

PGP Key Verification

Version 1.1, 08/26/2002

Stephen Gill
E-mail: gillsr@cymru.com
Published: 08/26/2002

Contents

| | |
|--------------------------|---|
| Introduction | 2 |
| Instructions | 2 |
| Step 1: Retrieve..... | 3 |
| Step 2: Fingerprint..... | 5 |
| Step 3: Encrypt..... | 6 |
| Step 4: Trust..... | 7 |
| Conclusion..... | 8 |
| References | 8 |

Credits

- Rob Thomas [robt@cymru.com] – For many clueful suggestions for improvements to this document.

Introduction

It is quite unfortunate that verification of imported PGP keys is often overlooked and avoided. Perhaps individuals place an undue amount of trust in public key servers and e-mails, or perhaps they simply place a higher priority on other day to day tasks.

How many times have you received a PGP key from a public key server or through e-mail and not verified its authenticity? Truthfully this author has done just so far too often. The fact that a key is stored on a public key server does not mean it should be implicitly trusted. Anyone can generate a key that looks like someone else's, but just because the self-signature checks out doesn't mean it should be trusted.

Likewise, receiving an e-mail with someone's public key does not mean it should be inherently trusted. There is no verification process for submitting a key to a public server, or to send an e-mail containing a public key to a user. Thus, the onus for authenticity verification lies solely with the user, NOT in the system or software itself.

Far too often we rely on security software as if it should be naturally trusted. We forget that security software is just as fallible as any other piece of code, especially when it is not used as it was designed.

Instructions

It would not be very difficult for someone to submit a falsified public key with the same credentials of another individual to confuse others and potentially gain access to otherwise private information. If sensitive data is encrypted with a fake key to begin with, all it takes is for the hijacker to

obtain access to the encrypted text assuming they were the ones who created the falsified key!

To that end, we delineate a very simple set of instructions to assist in the process of PGP key verification. Though the examples in this document focus specifically on MIT's PGP Freeware 6.5.8 [1] and Gnu's Privacy Guard (GnuPG) 1.0.7 [2], the principles of key verification apply to all PGP platforms.

Step 1: Retrieve

When retrieving new encryption keys through public key servers, the first step is to search for the e-mail address of the user in question. In this example, we are seeking the public key of the user with an e-mail address of johndoe@nowhere.com. Note that this address has been created for the purpose of this example only and does not actually exist. The PGP command to search for the key on a public key server is as follows:

```
# pgp -kvv johndoe@nowhere.com ldap://certserver.pgp.com
Pretty Good Privacy(tm) Version 6.5.8
Internal development version only - not for general release.
(c) 1999 Network Associates Inc.

Export of this software may be restricted by the U.S. government.

Key server: 'ldap://certserver.pgp.com', looking for user ID
"johndoe@nowhere.com".
Type bits      keyID      Date      User ID
DSS 1024      0xF3FAA202 2002/08/24
  DH 2048      0xF3FAA202 2002/08/24 John Doe <johndoe@nowhere.com>
sig           0xF3FAA202                (Unknown signator, can't be
checked)
DSS 1024      0x386BCE30 2002/08/25
  DH 1024      0x386BCE30 2002/08/25 John Doe <johndoe@nowhere.com>
sig           0x386BCE30                (Unknown signator, can't be
checked)
2 matching keys found.
```

Our search retrieved two keys belonging to the user in question. How do we know which is the correct one? In order to select the right key, an additional piece of information is required to further limit the selection, such as the actual Key ID from the user. Presumably the keyID should be obtained from the user to distinguish between the multiple keys. Once the proper keyID is known, the user's public key can be retrieved using the keyID instead of the e-mail address. The following command retrieves the appropriate key and stores it in a plaintext file named johndoe.pgp.

```
# pgp -kx 0x386BCE30 johndoe.pgp ldap://certserver.pgp.com
Pretty Good Privacy(tm) Version 6.5.8
Internal development version only - not for general release.
(c) 1999 Network Associates Inc.

Export of this software may be restricted by the U.S. government.

Key extracted to file 'johndoe.pgp'.
```

If we would have specified the User ID instead of the Key ID, all matching keys would have been downloaded instead of just one. Finally, the key is added to the local keyring as follows:

```
# gpg -ka johndoe.gpg
Pretty Good Privacy(tm) Version 6.5.8
Internal development version only - not for general release.
(c) 1999 Network Associates Inc.

Export of this software may be restricted by the U.S. government.

Looking for new keys...
DSS 1024/1024 0x386BCE30 2002/08/25 John Doe <johndoe@nowhere.com>
sig?          0x386BCE30          (Unknown signator, can't be
checked)

keyfile contains 1 new keys. Add these keys to keyring ? (Y/n) y

New userid: "John Doe <johndoe@nowhere.com>".
New signature from keyID 0x386BCE30 on userid John Doe
<johndoe@nowhere.com>
Keyfile contains:
  1 new key(s)
  1 new signatures(s)
  1 new user ID(s)

Summary of changes :

New userid: "John Doe <johndoe@nowhere.com>".
New signature from keyID 0x386BCE30 on userid John Doe
<johndoe@nowhere.com>

Added :
  1 new key(s)
  1 new signatures(s)
  1 new user ID(s)
```

The three PGP commands listed above can all be accomplished using one interactive request in GnuPG as follows:

```
# gpg --keyserver certserver.pgp.com --search-keys johndoe@nowhere.com
gpg: searching for "johndoe@nowhere.com" from HKP server
certserver.pgp.com
Keys 1-2 of 2 for "johndoe@nowhere.com"
(1)   John Doe <johndoe@nowhere.com> 2048
      created 2002-08-24, key F3FAA202
(2)   John Doe <johndoe@nowhere.com> 1024
      created 2002-08-25, key 386BCE30
Enter number(s), N)ext, or Q)uit > 2
gpg: requesting key 386BCE30 from HKP keyserver certserver.pgp.com
gpg: key 386BCE30: public key imported
gpg: Total number processed: 1
gpg:          imported: 1
```

If a public key is received via e-mail or downloaded from the web, then the retrieval process has essentially already been performed. GPG keys can be imported using the “gpg –import [files]” command. However, before using the key to encrypt any sensitive information it should be verified through an independent fingerprinting process.

Step 2: Fingerprint

Key fingerprint verification is actually much easier than it sounds. In a nutshell, both sides should compare the fingerprints of the public key in question. If the fingerprints match, then they can proceed to send a test message. If they do not, then the key can NOT be trusted.

The process of comparing fingerprints should be performed through a mechanism whereby the identity of the owner of the public key in question is known or trusted. This may come in the form of a phone call to a trusted phone number, a face-to-face conversation, photo ID verification, or all of the above. Obviously, the goal is to ensure that the person you are communicating with is who you think he or she is. If you know the key's owner and recognize their voice, it is easy enough to call them and verify the key's fingerprint over the telephone.

It is best to use a different communication method than the one that was used to send the key itself. A good combination is to send the key via e-mail, and the key fingerprint via a telephone conversation or received in person on a business card.

The command to display the fingerprint of the key in question is as follows:

```
# gpg -kvc johndoe@nowhere.com
Pretty Good Privacy(tm) Version 6.5.8
Internal development version only - not for general release.
(c) 1999 Network Associates Inc.

Export of this software may be restricted by the U.S. government.

Looking for user ID "johndoe@nowhere.com".
Type bits      keyID      Date      User ID
DSS 1024/1024 0x386BCE30 2002/08/25 expires 2002/09/24
                                John Doe <johndoe@nowhere.com>
      Key fingerprint = AF CE 9D 00 8C 22 08 73 C1 3B A1 73 3D 14
28 9F 38 6B CE 30
1 matching key found.
```

Both you and the key's owner should compare fingerprints to make sure they match. The GPG command to display a key fingerprint is as follows:

```
# gpg --fingerprint johndoe@nowhere.com
pub 1024D/91D5EF72 2002-08-25 John Doe <johndoe@nowhere.com>
   Key fingerprint = AFCE 9D00 8C22 0873 C13B A173 3D14 289F 386B CE30
sub 2048g/64DE55F1 2002-08-25
```

Using the fingerprinting procedure, you can verify and sign each other's keys, conveniently building up a safe network of trust. The "web of trust" model is another form of identity confirmation that does not require the fingerprinting method for keys already signed by trusted third parties. If the key in question has been signed (vouched for) by the mutually trusted key of a 3rd party, the key can be considered trustworthy.

Step 3: Encrypt

As the third and final safety measure, a test message should be sent to the user with the key in question requesting a response that includes the decrypted message. This ensures that the user is able to decrypt the messages that have been encrypted with their public key.

The following PGP command encrypts and signs a secret test message for user johndoe@nowhere.com:

```
# echo "<secret word>" > test && pgp -eas test johndoe@nowhere.com
Pretty Good Privacy(tm) Version 6.5.8
Internal development version only - not for general release.
(c) 1999 Network Associates Inc.

Export of this software may be restricted by the U.S. government.

A secret key is required to make a signature.

Recipients' public key(s) will be used to encrypt.

You need a pass phrase to unlock your secret key.
Key for user ID "nobody@pgp.com"

Enter pass phrase:

Passphrase is good

Key for user ID: John Doe <johndoe@nowhere.com>
1024-bit DSS key, Key ID 0x386BCE30, created 2002/08/25, expires
2002/09/24
WARNING: Because this public key is not certified with a trusted
signature, it is not known with high confidence that this public key
actually belongs to: "John Doe <johndoe@nowhere.com>".

Are you sure you want to use this public key (y/N)?y

Transport armor file: test.asc
# cat test.asc
-----BEGIN PGP MESSAGE-----
Version: PGP 6.5.8

qANQR1DBwE4DyH8GWHzUexgQA/973Daa1iA0pgtFOvCzNurDL4KJWrxo1tQwmLtR
UEVPMBQjQ+edwC1nsFwJMHocrYEBctqU6RZUgTyfv0Ckawh09sPRE2YcZsZKyMe5
jHRdL5dKlHp+iMFEl9q8/VM2xjgrMlbF4sRdnIGaiTJaE8kmeq67XhXcUQP1ZxaA
XnH3PwP/blNlQw5tqLPfccKzS4gDC4Oa24fj0MXqdGzFEHo+w3YPst3PUQyZXaAh
KjhyciMpqoUVfIH98Q/bP2NjAsGVqu0yRPn/Q7ZCQjY7eYF0oqfZYm6kJE8hJbS
59t0CH+r0FCiTSkdqaVK51uvJVazbmun5Y50t5Cgg9z4VWmnXifJiBbcwdHRon6d
kN0UImmP6JF5iD9+k7oNR1NNkiVbKUPObQgntC9N5m3tYMzHQ0gTkQYy1SxoCXHt
fzuOPwDkWzPTVTGliuAg+82pe7kKbkhKfGA1Qh2vMsL4kEf0f1Z6R4JL7MHSF3J+
Cle4YQX0vqqaIpc12c8BFY+0fx1l4LcMU5p4aq1N92Y=
=qdKt
-----END PGP MESSAGE-----
```

Since we have not yet marked this key as “trusted”, PGP prompts us for verification before encrypting our message. The resulting text should be mailed to the user. Once a proper response is received we can be reasonably sure that this is a valid key and we can begin to use it for further encryption to that user.

Using GnuPG the same procedure is as follows:

```
# echo "<secret word>" | gpg -ear johndoe@nowhere.com
gpg: 386BCE30: There is no indication that this key really belongs to the
owner 2048g/386BCE30 2002-08-25 "John Doe <johndoe@nowhere.com>"
Fingerprint: AFCE 9D00 8C22 0873 C13B A173 3D14 289F 386B
CE30
```

```
It is NOT certain that the key belongs to its owner.
If you *really* know what you are doing, you may answer
the next question with yes
```

```
Use this key anyway? y
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.0.7 (FreeBSD)
```

```
hQIOA4CYqe9k3lXxEAgA1xzrQA7ujIqIjMf4TjYPWd2R1pzGeDpasQfxn8MKqJtF
e7bEp6stCPPNkqmU+e807oTGhiGfzzWAIb1jkEKBb0hR8P2fZlOk5NCh9PmSA1Rm
5rsJ2KAN9+6B0jOExhFUJOxndR7OFkPICTK3j5snKznpf0fRDQXRFSNdGdum5ER+
JLHv3ie8CugEijG98qaeR8IF9CbyUnRcTHb0JGqHp2BUK3RY+TJHjVAZ71358T52
4rTT1iCokgrMMbBWtcGVwzTr9DdedAizY+eHyEKKv7hbpEuWNT0bLU/f3EwhXjY
oVBKdEUu/NGMNfo88yshTd4yX83VziNgM8y1zAWhjQgA5pfwih/9VRnhzXrFgtmx
kL1QNzTeV2Vxc+ec3h3mGDeiq/eQzkm9s50ZyhhDZ7RxOae6YFUz2/+sytdPwqo+
OHXW61WfaWWu+FH8bF1eXO1TA9+zvIHlulCBF5FfMVoj2jbrWxxmK6bzT2MVUx4Q
y0J9735+8b6qxjT4A5cee8ODsJLHX9nENICJz/0u1w66jxj1aDpqqf+q3vP5z9aV6
EgT3TmXvgc+AIYmXs4p+pgtUOHl21Muko4T2CogZuKINLClqBNeg07/TzqWcUw2
RL04zagH+Evc+K7OUeZ5FM2N5qi5B1iHok1z5+2bJ83RoB46gW54VtbqUSE2wEQ0
/8kqKIIVOMy6M2axb4g9a2HSzPwol0Iw7Z9vHnRQ1UW2Axc+q+lkh6jY/9pH
=RR/U
-----END PGP MESSAGE-----
```

Step 4: Trust

Only once the identity of a user and key has been fully verified should a key be marked as marginally trusted or trusted. Below, we mark the key in question as marginally trusted because steps 1-3 have been followed. If we believe the person in question will follow the proper procedures to fully verify other's identities before signing keys, then we can set it to fully trusted. If there is any doubt as to whether they will not follow the same steps that you did for key verification, then it is best to mark the key as marginally trusted only.

```
# gpg -ke johndoe@nowhere.com
Pretty Good Privacy(tm) Version 6.5.8
Internal development version only - not for general release.
(c) 1999 Network Associates Inc.
```

```
Export of this software may be restricted by the U.S. government.
```

```
Editing userid "johndoe@nowhere.com" in default key ring
```

```
Key for user ID: John Doe <johndoe@nowhere.com>
1024-bit DSS key, Key ID 0x386BCE30, created 2002/08/25, expires
2002/09/24
No secret key available. Editing public key trust parameter.
This key/userID association is not certified.
```

```
Current trust for this key's owner is: untrusted
```

```
Make a determination in your own mind whether this key actually
belongs to the person whom you think it belongs to, based on available
evidence. If you think it does, then based on your estimate of
that person's integrity and competence in key management, answer
the following question:
```

Would you trust "John Doe <john.doe@nowhere.com>"
to act as an introducer and certify other people's public keys to you?
(1=I don't know (default). 2=No. 3=Usually. 4=Yes, always.) ? 3

Similarly, the GPG syntax is as follows:

```
# gpg --edit-key johndoe@nowhere.com trust
gpg (GnuPG) 1.0.7; Copyright (C) 2002 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

gpg: checking the trustdb
gpg: checking at depth 0 signed=0 ot(-/q/n/m/f/u)=0/0/0/0/0/1
gpg: next trustdb check due at 2002-09-24
pub 1024D/386BCE30  created: 2002-08-25 expires: 2002-09-24 trust: u/u
sub 1024g/7CD47B18  created: 2002-08-25 expires: 2002-09-24
(1). John Doe <johndoe@nowhere.com>

pub 1024D/386BCE30  created: 2002-08-25 expires: 2002-09-24 trust: u/u
sub 1024g/7CD47B18  created: 2002-08-25 expires: 2002-09-24
(1). John Doe <johndoe@nowhere.com>
```

Please decide how far you trust this user to correctly
verify other users' keys (by looking at passports,
checking fingerprints from different sources...)?

```
1 = Don't know
2 = I do NOT trust
3 = I trust marginally
4 = I trust fully
5 = I trust ultimately
i = please show me more information
m = back to the main menu
```

Your decision? 3

```
pub 1024D/386BCE30  created: 2002-08-25 expires: 2002-09-24 trust: f/u
sub 1024g/7CD47B18  created: 2002-08-25 expires: 2002-09-24
(1). John Doe <johndoe@nowhere.com>
```

Command> q

If you wish to vouch for the key's authenticity you can do so by signing it with your own private key and mailing it back to its owner. In this way, others that have established a trust relationship with your public key can also trust the authenticity of the key you have signed.

Conclusion

PGP key verification is often bypassed at the cost of increased security exposure. In this paper we have reviewed three steps necessary to perform PGP key verification, including: retrieval, fingerprinting, and encryption. Following these steps allows us to place a greater trust in the keys we use to securely communicate with others.

References

[1] MIT's PGP Freeware
<http://web.mit.edu/network/pgp.html>

[2] GNU's Privacy Guard (GnuPG)
<http://www.gnupg.org>