OPSEC PGP Best Pratices

Author: Insider

New to PGP? Reading Material: http://greysec.net/showthread.php?tid=16
Source: https://gist.github.com/grugq/03167bed45e774551155

This is a guide on how to email securely. There are many guides on how to install and use PGP to encrypt email. This is not one of them. This is a guide on secure communication using email with PGP encryption. If you are not familiar with PGP, please read another guide first. If you are comfortable using PGP to encrypt and decrypt emails, this guide will raise your security to the next level.

PGP is used to protect email content, but there is so much more to secure email than simply encrypting content. This guide will attempt to lay out operational guidelines for secure email practices. Following these rules will enable you to use PGP to maximum effect. Protecting the content of an email conversation is just the start of secure email.

It is also critical to secure the operational procedures surrounding email use. This includes things like how you write your emails, how you store them, how you store your keys and use them, and so on. Everything related to the process of sending and receiving email must be secured. Operationally secure email is not much more difficult than normal email, it just requires a little more attention to detail, some changes to default settings, and consistent discipline.

METADATA

Metadata provides the context to the content of the email. Protecting the email content *(with PGP)* will significantly enhance the security of the communication. Frequently, the context *(the metadata)* is sufficient to learn a great deal about the communication even without the content. Unfortunately, even PGP encrypted email leaves *comms* metadata exposed, this includes:

- Subject
- To
- From
- Dates and Times
- IP Addresses
- Email Application

Minimising the contextual information leakage from the *comms* starts with knowing what metadata will be exposed. Where possible, and relevant, take control over that information and *unlink* it from data linked to you. For example, you can control the From field by creating a new email account. The IP address of the sending email client can be changed by using a **VPN**, **Tor**, or a **public Internet connection**.

One option to consider is the use of Torbirdy which integrates Tor directly into Thunderbird *(as well as sanitizes emails slightly)*. The usual caveats about Tor apply: do not rely exclusively on Tor, if you need to protect your IP address then use an IP address that is not attributable to you. The Subject must *never* be relevant. There is little else you can do about the; To, From, and IP as those are controlled by the infrastructure. However, you have complete control over the Subject. All acceptable subjects are listed below:

- ...
- xxx
- ;;;
- :::
- <blank/empty>

The Subject must never ever refer to the content of the email, even obliquely. For example, *"Subject: Your cocaine has shipped!"* is a total email security failure. **Subjects should be empty!** Disable the *Warn about empty subject* setting on your email client.

SAFER KEYS

Key loss is catastrophic. The primary problem with PGP is that there is *no Forward Secrecy*. Losing a key means that all content encrypted with that key is compromised. The are two ways of dealing with this:

I. Create a single *master holy grail* key and **guard it with your life**.
II. Create keys frequently and **destroy them as soon as they are no longer needed**.

If you are going to go with option **#1** then you should probably use an OpenGPG smart card. There are a number of vendors that sell them. Mitigate by having more keys for shorter times. A more secure mitigation against key loss is to generate new keys frequently, use them for specific operations, and then destroy them. For example, when traveling, create a new **Travel PGP Key** and use that until you are back home. That way if anyone compromises your travel laptop they only breach the compartment for the duration of your travel. The impact of the compromise is contained by the limitation on the utility of the PGP key. So -- **more keys, more often**.

Publishing? Not a great idea. If you are serious about the anonymity of your PGP protected communications, you need to avoid publishing your key to the *keyservers*. Exchange your key only with the person(s) that you will communicate with. This will mitigate against future attribution attempts.

You can **deniable exchange keys** by having an easily *locatable* and identifiable public key. Tell all correspondents to use your public key to contact you and include their public key in the encrypted body of the message. This mitigates against both Mallory and Eve attacking the key exchange step. Just ensure that your public is verifiable through multiple channels, for example: Twitter, Facebook, blog, public mailing lists, key servers, etc. The more channels that host your PGP key fingerprint, the harder it is for Eve to attack them all.

Have a public signal that will alert people that your key has been compromised. For example if you post a specific message on Twitter *(e.g. "Someone has a case of the Mondays!")* then your correspondents will be alerted that your key is no longer valid. They must *rekey* internally and disregard all future *comms* using the old key.

WRITING

There are fundamentally two options when composing an email: you can either write the email in a text editor, or in an email client. For security, that is, control over the process, a *text editor is safer*. For convenience most people will use their email client.

- **Safest:**
  - Write in a text editor.
  - Encrypt on the command line.
  - Only expose already-encrypted data to the email client.
- **Realistic:**
  - At least restrict the plain text to the local system you control.

Drafts can be a significant source of risk. The storage and handling of drafts must be approached with care. In particular, make sure that they are encrypted *(**never stored in plain text**)*, that they are stored locally *(where you can be sure of deleting them)* and that they are deleted after use. Make sure to configure your email client to store drafts locally and to encrypt them before writing them to disk.

- Do not store on server.
- Ensure they are encrypted.
- Ensure they are deleted after use.

Delete the content in a reply, only quote the relevant parts if necessary. Mitigate against the potential threat of any single email being compromised by limiting the information within any single email. The more information is stored in the mental context of the correspondents, the less useful the information in the email is to an adversary. Wherever possible minimize the amount of irrelevant contextual information within the email body. Keep it short, simple, and clean.

For attachments, PGP has all the capability of tar or zip. It is possible to include all the files you need to include as a pure PGP messages without having an attachment called *secret-leaked nsa-docs.tar.gz.gpg.asc*. The program to use is **gpg-zipand** it takes both *–tar=* command line options and *gpg* command line options. Use this to bundle your files and send them as an opaque encrypted blob.

PGP encryption stores metadata about the decryption keys in the encrypted data. This is a simple optimization to allow the recipient to rapidly determine whether the email can be decrypted by a private key they possess. This information also allows an attacker to determine who can read the email. If the email is intended to be truly anonymous, this metadata must be discarded. Fortunately, there is a *gpg* command line flag for this: *--throw-keys*. **Ensure that --throw-keys is added to the command line when encrypting data.** PGP/MIME and inline PGP -ear aren't the same. For more control and compatibility, use inline PGP.

SENDING

Ensure you encrypt by default. Accidentally sending an *unencrypted* email is a potentially fatal error. Configure your email client to always encrypt by default. If you want to send a plain text email, then deliberately set that option. **Signing an email provides guarantees that the content was written by someone that possess the private key.** That it is, it positively identifies at least one person involved in the email exchange. **Think carefully about whether you want to be positively identified as the author of the email. Set your email client to not sign messages by default.**

Store sent messages locally, then delete them. After the email has been sent and is no longer operationally useful, delete it. To make sure that you can do this, configure your email client to store your *Sent* messages locally. When in doubt, store locally. At least you will have some control over whether it is thoroughly deleted.

RECEIVING

- Delete emails after they are no longer necessary.
- INBOX ZERO.
- OUTBOX ZERO.
- DRAFTS ZERO.
- TRASH ZERO.
- Regularly delete all emails. Easiest way is to set the Trash to erase everything after 30 days, and then delete every email after it has been processed.

Riseup.net PGP CONFIGURATION GUIDE

There is a PGP configuration guide from riseup.net. If you want to incorporate their best practice recommendations without any of their terrible advice about using key servers etc, then simply install this gpg.conf into your ~/.gnupg.

Remind your correspondents to practice the same PGP hygiene!