Sicurezza e privatezza lab

Environment setup

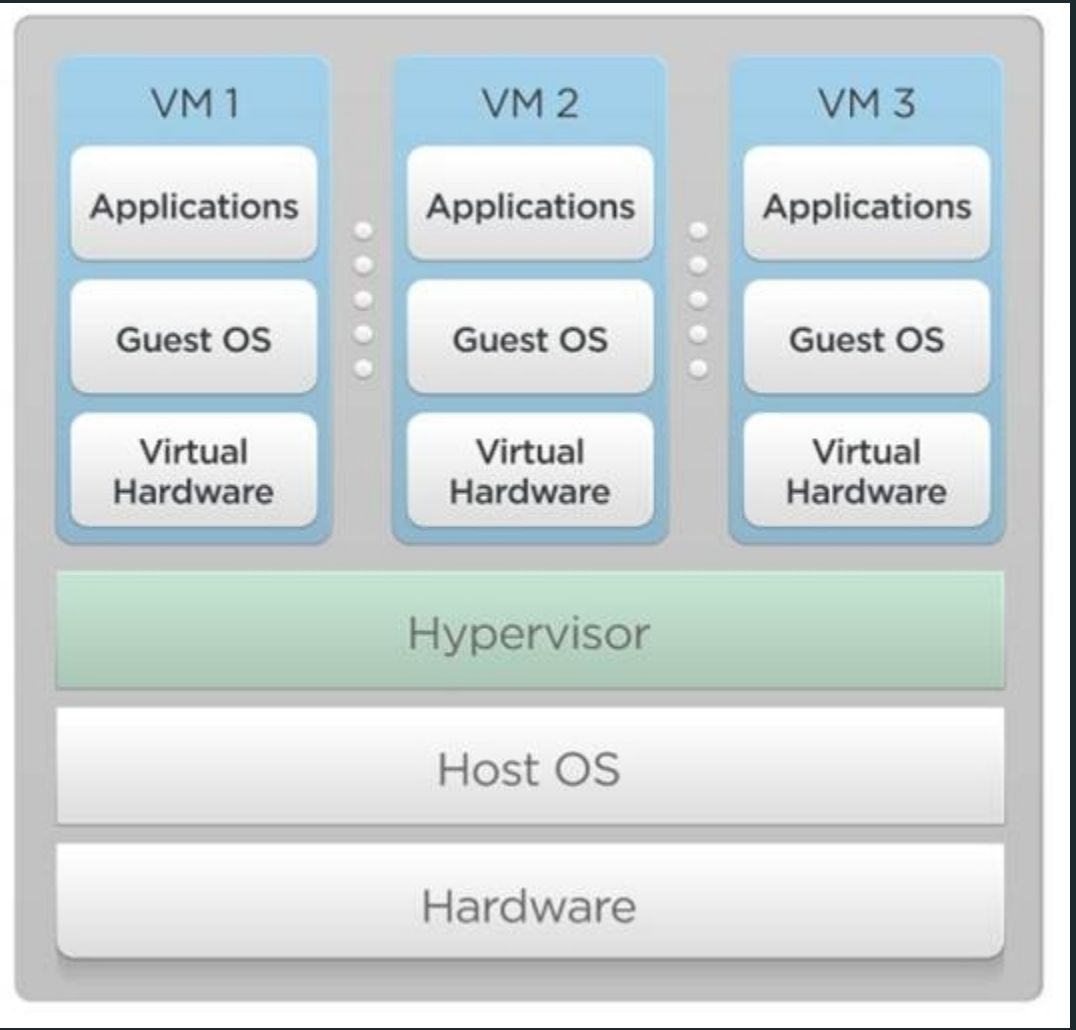
Dott. Matteo Zoia

SPLab@di.unimi.it

Virtual machine

Environment setup

A virtual machine (VM) is a virtual environment that functions as a virtual computer system with its own CPU, memory, network interface, and storage, created on a physical hardware system. Software called a hypervisor separates the machine's resources from the hardware and provisions them appropriately so they can be used by the VM.



Reference

Environment setup

Chapter 7, Virtualization and the cloud

Modern operating System (4 ed.), Andrew S. TANENBAUM, Herbert BOS

Tools and OS

Environment setup

Why virtual machine?

- Alignment on the same configuration
- Isolated Environment

Guest OS and distribution

- Distribuzione Linux, consigliata Kali
  - https://www.kali.org/docs/virtualization/install-virtualbox-guest-vm/
  - https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/
- Virtual Box
- (Metasploitable 2)

End

Sicurezza e privatezza lab

GPG and the web of trust

Dott. Matteo Zoia

SPLab@di.unimi.it

Against centralization

Web of trust

- X.509 standard using certificates and **certificate authorities** (CAs). It is best suited for structured organizational hierarchies with an **implicitly trusted authority** that vouches for all issued certificates. It’s the standard that is behind SSL/TLS and S/MIME email encryption.
- X.509 is highly centralized, so back in 1991 the first implementation of a new concept called **web of trust** start spreading. It was developed with a specific goal to not require centralized certification authorities, but instead rely on **trust relationships** between regular users. It was first implemented in the original PGP software and then become an open standard, known as OpenPGP.

You may already use GPG

Web of trust

If you've ever used Linux, you've most likely used OpenPGP without even realizing it. The open-source implementation of OpenPGP is called GnuPG (stands for GNU Privacy Guard), and nearly all distributions rely on GnuPG for package integrity verification.

Next time you run "apt install" or "apt update", each package will be verified against its cryptographic signature before it is allowed to be installed on your system.

This assures that the software has not been altered between the time it was cryptographically signed by distribution developers on the master server, and the time it was downloaded to your system.

Who to trust?

Web of trust

- The “trust” is the certainty that the public key we have for Alice actually belongs to Alice. If you and Alice live in the same house or neighborhood, establishing such trust is easy, you meet for coffee and exchange your public keys face-to-face.
- But what if you need to securely communicate with Chloe? She’s Alice’s good friend, but she lives on the other coast and the two of you have never met. Or what if Chloe invites Dharma and Ezri, who in turn invite Finn and Gabby? How can you trust a key of a person you’ve never met?

X.509

Web of trust

- This is where X.509 and OpenPGP diverge in their approach to solving this problem.
- X.509 establishes a system of trusted authorities. Say, everyone on the West coast must have their key verified (signed) by Alice before it is to be trusted, while everyone on the East coast must have their key signed by Chloe. Alice and Chloe cross-sign their own keys, so you end up with a trust hierarchy. As long as you trust Alice, you can securely communicate with everyone else.
- This is simple and straightforward, but it has one important downside. Evil Eve only needs to get access to Chloe’s private key in order to infiltrate your entire organization.

OpenPGP

Web of trust

- OpenPGP decided to choose a different approach — instead of having designated trust authorities (CAs) like Alice and Chloe, whom everyone must trust for the CA hierarchy to work, OpenPGP lets the user decide whom you trust, and how much. The resulting framework is called the “Web of Trust.”

Key generation

GPG

kali@kali: ~

File Actions Edit View Help

```
kali@kali:~$ gpg --full-generate-key
gpg (GnuPG) 2.2.20; Copyright (C) 2020 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
  (14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 2048
Requested keysize is 2048 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 2y
Key expires at Mon 10 Oct 2022 06:48:11 AM EDT
Is this correct? (y/N) y
```

```
gpg: key 0A434563AFE43E9E marked as ultimately trusted
gpg: directory '/home/kali/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/kali/.gnupg/openpgp-revocs.d/2DED60DCDA1D69006BDED4BE0A4
34563AFE43E9E.rev'
public and secret key created and signed.

pub   rsa2048 2020-10-10 [SC] [expires: 2022-10-10]
      2DED60DCDA1D69006BDED4BE0A434563AFE43E9E
uid                      Matteo Zoia (S&P Lab.) <SPLab@di.unimi.it>
sub   rsa2048 2020-10-10 [E] [expires: 2022-10-10]

kali@kali:~$
```

OpenPGP, validity

Web of trust

- It is important to understand the difference between **trust** and **validity**, as they are two different sides of the same coin. In GnuPG parlance, “validity” represents our certainty that they key actually belongs to Alice. In the below output, “validity: full” tells SPLab that he can be absolutely certain that it’s Alice’ key. But what is the meaning of “trust: unknown”?

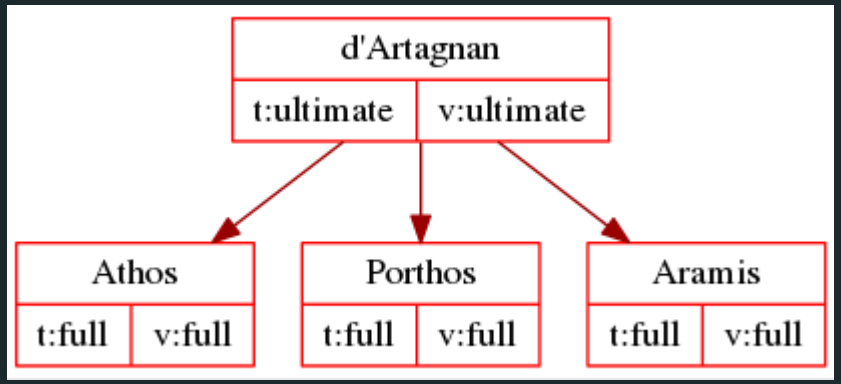
OpenPGP, trust

Web of trust

- In this case “trust” (also called “owner trust”) is how much SPLab trusts Alice to verify other people’s identities (by looking at passports, asking their mutual friends to vouch for them, etc). Let’s say SPLab fully trusts Alice to do a good job verifying people’s identities before he signs their keys. He edits Alice’ key and sets owner trust to “full”.

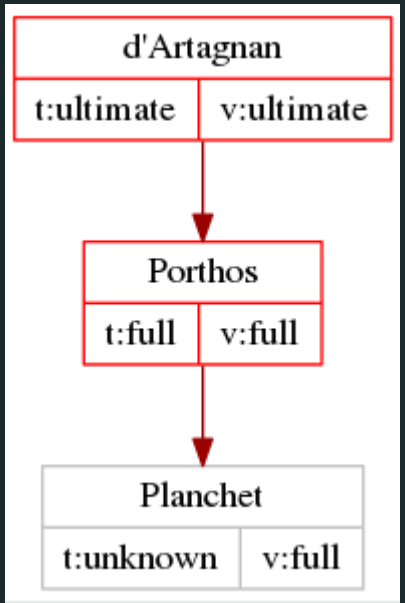
OpenPGP, example

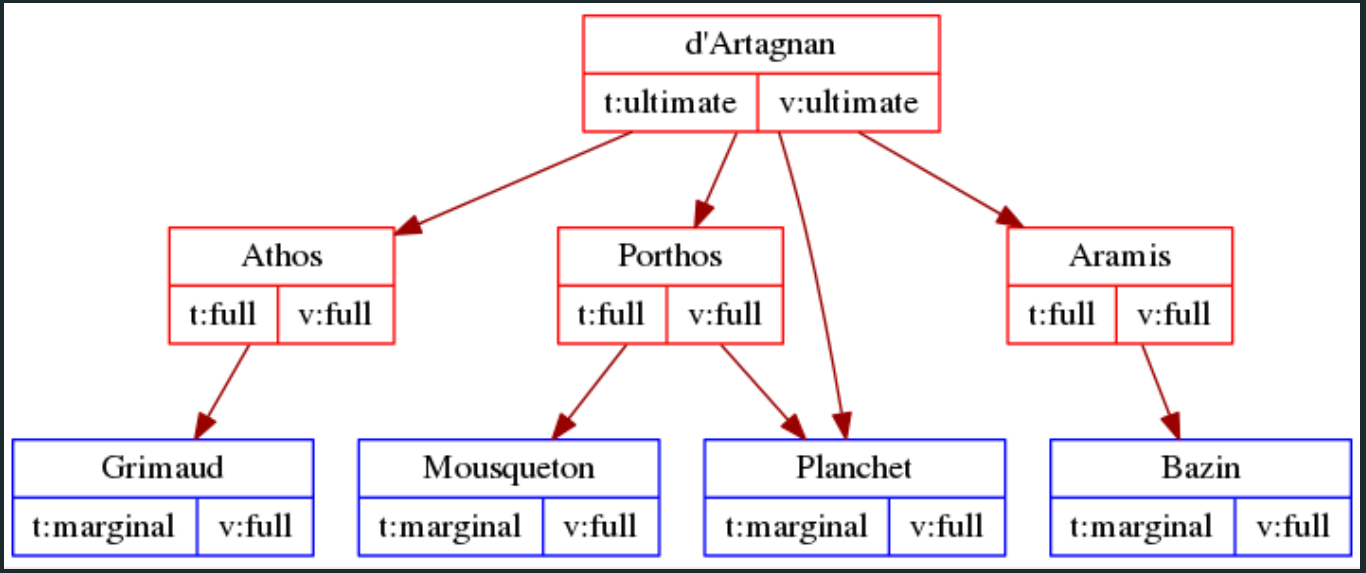
Web of trust



(1)

(2)





(3)

https://www.linux.com/training-tutorials/pgp-web-trust-core-concepts-behind-trusted-communication/

OpenPGP, example

Web of trust

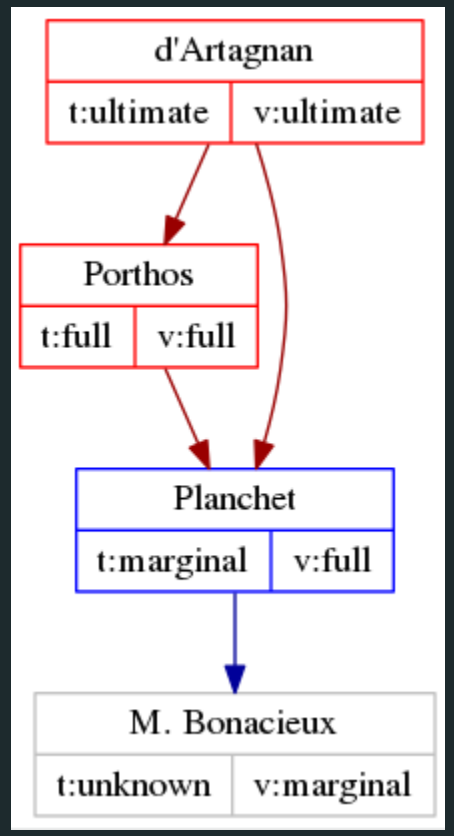
- You will notice that we again have “trust: unknown” for Chole — this is because we’ve not assigned owner trust to his key yet. As a general rule, **“validity” is something we calculate from the signatures on the person’s key**, while **“trust” is something we must assign to each key ourselves** in order for it to say something other than “unknown”.

OpenPGP, marginal trust

Web of trust

- What happens when we find a key signed by someone whom we only trust marginally?
- As you can see, the validity is marked as “marginal,” whish is gpg’s way of saying that there is only marginal assurance that this a people is actually who he says he is.

(4)

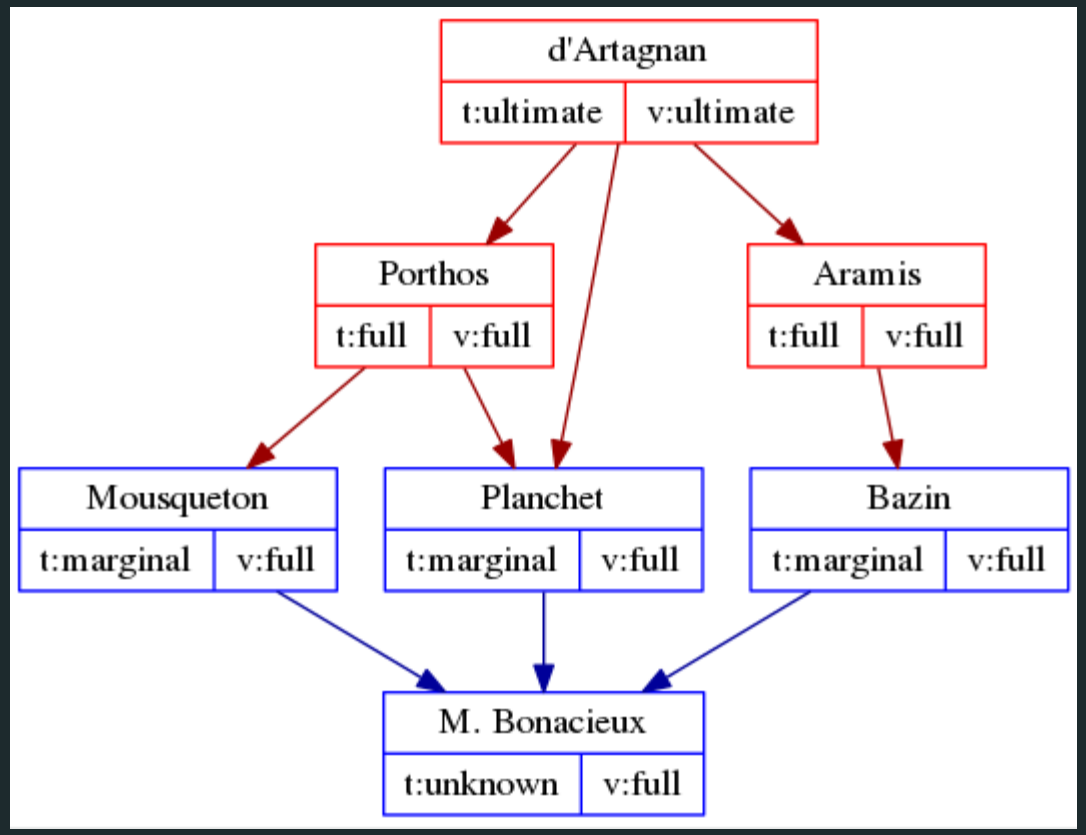


OpenPGP, marginal trust

Web of trust

- To strengthen the key validity, we need to have either someone else whom we fully trust to sign his key, or we need to find two other people with marginal trust, so there's a total of 3 marginal-trust signatures on the last key.

(5)



OpenPGP, keyserver

Web of trust

- Once you sign someone's key, how do you let others know about it? If your web of trust consists of only a handful of people, you can simply email the signed public key around, but this obviously is not very efficient as your organization grows. The PGP community recognized this problem early on and set up designated servers for distributing public keys. Many of them have web interfaces that allow searching and downloading public keys, but this functionality is also built-in into the GnuPG command-line tool.

```
$ gpg --keyserver pgp.mit.edu --search torvalds linux-foundation
$ gpg --keyserver pgp.mit.edu --send-key 329DD07E
```

End