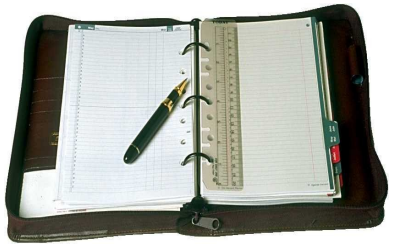
Security in Open Source and Linux

Giuseppe “Gippa” Paternò
Red Hat - Solution Architect & EMEA Security Prod. Expert
gpaterno@redhat.com

Who am I

- Currently Solution Architect and EMEA Security Expert in Red Hat
- Previously Security Solution Architect in Sun and also in IBM
- Red Hat Certified Security Specialist (RHCSS) and Cisco Certified Network Professinal (CCNP)
- Part of the italian security community *sikurezza.org*
- Published books and whitepapers
- Teached wireless security on “master sulla sicurezza informatica”, Politecnico di Milano
- Forensic analisys for local govs
- More on:
  - http://www.gpaterno.com/
  - http://www.linkedin.com/in/gpaterno

Agenda



- The “Defense in Depth” philosophy
- Advantages of OSS on Security
- Statistics about Linux vulnerabilities
- Misleading messages
- Security of the Operating System
  - Hardening, minimization and how Linux is positioned
  - Kernel security and other features
  - The importance of patching
  - MAC systems: SeLinux
- Authentication and authorization
  - Smart Cards and Biometric authentication
  - Intergration with Linux and with applications
  - Authorization and profiling, i.e. assigning roles
- Application protection
  - Be aware of the “default” installation
  - Use of the application-layer firewalls

“ The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards - and even then I have my doubts. ”

Eugene H. Spafford, director of the Purdue Center for Education and Research in Information Assurance and Security.

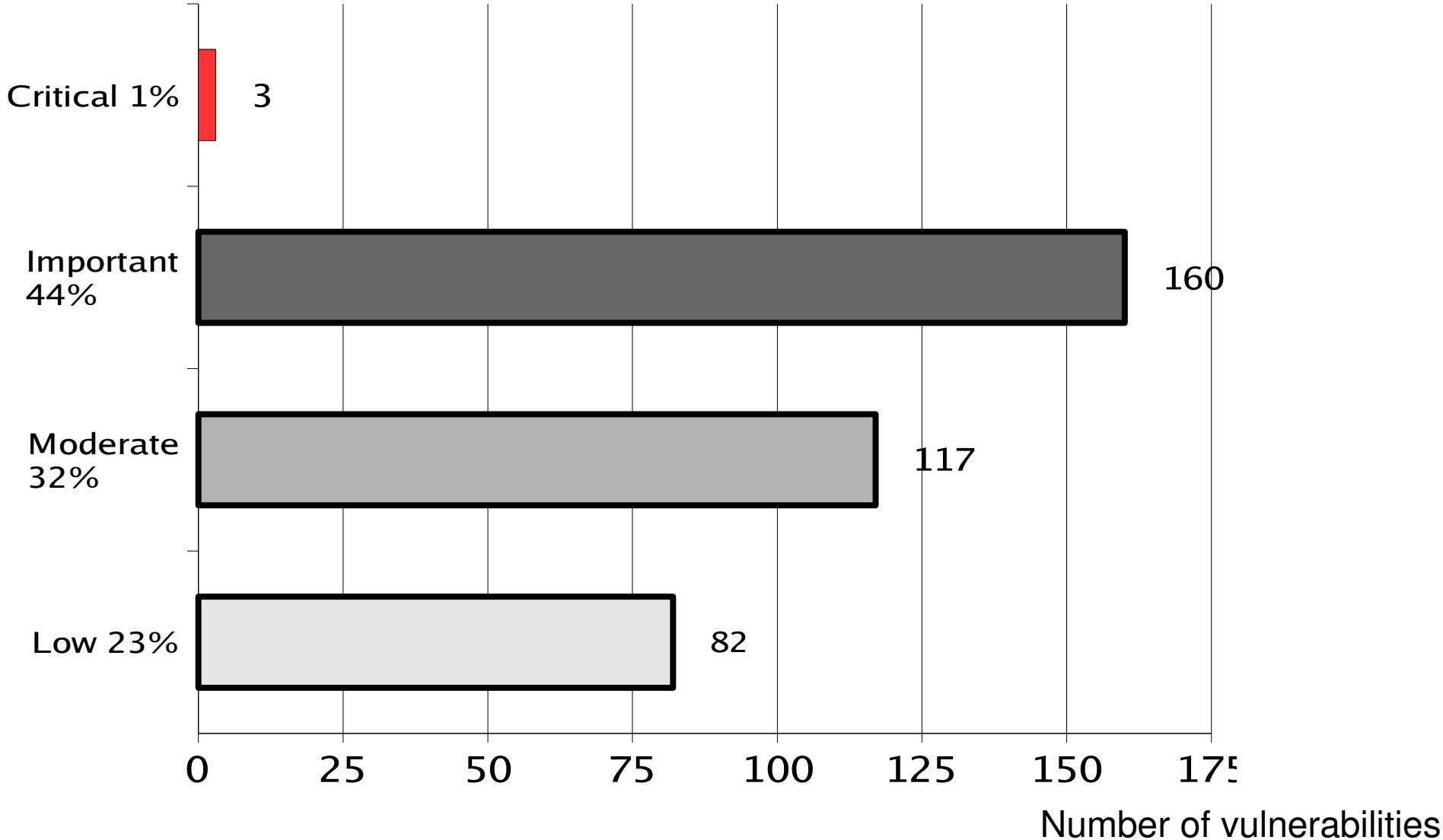
The “Defense in Depth” philosophy

- Any system, configuration, application and even software development have to be thought in a secure way
- It uses several security layers to secure systems and data.
  - Use of multiple computer security techniques to help mitigate the risk of a one defence being compromised or circumvented
  - Ex: using antivirus on both workstations and file/mail servers
- This concept should be applied in every systems, expecially on those that hold sensible data or whenever security is highly required
- This translate in:
  - Secure the OS
  - Secure access to the network
  - Use a common/central authentication and authorization
  - Secure the applications
  - Think and develop applications in a secure way

Advantages of OSS on Security

- Open Source means that the code is publically available
- A great number of people (the community) can evaluate the code of the project and:
  - Find memory leaks in the code (programming error)
  - Find a poorly secure algorithm/protocol (design error)
  - Have a peer review (from expert people)
- As a result we will (and a given vendor will):
  - Have the best code and algorithm
  - Have a quick fix for the problem (even in day 0)
    - Most of the “white-hats” produces both an exploit and a patch for the program
- Most of the projects reuses standard algorithm/code (libraries), thus eliminating common problems.

Linux (RHEL) default install vulnerabilities:



Misleading messages

> "Year-to-date for 2005, Microsoft has fixed 15 vulnerabilities affecting Windows Server 2003. In the same time period, for just this year, [Red Hat] Enterprise Linux 3 users have had to patch over 34 vulnerabilities and SuSE Enterprise Linux 9 users have had to patch over 78 vulnerabilities"
>
> -- Mike Nash, Microsoft, Feb 2005

- However, using the Microsoft severity scale:
  - Windows Server 2003 had three critical vulnerabilities
  - In Red Hat Enterprise Linux 3 full install had zero critical vulnerabilities
  - And this only counts their vulnerabilities that actually got disclosed
  - ... and not all vulnerabilities are disclosed ;-)

Security of the OS: hardening

- Out of the box, nearly all operating systems are configured insecurely.
- Hardening is minimizing a computer's exposure to current and future threats by fully configuring the operating system and removing unnecessary applications.
- How it works:
  - Remove unecessary users/groups
  - Disable unused services
  - Configure securely all used applications
  - Configure the strongest authentication possible
  - Use and configure firewalls/IDS/IPS/MAC Systems...

Security of the OS: minimization

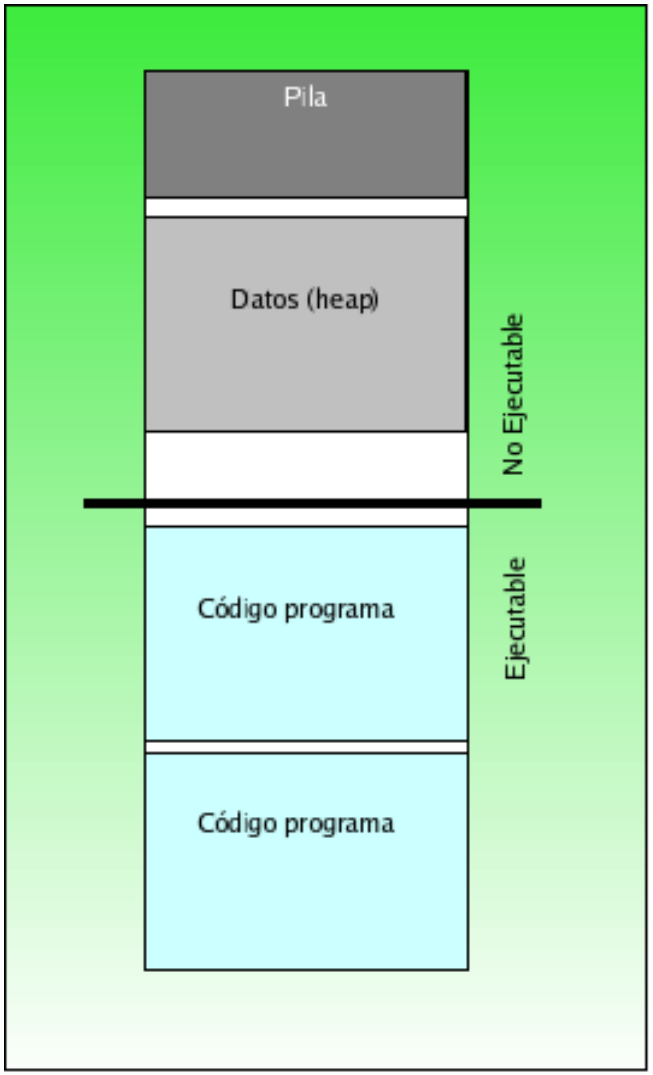
- Minimization is reducing the amount of software on a system
- The majority of system penetrations are accomplished through the exploitation of security holes in the operating system (OS) itself
- Fewer software components on a server means fewer security holes to detect and fill
- Minimizing the number of OS modules installed on a server can greatly improve overall system security by reducing the number of vulnerabilities.
- Usually OSes are installed with everything by default
  - Solaris, Windows, etc...
  - Vendors do not usually support miminized systems

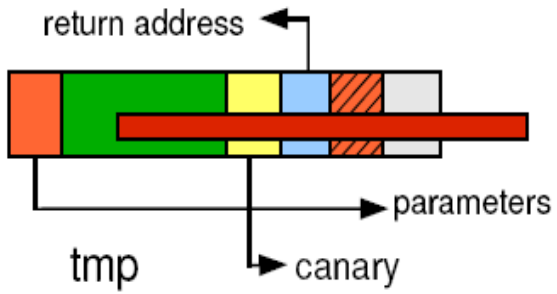
Security of the OS: how is Linux

- Mostly Linux distribution are hardened and secured by default (at least RHEL and Fedora are ;)
  - Only SSH opened and other services in loopback only
  - No unsecure/unencrypted services (telnet/ftp/r*)
  - Firewall and Mandatory Access Control (MAC) systems enabled by default
  - All unnecessary users/groups set to nologin
  - Default packages set to minimum (minimization)
    - The admin installs only what is strictly necessary to run the server
    - Minimal systems are usually supported!!

Security of the OS: kernel

- Buffer overflow:
  - It injects an arbitrary code (usually a shell) in the program's data area and execute it
  - Attackers gain access and privileges of the exploited program
- How to prevent:
  - No-Execute and Exec-shield technology
    - Software emulation of the no execute of the data area
    - Flag data memory as non-executable and program memory as non-writeable
  - PIE (Position Independent Executable)
    - Randomization of the application address in the stack





Security of the OS: other features

- Restricted Memory Access
  - Restricts how the kernel memory (/dev/mem) can be overwritten. This prevents several rootkits from functioning resulting in a safer and more secure system.
- Kernel signature (ksign)
  - Signature of kernel modules, to allow only certains gpg signed kernel modules to be loadeed
  - Avoid rootkits than hide themselves
- Secure application compile (FORTIFY_SOURCE and Stack-Smashing protector)
  - "FORTIFY_SOURCE" is a gcc option that detect and prevent a subset of the buffer overflows before they can do damage (unchecked buffer size)
  - Compiled programs marked with “canary words” (see reference)
- ELF (Executable and Linkable Format) Data Hardening
- RPM Signing
  - Each package/application is signed from the vendor so that any change is tracked (same effect as tripwire)

Security in the OS: MAC and SELinux

- Mandatory Access Control (MAC) is a kind of access control defined by the Trusted Computer System Evaluation Criteria
  - "[...] restricting access to objects based on the sensitivity of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity". (from Wikipedia)
- The most used MAC system in Linux is Security Enhanced Linux (SE Linux)
  - Developed initially by NSA (National Security Agency)
  - Several contributors, such as Red Hat, Tresys, IBM, ...
  - Certified Common Criteria (EaL4+) and used by military
    - When used in "stricted" mode is even more secure
  - Based on policies that confine user programs and system services to the minimum amount of privilege they require to do their jobs

SE Linux: features overview

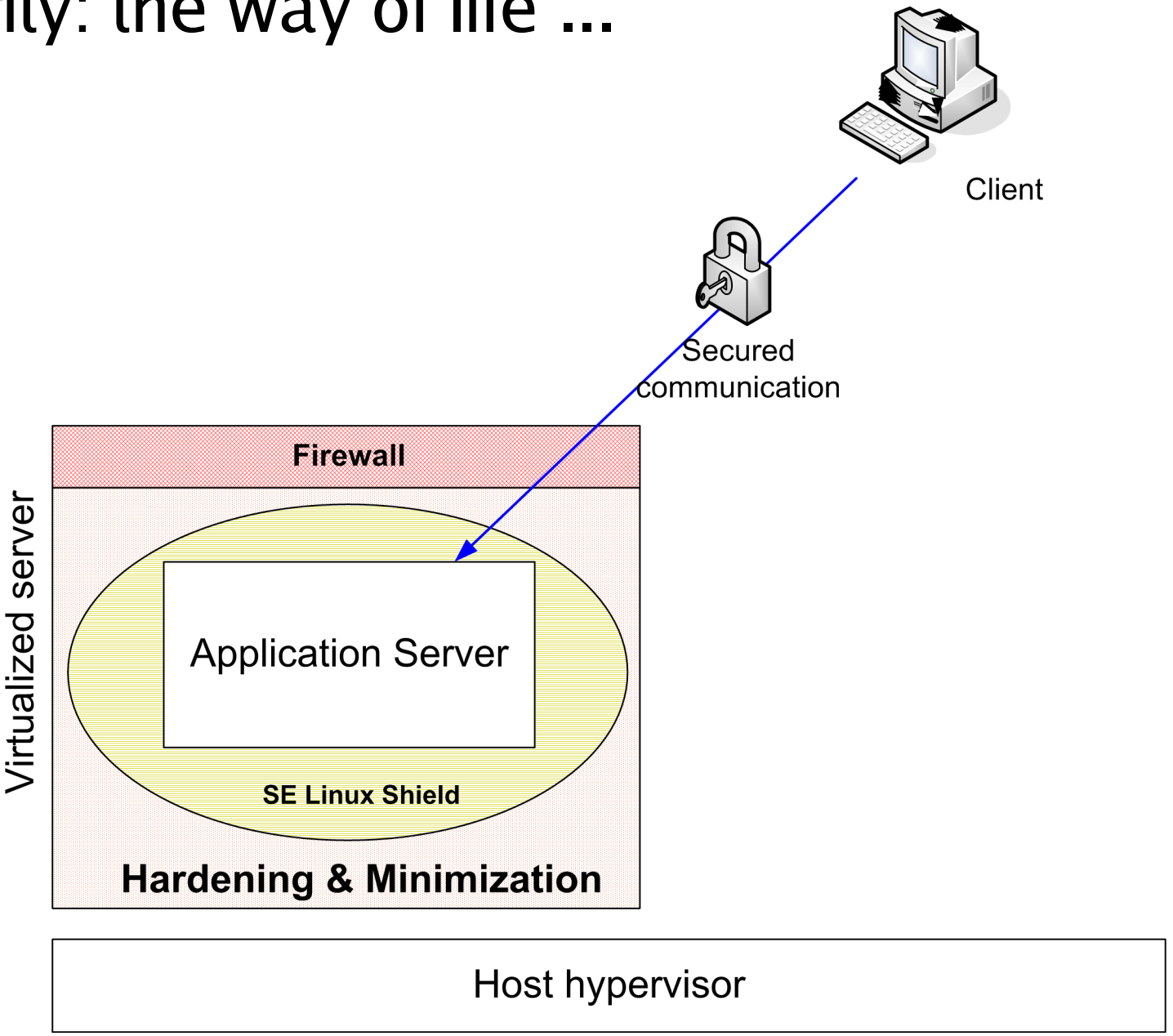
- Clean separation of policy from enforcement
- Well-defined policy interfaces
- Support for applications querying the policy and enforcing access control (for example, crond running jobs in the correct context)
- Independent of specific policies and policy languages
- Independent of specific security label formats and contents
- Individual labels and controls for kernel objects and services
- Caching of access decisions for efficiency
- Support for policy changes
- Separate measures for protecting system integrity (domain-type) and data confidentiality (MLS aka Multilevel security)
- Very flexible policy
- Controls over process initialization and inheritance and program execution
- Controls over file systems, directories, files, and open file descriptions
- Controls over sockets, messages, and network interfaces
- Controls over use of "capabilities"

Security in the OS: patching

PATCHING IS IMPORTANT!!!

- Always keep up-to-date the system's security patches
- It is important a management system that is able to apply all the patch seamless and avoid human errors
  - After buffer overflows and “zero day”, the major security concerns are about human error on configuration files
- That is:
  - Applying automatically patch and rollback if necessary
  - Compare software between machines (ex: cluster)
  - Use a standard, proven, configuration file for common services (ssh, authentication, ...) and distribute it
  - An example is Red Hat Network Satellite

Security: the way of life ...



Authentication and authorization

- Just to be clear on terms:
  - Authentication is proving who you claim to be (who you are)
    - Deals with identifying the users
    - Can be:
      - Username/Password (and variants such as kerberos)
      - One Time Passwords (OTP)
      - Two Factor (key based: Smart Cards/Biometric)
  - Authorization is giving permission to users (what you can do). Sometimes referred as “profiling”.
    - After the authentication phase, the user is profiled
    - The application (ex: login shell, web application, ...) will give appropriate rights based upon certain parameters
    - Parameters usually stored in LDAP, but also in databases

Strong authentication: Smart Cards/Biometric

- Autentication usually based on username and password
  - Passwords can easily eavesdropped
  - Kerberos and cryptography helps on protecting passwords but ...
  - ... don't prevent users to give away they're passwords or copying it (keyloggers, post-it, social engineering, ...)
- The best way is the famous “something you have” and “something you know”
  - Smart Cards and biometric autentications cover “something you have”
  - PINs or pass-phrases are usually “something you know”
  - The only way is stealing the smart card (and is not possible for biometric apart kidnapping ;)
  - Smart Cards hold private keys of a PKI infrastructure (and PIN or fingerprints unlock the keyring)
  - It can be revoked at any time

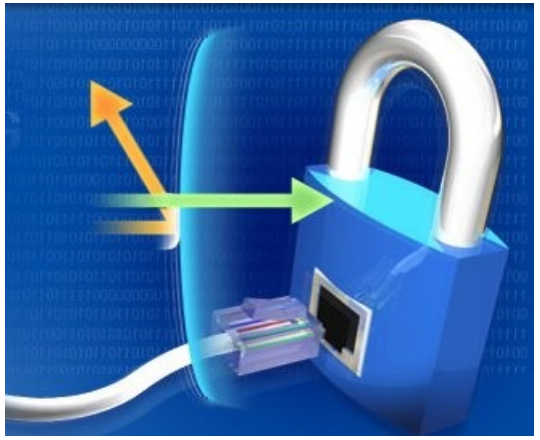
Smart Cards and Open Source

- Smart Cards are integrated with the OS and the applications
- In the login phase the smart card is handled by the PAM subsystem (pam_pkcs11)
- In intranets it can be handled with kerberos through the pkinit (public key init phase), so without modifying the applications
  - only KDC and client must be aware of x509 certificates
  - No modification to existing “kerberized” applications such as browsers, mail login, ssh, ...
- In extranets and public sites, web applications can deal with client certificate authentication
  - No need to exchange other informations apart from the CA and CRLs certificates
- PKI infrastructures (i.e. Software for handling Certification Authorities) are available as OSS and supported by vendors
- GPG also support smartcards :-)

Authorization and Profiling (roles)

- Every user should be granted the least privileges to do the job
- Profiles and roles should be stored on a central repository
  - Advantage is that authorization is cross to all systems/applications
  - Usually are stored in LDAP
    - OpenLDAP and Fedora Directory Server (RHDS) are the most famous in the OSS world
- Each application should be designed to use profiling
  - Java applications should use the JAAS APIs
  - PHP/C/C++/Python should perform look-ups in LDAP directly
- Examples (in infrastructure software)
  - Apache with mod_authz_ldap
  - AllowUsers in SSH
  - Sudo with LDAP

Application Security



- Each application should be configured securely!!!
  - This statement can't be stressed enough
  - Most of the “default installations” are opened
  - Hardening should involve the whole platform, not only the OS
  - Configuration depends on the application itself
    - Well-known for “infrastructure” software such as Jboss
    - Contact your vendor or the OSS mailing lists for more information
- Use application-layer firewalls
  - In the “Defense in Depth” philosophy give the edge protection layer
  - Everything incapsulated on port 80/443 (and 25) !!!
  - The most valuable example in OSS is ModSecurity
    - Web Application Firewall
    - Real-Time Monitoring and Attack Detection
    - Attack Prevention and Just-in-time Patching
  - Other examples: spamassassin, clamav, squid + dansguard, SIP Express Router (SER), ...

References

- Fedora security
  - http://fedoraproject.org/wiki/Security/Features
- ExecShield
  - http://www.redhat.com/magazine/009jul05/features/execshield/
- ModSecurity
  - http://www.modsecurity.org/
- GCC Compiler
  - http://en.wikipedia.org/wiki/Stack-smashing_protection
  - http://gcc.gnu.org/ml/gcc-patches/2004-09/msg02055.html
- SE Linux
  - http://www.nsa.gov/selinux/
  - http://fedoraproject.org/wiki/SELinux
- RHEL certifications
  - http://www.niap-ccevs.org/cc%2Dscheme/st/?vid=10125

Questions?

Thank you!

Giuseppe “Gippa” Paternò
Red Hat - Solution Architect & EMEA Security Prod. Expert
gpaterno@redhat.com