

## Lab 11 - Privacy Technologies

### Overview

Privacy is a usually included in the larger security landscape, but it deals with aspects that concern people more that technologies and tries to answer a very tough question: “How to access/compute data without the owner know who you are?”. While, like everything, is a sword with two blades, it tries to allow people own their data in the digital world and to provide anonymity while browsing the Internet.

### Exercises

#### 00 [0p]. Users

Create the following users: **red**, **green** and **blue**. Make sure that you can ssh into the VM using this users. For example, copy the “.ssh/” directory from student to the newly added users and “chown” it accordingly.


```
sudo useradd -m -s /bin/bash red
sudo useradd -m -s /bin/bash green
sudo useradd -m -s /bin/bash blue
```

#### 01 [50p]. Pretty Good Privacy

Pretty Good Privacy (PGP) is an encryption standard that can be used to authenticate in a distributed manner. GNU Privacy Guard (GPG) is an open-source implementation of the PGP standards. In this exercise you are required to send one file encrypted from one user to the other.

For the next exercises, you will need to be logged in as users red/green/blue via ssh in order to generate the gpg key.



- Unfortunately, gpg doesn't work when the user is with su (tty permission problems, owned by student). If you want to do this, either use ssh, or tmux after logging in: it allocates a new TTY ;)
- Generate a private/public key using the gpg tool for each of the three users previously created. **Use <red|green|blue>@cs.pub.ro for the emails ;)**
- First, we are going to send **red**'s public key to **green**. Export it into an ASCII file format and import it into **green**'s account.

 After importing the key you should list it and double check that it was stored in the public ring. At this moment the key is not trusted yet, we will do this in a future step.


- You should see something similar (for red and green):

```
green@isc:~$ gpg --list-keys
/home/green/.gnupg/pubring.gpg
-----
pub   2048R/13C73580 2019-04-23
uid           green <green@cs.pub.ro>
sub   2048R/F1C1FF9A 2019-04-23


pub   2048R/860244A1 2019-04-23
uid           red-student <red@cs.pub.ro>
sub   2048R/E7626ADD 2019-04-23
```


 The description of fields is available  here.

- Create a text file with some contents and encrypt it. (echo “text” > secret\_file.txt)
- Send the encrypted file back to **red** and decrypt it.
- The next step is to create a trust channel between **blue** and **red** using **green** as a trusted party. To do so, **green** must firstly sign **red**'s key and export both his key and **red**'s to **blue**. Move the exported files into **blue**'s directory and import them. After the import was done, list the keys available to **blue**.


 The signing process typically involves manually verifying the fingerprint of the key

- Now, **blue** should mark **green**'s key as trusted (by signing it). After this, as the **red** user, create a file with an important message and sign it (do not encrypt it for this step). Transfer the file to **blue**, read the file and verify the signature.
- In the default setup mode, the last step should have given a warning stating that the key is not trusted while still being valid (“Good signature”). This is because GPG uses a more complex trusted model. As a last step, login as the **blue** user and change the trust level for **green**'s key to “I trust ultimately”. After this verify the previous file signature again.

 The web of trust allows a more elaborate algorithm to be used to validate a key. A more flexible algorithm can now be used: a key K is considered valid if it meets two conditions:

- it is signed by enough valid keys, meaning
  - you have signed it personally,
  - it has been signed by one fully trusted key, or
  - it has been signed by three marginally trusted keys; and
- the path of signed keys leading from K back to your own key is five steps or shorter.  ref

#### 02. [40p] TOR

The Tor (The Onion Routing) project is an implementation of the more generic “onion routing” idea that allows a user to gain network anonymity while surfing the Internet. The mechanism that allows for a private surfing is based on re-encryption and “randomly” routing of the packet at the level of each router within the network, allowing each router to only know the previous and the next router in the route (not the source/destination of the packet)  ref. Accessing the Tor network can be done either through a local proxy of via a Browser pre-configured with the proxy server.

- First, please install `tor` :

```
sudo apt update
sudo apt install tor
```
- Enable SOCKS proxy by editing /etc/torrc and uncommenting `SOCKSPort 9050 ;)`


 Tor only supports TCP traffic, some make sure your DNS queries are done over TCP.

- torsocks** is a tool that forces any opened program to use the Tor network for connectivity. Open a shell and find out your real IP address. Now, open a shell using **torsocks** and find out the IP address via the Tor network. Restart the **tor** service and discovery your newly allocated IP address.



```
dig TXT +tcp +short o-o.myaddr.l.google.com @ns1.google.com | awk -F'"' '{ print $2}'
```

- You are going to configure your local Firefox browser to use the Tor proxy on the VM. First, use ssh local port forwarding to make port 9050 available to your machine:

```
ssh -J <username>@fep.grid.pub.ro -L 9050:localhost:9050 student@<VM_IP>
```
- Next, change the **Firefox** Network Settings to use Socks5 proxy using the IP address and port from your VM. You can verify that your browser is using Tor by accessing the following  website.

#### 03. [10p] Feedback

Please take a minute to fill in the  feedback form for this lab.

Search

#### Lectures

- Lecture 01 - Introduction
- Lecture 02 - Cryptography
- Lecture 03 - Hardware Security
- Lecture 04 - Access Control
- Lecture 05 - Authentication and Key Establishment
- Lecture 06 - Application Security
- Lecture 07 - Operating System Security
- Lecture 08 - Network Security
- Lecture 09 - Web Security
- Lecture 10 - Privacy Preserving Technologies
- Lecture 11 - Forensics

#### Labs

- kernel
  - Lab 01 - Introduction
  - Lab 02 - Cryptography
  - Lab 03 - Authentication and Key Establishment Lab
  - Lab 03 - Hardware Security
  - Lab 04 - Access control
  - Lab 05 - Authentication in Linux
  - Lab 06 - Application Security
  - Lab 07 - Operating System Security
  - Lab 08 - Network Security
  - Lab 09 - Web Security
  - Lab 10 - Forensics
  - Lab 11 - Privacy Technologies
  - Lab 12 - Security and Machine Learning

#### Support

- Useful resources
- Virtual Machine

#### Table of Contents

- Lab 11 - Privacy Technologies
  - Overview
  - Exercises
    - 00 [0p]. Users
    - 01 [50p]. Pretty Good Privacy
    - 02. [40p] TOR
    - 03. [10p] Feedback