key <cdimage@ubuntu.com>" imported
gpg: key 46181433FBB75451: public key "Ubuntu CD Image Automatic Signing Key <cdimage@ubuntu.com>" imported
gpg: no ultimately trusted keys found
gpg: Total number processed: 2
gpg:               imported: 2
```

Notice the extract identical line. The key comment has also been spoofed making look exactly the same as the legitimate key.

```
gpg --verify SHA256_evil_sums.gpg ubuntu-evil.iso
```

Will produce an almost identical result to the real example. Please note: `x` s have been used to show the bytes of the key we're not controlling. This is the only section of the attack that makes it visible things are not the way they're supposed to. *This can be easily missed*.

```
gpg: Signature made Fri 25 Mar 04:36:20 2016 GMT
                    using DSA key ID 46181433FBB75451
gpg: Good signature from "Ubuntu CD Image Automatic Signing Key <cdimage@ubuntu.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: XXXX XXXX XXXX XXXX XXXX XXXX 4618 1433 FBB7 5451
```

Further work

This attack can then be taken on and used with real users, I could gather stats on how many users would be affected etc

References

[1] https://evil32.com/examples.html