Security and security controls in operating systems

A *quantitative* approach
2015-02-16

Andreas Jonsson
andreas@romab.com
Robert Malmgren
rom@romab.com

1 minute presentation

- Consultant in IT and infosec since 20+ years
- Working alot on with critical infrastrucutre protection, process control, SCADA security etc, but also in financial sector, government, etc
- Work covers everything from writing policies, requirement specs and steering documents to development, penetration testing, incident handling and forensics

Outline of talk

- Intro
- Background and basics
- Security problems & vulnerabilities
- Formal security models
- Example of operating systems and security
- Trends

Some short notes

- The focus is on general operating system used in general computers - COTS products
  - *Embedded systems*, *code for micro controllers*, etc often lack most fundamental security features
  - Some experimenal OS's and domain specific solutions have better-than-average security concepts and security controls, e.g. military grade usage

Background and basics

Intro - foundation

- Modern software is normally formed into components, parts and layers in *systems*
- Complex systems
  - …run multiple programs at once,
  - …have multiple users,
  - …store huge amounts of data,
  - …is interconnected via networks

Intro - foundation

- This there is to built-in security into the foundation of the systems - the operating system
    - To identify and authorise users of the system
    - To allow for an environment where necessary basic controls are in place
    - To prevent unauthorised access to OS resources

Intro - *just the basic facts*

- All software is prone to bugs
- Some bugs will have an impact that can have security implications - data leaks, destruction of data, privilege escalations

Intro - *just the basic facts*

- Some bugs help to circumvent security mechanisms
- Some security designs are flawed, or build on flawed assumptions

Intro - *just the basic facts*

- Some bugs are undiscovered for some time, they lay latent
- Once discovered, they can be abused, if it is an security vulnerability, that can be exploited
- A discovered security bug, is sometime called a 0day, until it is mitigated

Some concepts and principles

TCB - *Trusted Computing Base*

RBAC - *Role Based Access Control*

MAC - *Mandatory Access Control*

Principle of least privilege

DAC - *Discretionary Access Control*

Principle of least surprise

Operating system security

- Security problems in the operating system can affect the integrity of the system itself
  - Someone else can control the system to their own liking - *pwnd!*
  - Bugs in OS kernel can affect system integrity
- Security problems with the operating system can in turn affect the security in *applications* and *subsystems* (databases, middle ware, etc)

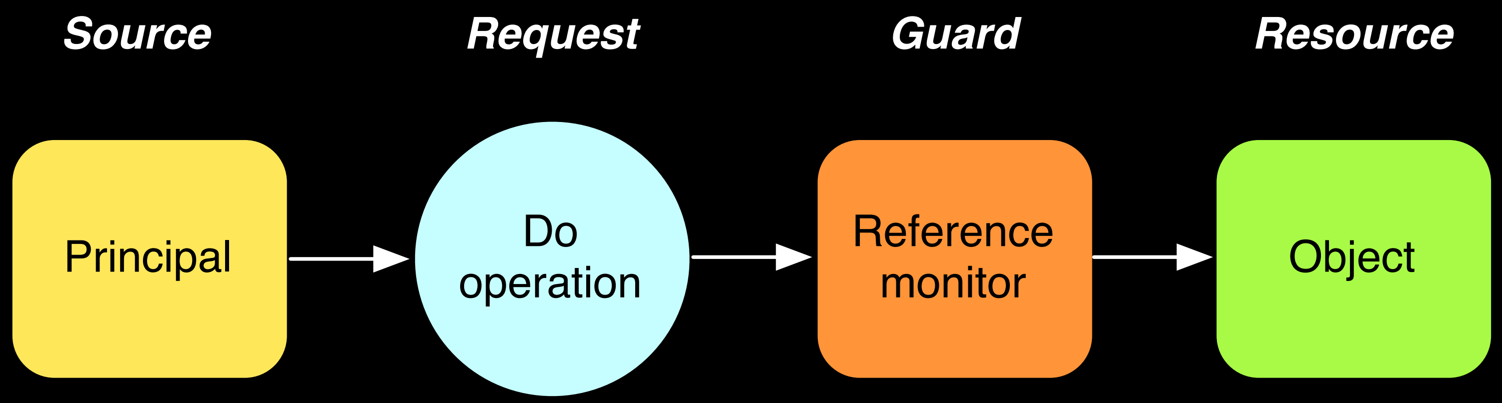
http://en.wikipedia.org/wiki/Pwn

Capabilities and requirements

Need	Description	Example
Protect a system resource	*Prohibit* ***malicious*** *or* ***unintentional*** *access to system resources*	System tables, direct access to I/O-units, memory protection
Authorization checks for usage of system calls and system resources	*Provide controlled access to system, so that system mainain system integrity and provide continuous security to application and information*	reference monitor

Some important concept

- Reference monitor



- Trusted Computing Base, TCB

[1] Lampson et al: *Authentication in Distributed Systems: Theory and Practice*

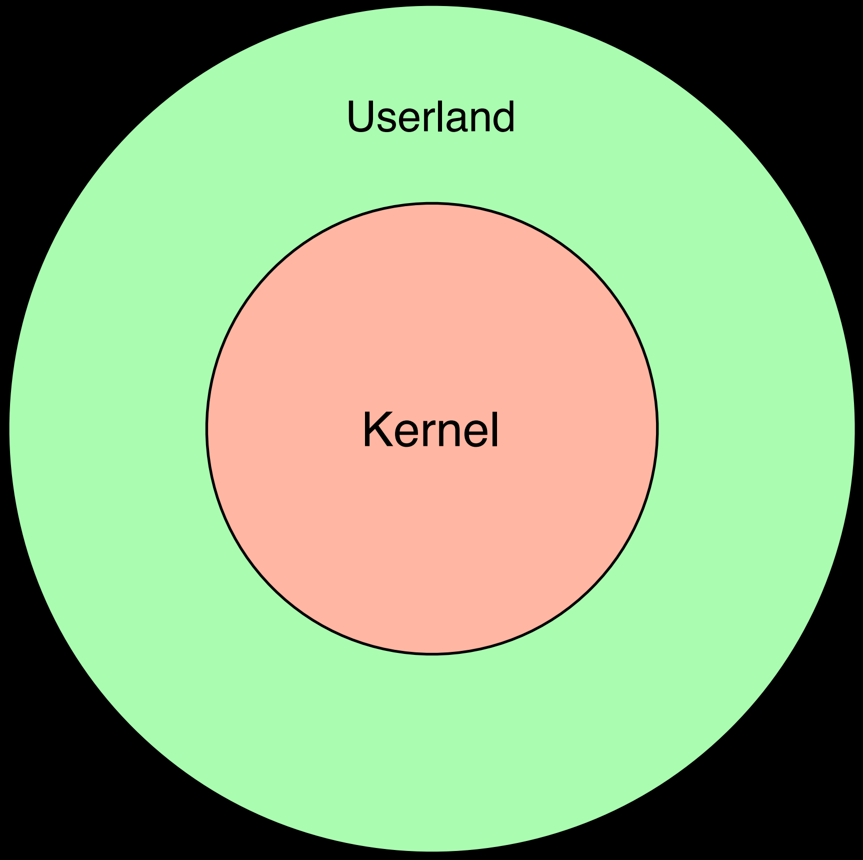
Principles for secure design*

Economy of mechanism	Keep the design as simple and small as possible
Fail-safe defaults	Base access decisions on permission rather than exclusion
Complete mediation	*Every access* to *every object* must be checked for authority
Open design	The design should **not** be secret
Separation of privilege	technique in which a program is divided into parts which are limited to the specific privileges they require in order to perform a specific task
Least privilege	Every program and every user of the system should operate using the least set of privileges necessary to complete the job
Least common mechanism	Minimize the amount of mechanism common to more than one user and depended on by all users
Psychological acceptability	It is essential that the human interface be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly

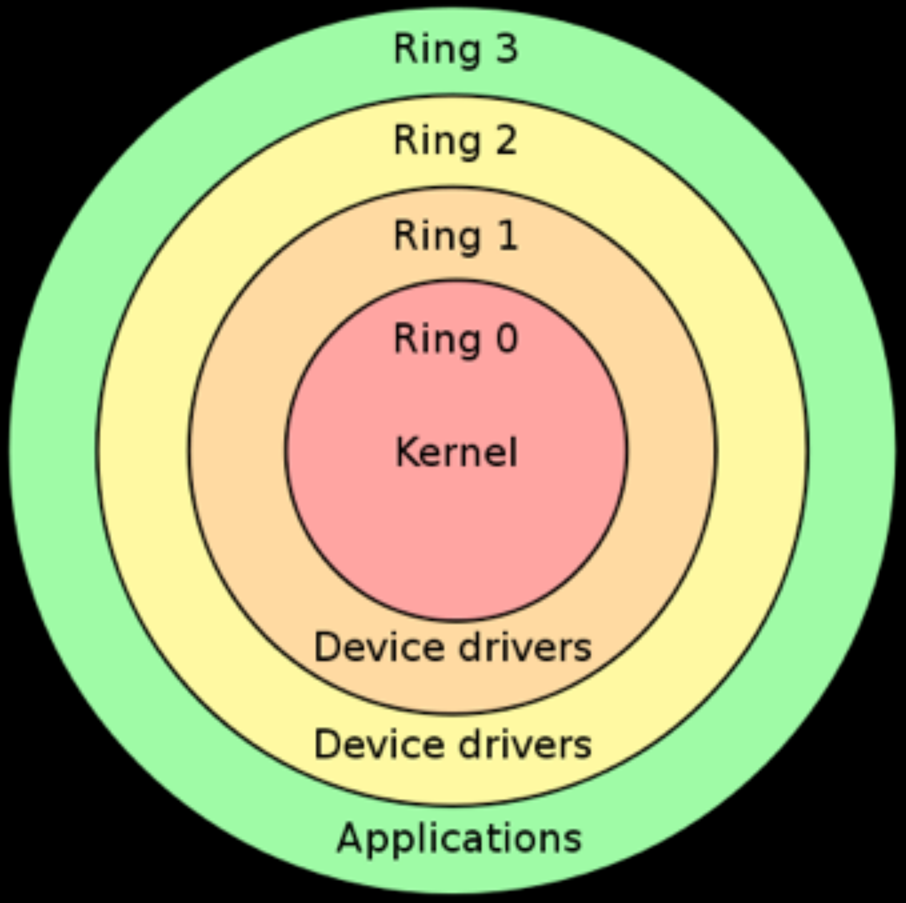
JEROME H. SALTZER et al *The Protection of Information in Computer Systems* http://www.cs.virginia.edu/~evans/cs551/saltzer/

The classical *ring model*

UNIX



x86



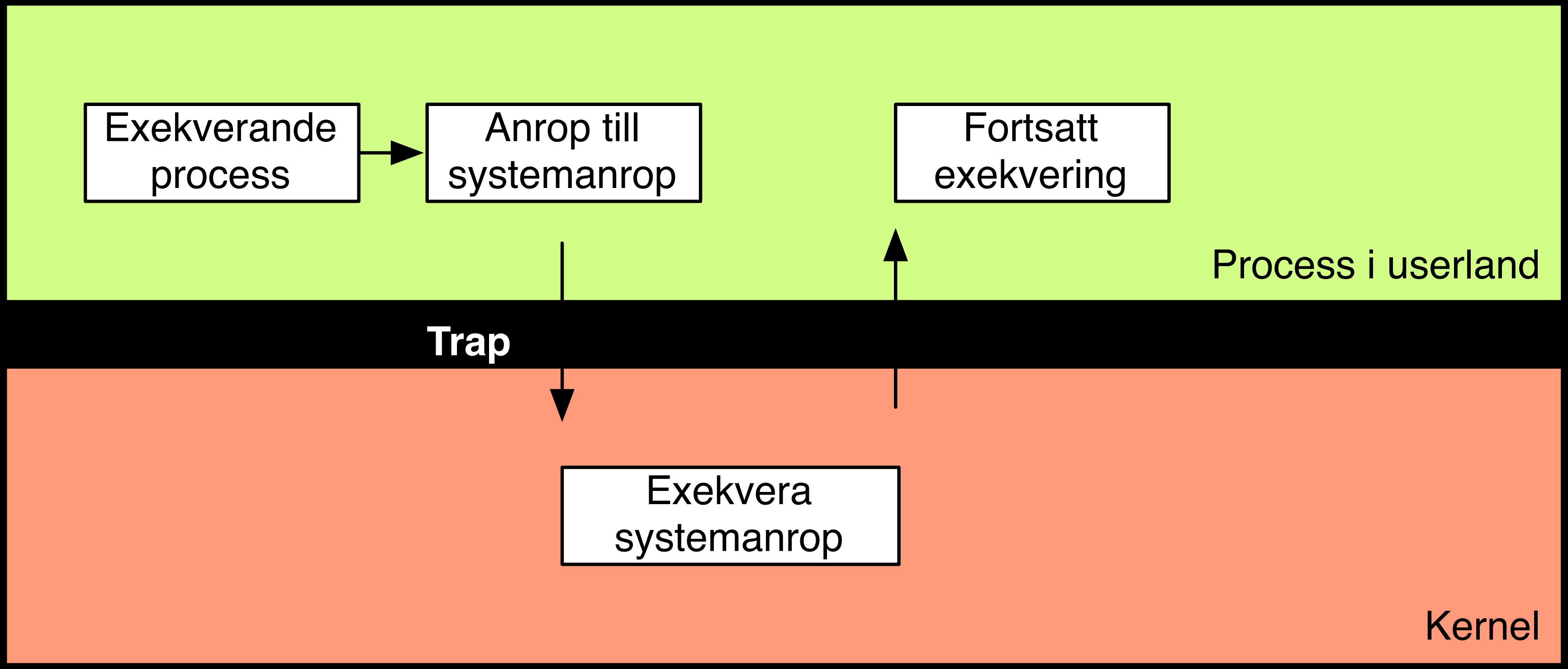
Least privileges

🟩
🟨
🟧
🟥

Highes privileges

Källa http://en.wikipedia.org/wiki/File:Priv_rings.svg

Interaction between application and OS


Exekverande process

Anrop till systemanrop

Fortsatt exekvering

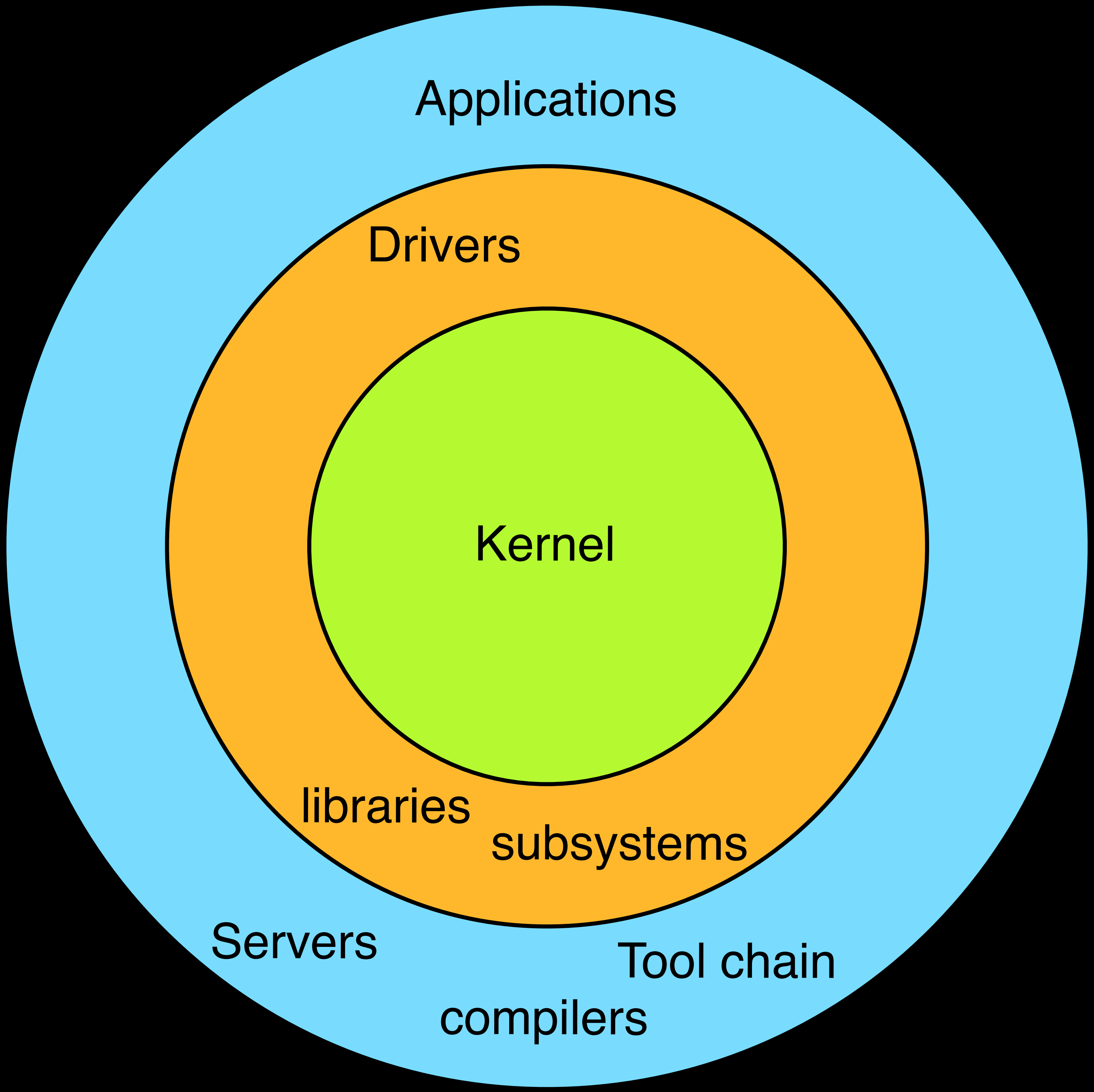
Process i userland

Trap

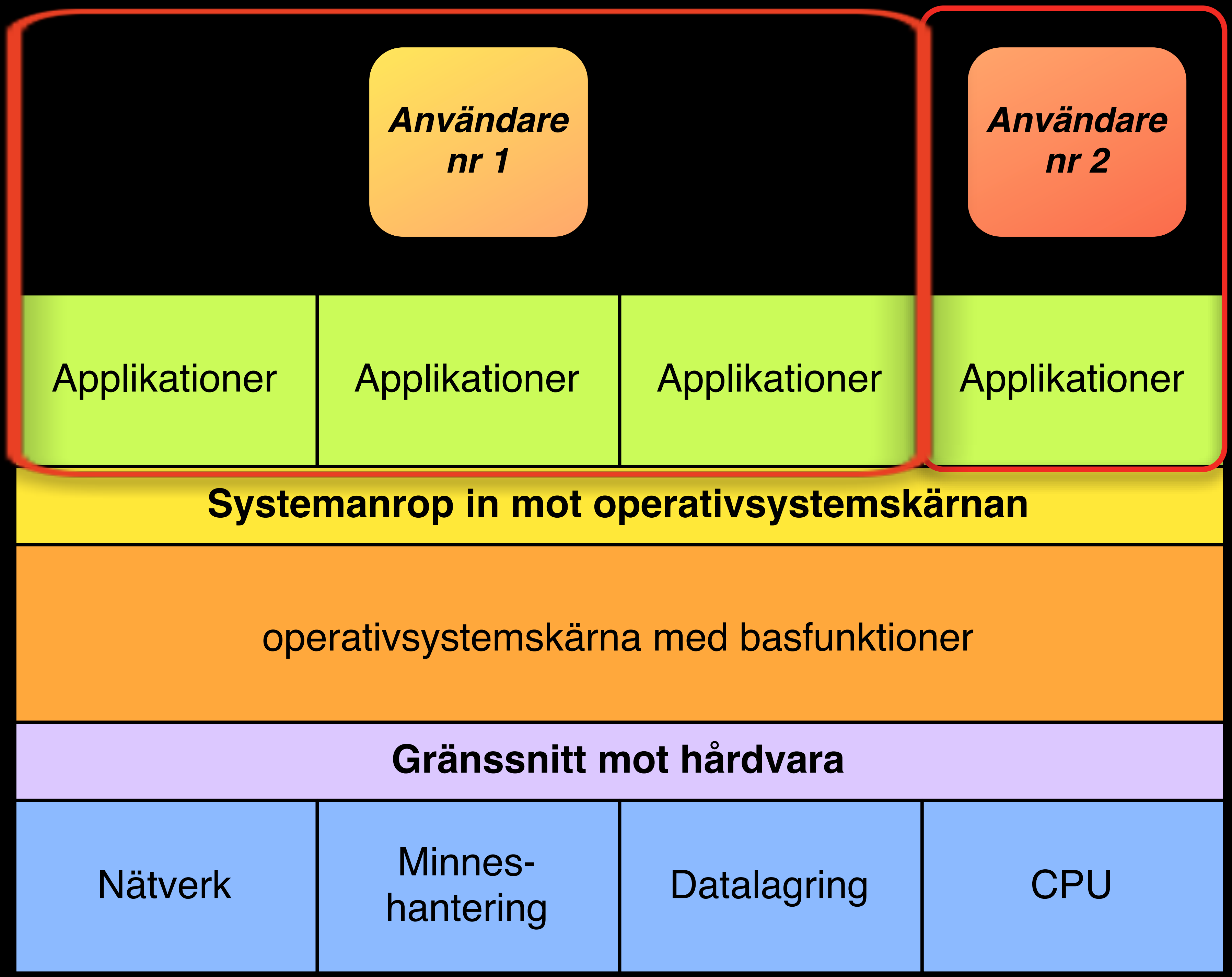
Exekvera systemanrop

Kernel

Overview of operating system (1/2)



Overview of operating system (2/2)



Problem with these pictures and concepts

- Layering violation
  - some software might skip a layer and call an underlaying layer directly and hence bypass controls
- In some scenarios attackers might come an unexpected way
  - Attacking from host operating system against guest operating systems in a virtual machine environment

Memory handling

- RAM memory is a central resourse that in a controlled way must be shared and handled *between operatingsystem, applications and other components*
- Modern computer systems have hardware support for memory protection, e.g. MMU
  - OS support is required to use the hardware supported memory protection

File system

- A file system is often a central component in a computer system w.r.t. security and protection
- Besides the actual *file content*, there is *meta data* that is of importance
  - File owner, dates of creation/change/access, access information, security labels, etc
- Manipulation of meta data can in some cases be more serious security breach than the manipulation of the file content itself. Or a combo of both can be misleading and hide the fact that a file has been altered

Local filsystem

File system	Description	Comment
FAT	*No access control*	Classic DOS
NTFS	*Discretional Access Control via ACL*	Advanced possibilities to make controls
UFS	*Discretional Access Control, writing & program execution for owner, group, "others"*	Simple access controls

Network file systems

File system	Description	Comment
NFSv3	*Hostbaserad accesskontroll, uid*	Trivial to circumvent

NFSv4	*Secure RPC, KRB5a, KRB5p, KRB5i*	Require a Kerberos server, KDC
		a= authentication
		i=integrity = calculate MAC
		p= privacy = encrypt packet

SMB/CIFS	*KRB5a*	

Comparing security in Operating systems (1/4)

- When in time was the system developed?
    - What was the state-of-the-art at that time?
    - What trends where currently in fashion?

Comparing security in Operating systems (2/4)

- Development methodologies
  - Open Source or Closed Source?
  - What support do one use to ensure that security is *built into the product*?
  - How does one ensure that implementation is a correct representation of the design, that is a correct interpretation of the analysis?

> *"Given enough eyeballs, all* _bugs_ *are shallow"*
> - Linus' Law

Comparing security in Operating systems (3/4)

But really, what good is this comparison?

Write more code = get higher salary?
Manage a 200K-SLOC project is *cooler* than a 5K-SLOC?

More code = more bugs?

More code = more *security checks* and *advanced concepts* like crypto, resillient failure checking built into everything?

But certainly, complexity is considered bad and evil in the context of security.
And there is often a relation between complexity and size of program

Comparing security in Operating systems (4/4)

- What can one gain by having formal certification of operating systems, subsystems or application
  - Trusted Computer System Evaluation Criteria (TCSEC), Common Criteria, etc
- More a theoretical excersice than of any real value?

Example of different protection solutions

General example of control principles

Security controls	**Description**	**Example**
Random numbers	*Make a resource non-deterministic*	File names, proccess ID,'s port numbers, sesssion keys, transaction numbers, DNS queary ID's, timing
Encryption	*Protection against eavesdropping or unauthorized access*	network traffic, file content, disk partitions, memory pages, swap files/ area
Hash values	*Protection against unnotised changes,*	passwords, checksums on files
Logs	*Traces, error messages and dumps from systems and applications*	Syslog, eventlog, audit, BSM

General example of control principles

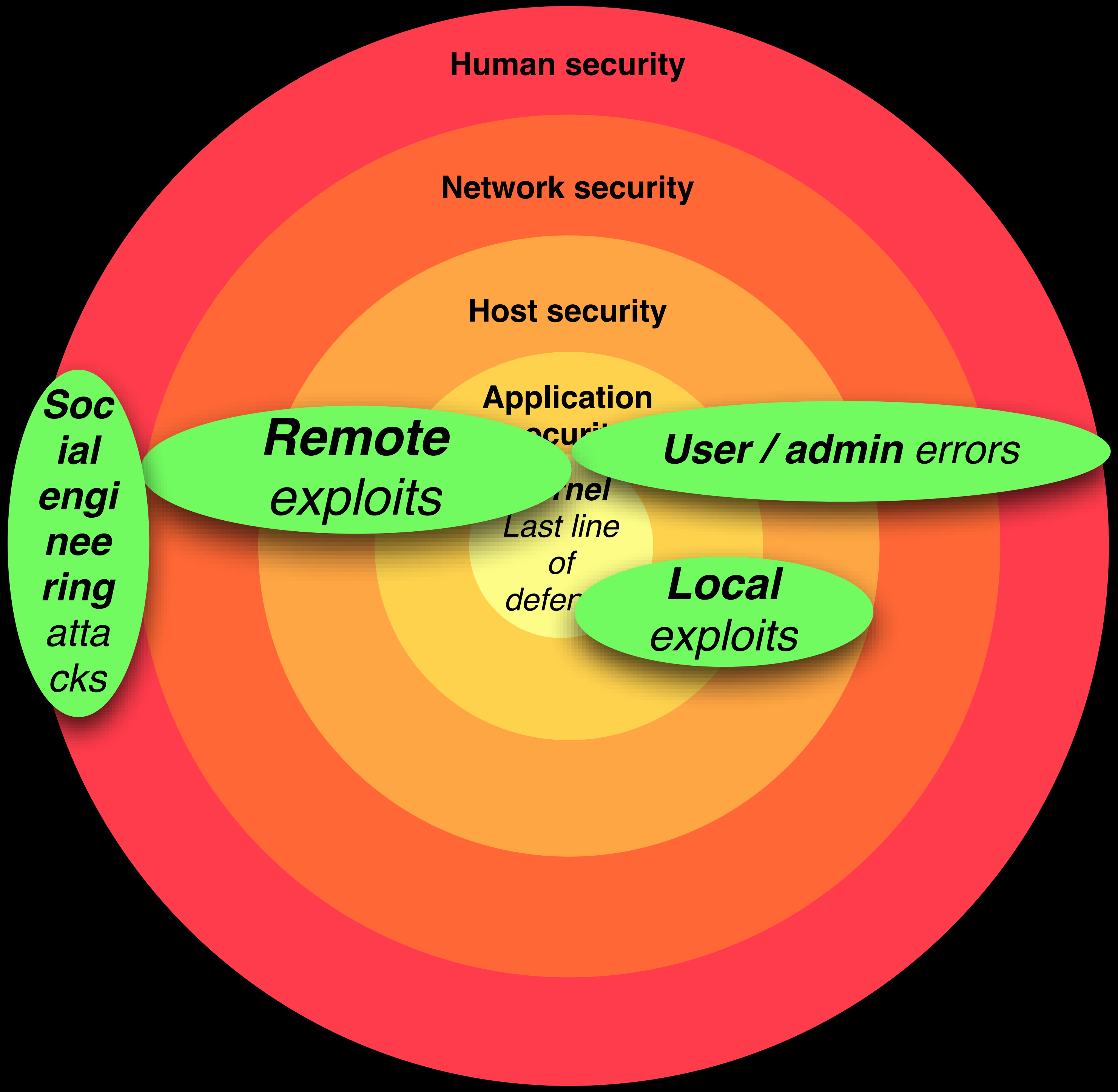
Security controls	Description	Example
Compiler generated airbag - canary	*Make sure buffer overflows dont gets undetected*	ProPolice, VisualStudio /GS
ASLR	*Make sure its hard to write code that knows of addresses to misuse. Where did that lib go?*	Windows Vista/7/2008, OpenBSD, Linux, MacOSX, etc
DEP, NX, W^X	*Make sure memory is not executable*	Windows Vista/7/2008, OpenBSD, Linux, MacOSX, etc

General example of control principles

Security controls	**Description**	**Example**
Scrubing, zeroing	*Make sure that old data areas are cleaned before usage or returned to system*	file systems,VM system

Examples of vulnerabilities and attacks

Where do attacks occur?



Examples of threats and attacks

Wrong file permissions

plain text in RAM

Confidentiality

Bypasswd security checks

fork bombs

SYN flood

malformed network packets

Availability

*un*intentional filling of disk partition

intentional filling of disk partition

Manipulated system configuration

System integrity

Manipulated program binaries

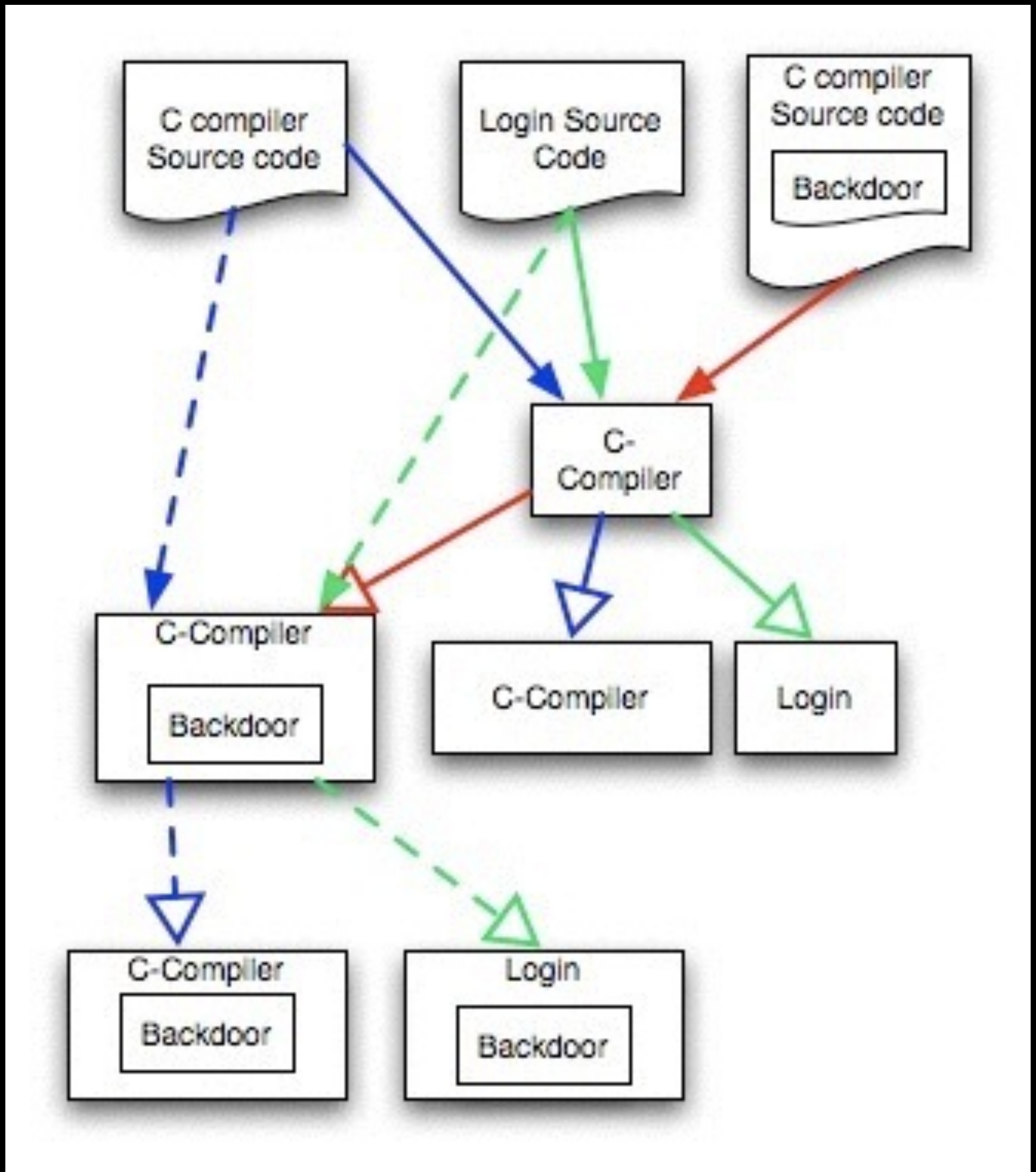
Manipulated user files

Data integrity

Zapped system logs

Some exempels of classic attacks (1/2)

- Ken Thompson’s trojanized c compiler
  - Modify the source code to the compiler to recognize if it recompile itself or the login program - insert backdoor in login
  - recompile compiler
  - remove source code changes and recompile the compiler
  - recompile the login program with the modified compiler
- No visible signs for humans or tools to see the backdoor in source code. Calls for binary inspection or decompilation.



Ken Thompson - TURING AWARD LECTURE: *Reflections on Trusting Trust.*
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.5728&rep=rep1&type=pdf

Some exempels of classic attacks (2/2)

- Create a symbolic link that is used to trick the system to overwrite an important file at a controlled point in time

```
ln -s /tmp/core /etc/passwd
```

Example of attacks

Attack method	Description
Rootkit	*Replace parts of applications or kernel with attackers code.* *Often contain built-in protection and deception parts to hide rootkit itself, as well as malicious code.* *Often created / built upon modified original source code.* *Name derives from earliest versions of threat that was created on UNIX systems*
time-of-check-to-time-of-use (TOCTTOU)	*Type of race-condition bug caused by (maliciously controlled) changes in a system between the checking of a condition (such as a security credential) and the use of the results of that check*

Example of attacks

Attack method	Description
Buffer overflow	Attacks that allow an attacker to *deterministically alter the execution flow of a program by submitting crafted input to an application.* Executable code is written outside the boundaries of a memory buffer originally used for storing data. The executable parts is somehow made to execute, eg by manipulate return adress to be used when a function call is finished. Real world examples: OpenBSD IPv6 mbuf's* remote kernel buffer overflow[1], windows kernel pool Synonyms and variants: Buffer overrun, Stack smashing, Heap smashing, format string bugs, memory corruption attack

* An *mbuf* is a basic unit of memory management in the kernel IPC subsystem

[1] http://www.coresecurity.com/content/open-bsd-advisorie

Example of attacks

- Attacks by attaching malicious hardware to buses and ports
  - Firewire and other DMA based methods to access memory of a computer (*evil maid attacks*)
  - UEFI attacks via Thunderbolt (*thunderstruck attack*)
  - Using JTAG interfaces to snoop & manipulate bus
- Removal of physical memory chips, (*cold boot attacks*)

Remember that there is a number of ways that all OS security controls can be bypassed, especially if the operating system is not running - a very good side-channel attack ;-)

Attacks and counter measures

?

Hijacking JIT compilers

ROP attacks

Address Space Layout Randomization (ASLR)

No-executable (NX, W^X) stacks

Data Execution Prevention (DEP)

More advanced buffer overflows, defeating canary

Stack canaries

Buffer overflow/memory corruption attacks

Note - several of these counter measures does not work for protection **within** the kernel

Attacks and counter measures

- *Chaining of attacks - combining a number of exploits to achieve goal*
  - finding and abusing a number of different vulnerabilities might allow an attacker to achieve goals not possible with just one potent exploit
  - *Code execution in gadgets (ROP) + sandbox escape + elevation of privileges + execution of privileged code*

Security models

Security controls in operating systems

Discretionary access controls

Subject *A* can decide how *an object* created by *subject* *A* can be interacted with by *subject* *B*

Mandatory access controls

The *System policy* decide how object from *subject* *A* can be interacted with by *subject* *B*

Discretionary access control *DAC*

- Exists in all COTS systems
- Conceptuallt work by having a subject (= user) have ownershop of an object (e.g. file, process) and by beeing owner have right to control access rights to the object
- Used in UNIX, Windows NT derived OS's, etc

Discretionary access control *DAC*

- Became a mandatory requirement to sell computers to DoD 1985
- First implemented in Windows NT 3.5 and Apple Mac in Mac OSX
- Simple concept to understand and administrate. Especially in (classic) UNIX where conceptually everything is a file in the filesystem

Role Based Access Control *RBAC*

- Created to implemented the *least-privilege* principle
- No users should have high privileges, all those should get these privileges from roles with clearly defined rights

Mandatory Access Controls *MAC*

- Mandatory access controls where created to naturaly implement the security policies of military organizations
- Lots of resources and research have been spent in this area
- But it have been hard to develop models and implementations that work well in real-life situations, especially in ordinary organisations

The *BIBA Integrity model*

- Created by Kenneth Biba in 1977.
- Primarily goal is to maintain ***system/data integrity***.
- Is built on a system policy where the following is legio:

“no *write up*, no *read down*”

- Can be seen as *BLP upside-down*

BIBA Integrity model

- A subject can only write to its own level and to lower levels
  - cannot write to higher labled objects
- A subject can only read its own level and higher

- *The point is that objects on one level cannot be affected by corrupted data on a lower level*

BIBA Integrity model

- Relatively simple policy, which simplifies implementation and administration
- Still hard to get a full working BIBA solution to gain acceptence in the real world
- Two example of where versions of BIBA is implemented in COTS is in Windows Vista and FreeBSD (TrustedBSD)

http://msdn.microsoft.com/en-us/library/bb625957.aspx
http://www.freebsd.org/cgi/man.cgi?mac_biba

BIBA Integrity model - in practise

- A dedicated system that runs a nameserver will be configured to use a biba policy
- *System binaries* and *system directories*, *name server binaries*, *DNS zone information* must be classified to belong to the ***high** integrity label*
- The subject, (in this case a users that runs the DNS server program, bind) is set to have a ***low** integrity label*

BIBA Integrity model - in practise

- The result is that the running name server process ***cannot alter any data at all*** (config files, zone files) in case it get hacked, but is allowed to read all necessary information

- To administer the name server, a separate user with an integrity label of **equal** level or higher (but that has some read issues), than the zone file must log in to *edit* it

BIBA integrity model

```
root@freebsd# setfmac -R biba/high /var/named
root@freebsd# setpmac biba/low /etc/rc.d/named start
```

Note: temporary filer, pid-filer etc must also be given biba/low labels

BIBA in *Windows Vista*

- Used in few places, most notably in IE that is runned in low-privilege mode.
  - Few tools to administer it
  - Will stop working if UAC is disabled - and UAC is often disabled
- Have 4 integrity levels: low, medium, high and system
- Microsoft dumped "no read down" - which sort of destroys the model....

UAC = User Account Control

Drawbacks with BIBA

- Hard to get a policy that really works in real-life situations

- Hard to strictly follow the model. Almost all implementations of BIBA have extra verbs that isn’t part of original BIBA model, eg Freebsd’s BIBA/equal, which is a way to make an object or subject that is excluded from the policy

http://fuse4bsd.creo.hu/localcgi/man-cgi.cgi?mac_biba+4

Multilevel security - *MLS*

- Focus is on confidentiality of information and information flow, not system integrity
- Also known as the Bell-La Padula, BLP, model
- Simple general rule:

  ”no *write down*, no *read up*”

- No write down is also known as the *confinement property* or *-property (star)
- Normally used in military style information management situations

http://www.albany.edu/acc/courses/ia/classics/belllapadula1.pdf
http://csrc.nist.gov/publications/history/bell76.pdf

Multilevel security - *MLS*

- MLS is built upon that all subjects get a *clearance level*, which is then used check the classification of an object

Level	**user**
TopSecret	Andreas
Secret	
classified	Robert
unclassified	

Note: The *security labels* used in the example is used from the classical military style example. They can be arbitrarly things, like ”outside”, ”DMZ”, ”inside” etc

Multilevel security - *MLS*

- A simple example of usage of MLS

  - During some field work Robert writes an report rapport, classified as “secret”.
  - Andreas and all users with clearance *secret or higher* can read, but not modify the report.

Multilevel security - *MLS*

- If andreas edit the report, his clearance is *tainting* the report, and the new classification is now "*top secret*".
- Information have a tendency to raise up in MLS system, since there are nothing that can happen that make is go downward.
- In the end, someone need to perform a manual reclassification work

Multilevel security - *MLS*

- Since DoD require MLS, a lot of COTS system have gotten MLS support
    - TrustedAIX, TrustedSolaris, TrustedHP-UX

- A standard problem is that they are several releases behind the stock version of the operating system, since they need to be re-evaluated all the time

Multilevel security - *MLS*

- MLS is problematic, but does solve *some hard-to-solve standard problems*
  - In a MLS sytem, a compromized web reader does not automatically get access to all files that a subject owns, just because it is runned by the subject (user) that created/owns the files
    - Encryption key files, secret reports, sensitive files

MLS in “modern” systems

- MLS exists in a number of different modern OS’s OS, e.g. FreeBSD, Solaris (trusted extensions), AIX 6.1 and Linux (SELinux enabled)

- Differences to the theoretical model is small, but important to make a practicaly useful system

http://www.freebsd.org/doc/en/books/handbook/mac-mls.html

http://www.ibm.com/developerworks/aix/library/au-AIX_MLS/index.html

MLS in “modern” systems

- MLS in Solaris 10 with ***trusted extensions*** is built on setting labels on container level, rather than subject and object level
- In SELinux it is often not used since TE, *Type Enforcement* both is more flexible and simpler to use to protect the information

POSIX 1.e

- IEEE standard defines *capablities*, acl, mac och labels
- The IEEE work was ended 1999 after 13 years and was never completely finished
- Lots of implementations of security models in UNIX have its root in POSIX 1.e

LSM - Linux Security Module

- Was created by Crispin Cowan/imunix 2001
  - To avoid locking certain security models into the Linux Kernel
- Framework to implement security models in Linux with as few kerne changes as possible
  - Also used to implement other security features, such as intrusion detection, etc
- Standard since 2.6 kernel
- Not completely different the MAC-modules in fbsd (trustedBSD) and kauth in netbsd

http://netbsd.gw.com/cgi-bin/man-cgi?kauth+9+NetBSD-current

http://en.wikipedia.org/wiki/Linux_Security_Modules

Apparmor

- Implemented using LSM for the Linux kernel
- Is built to create a white list for what application is allowed to do
- Implementents part of posix 1.e (capabilities)
- Mandatory

http://en.wikipedia.org/wiki/AppArmor

Apparmor

- Poison of choice in Ubuntu och SLES, instead of SELinux that competitors have chosen
- Much simplier policy language / configuration than other mandatory access controls
- Have a wizard functionality to create policies

Apparmor - rules

Symbol	Meaning
?	Any symbol besides /
*	any number of symbols besides /
**	* + /
[abc]	a, b, or c
[a-c]	a, b, or c
{ab,cd}	ab or cd

Apparmor - rules

Abbrev	Meaning
r	read
w	write
ux	unconstrained execute
Ux	ux + scrubed env
px	disc profile execute, *change* profil
Px	px + scrubed env
ix	inherit exec, *keep same* profil
m	Allow PROT_EXEC with mmap(2)
l	link

Apparmor - example for firefox

```
/usr/lib/firefox/firefox.sh flags=(complain) {
  /bin/basename rmix,
  /bin/bash rmix,
  /bin/gawk rmix,
  /bin/netstat rmix,
  /dev/log w,
  /dev/null rw,
  /dev/tty rw,
  /dev/urandom r,
  /etc/fonts/** r,
  /etc/ld.so.cache rm
  /etc/localtime r,
  /etc/magic r,
  /etc/opt/gnome/** r,
  /etc/passwd r,
  /etc/resolv.conf r,
  /home/*/.fontconfig/** r,
  /home/*/.gconfd/* rw,
  /home/*/.gconf/ r,
  /home/*/.gconf/* rw,
  /home/*/.gnome2_private/ w,
  /home/*/.mozilla/** rw,
  /home/*/.Xauthority r,
  /lib/ld-2.5.so rmix,
  /lib/lib*.so* rm,
  /opt/gnome/lib/GConf/2/gconfd-2 rmix,
  /opt/gnome/lib/**.so* rm,
  /proc/meminfo r,
  /proc/net/ r,
  /proc/net/* r,
  /tmp/gconfd-*/ r,
  /tmp/gconfd-*/** rwl,
  /tmp/orbit-*/ w,
  /tmp/orbit-*/* w,
  /tmp/ r,
  /usr/bin/file rmix,
  /usr/lib/browser-plugins/ r,
  /usr/lib/browser-plugins/** rm,
  /usr/lib/firefox/firefox-bin rmix,
  /usr/lib/firefox/firefox.sh r,
  /usr/lib/firefox/** r,
  /usr/lib/firefox/**.so rm,
  /usr/lib/gconv/** r,
  /usr/lib/gconv/*so m,
  /usr/lib/lib*.so* rm,
  /usr/lib/locale/** r,
  /usr/share/** r,
  /var/cache/fontconfig/* r,
  /var/cache/libx11/compose/* r,
  /var/run/dbus/system_bus_socket w,
  /var/run/nscd/passwd r,
  /var/run/nscd/socket w,
  /var/tmp/ r,
}
```

Note that this configure is very firefox and linux *version* specific

Apparmor - critics

- path-based instead of inod baserad
- The simplification wrt the wizarden, makes the simplification too much
- Only includes definied program, not the systemet as such or other programs
- Often is markedet to be more than it really is, e.g. RBAC

SElinux / Type Enforcement (te)

- *Type enforcement* is built on the concept that a subject is attachted to a *domain* and that object is attached to *types*

- In a matrix one define how domain-to-domain and domain-to-type interaction is allowed.

SELinux

- In the SELinux there is a security matrix called policy which can be targeted, strict, permissive or enforcing.
- targeted - what is allowed besides that which is explicit prohibited
- strict - nothing is allowed beside that is explicitly allowed

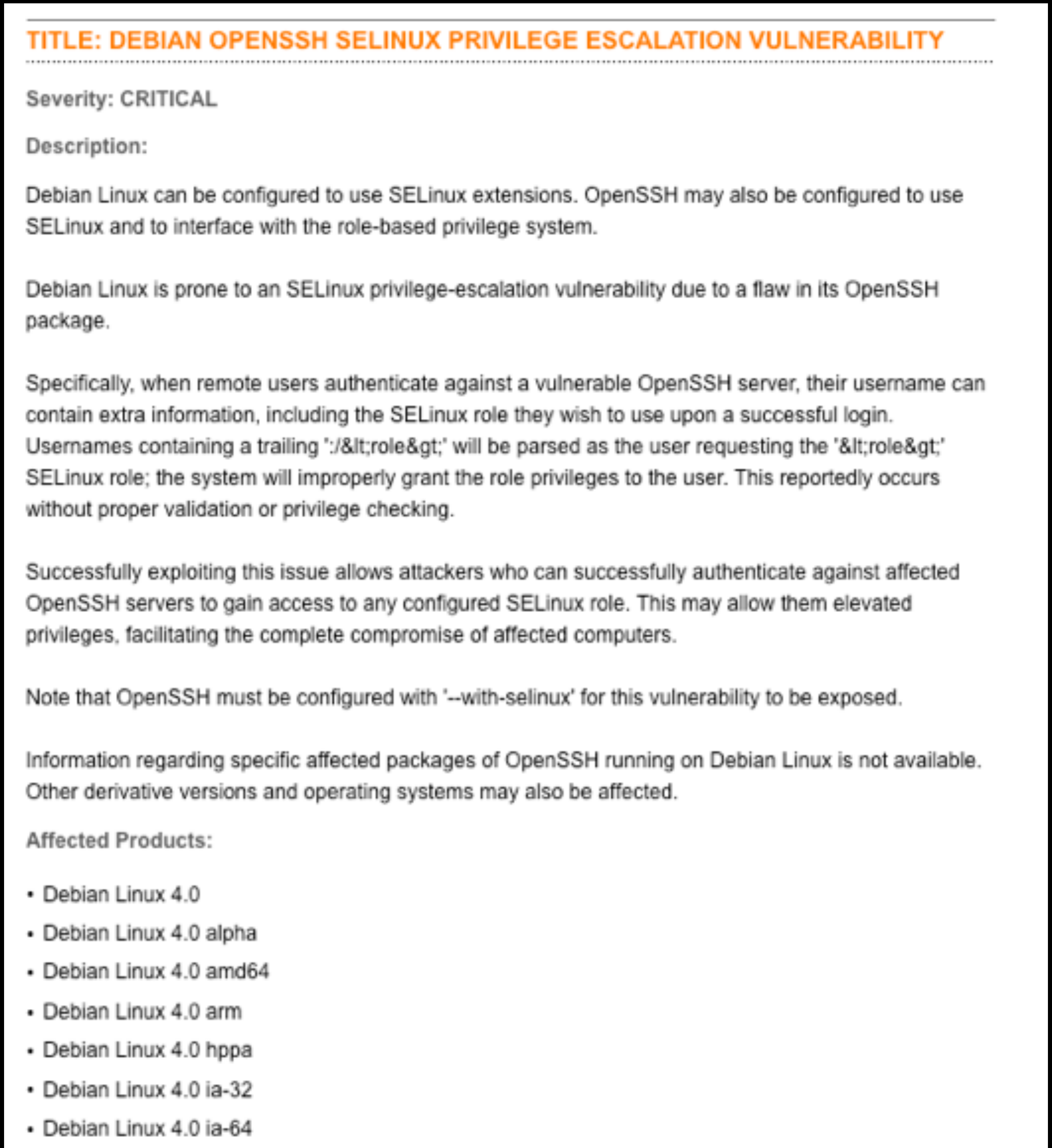
SELinux

- SELinux is used to lock things down - primarily services, but can in theory lock down anything
  - The focus on locking down services (e.g. network services) will result in that authorized users *will not be locked down* and gain advantages of any security controls from SELinux

SELinux

- Reference policy is maintained by tresys*
  - earlier by NSA
- Contain a few “trusted programs”,
  - e.g. su, sshd, login.
- These trusted programs must be able to perform so called ***domain transitions***.

* https://github.com/TresysTechnology/refpolicy/wiki



TITLE: DEBIAN OPENSSH SELINUX PRIVILEGE ESCALATION VULNERABILITY

Severity: CRITICAL

Description:

Debian Linux can be configured to use SELinux extensions. OpenSSH may also be configured to use SELinux and to interface with the role-based privilege system.

Debian Linux is prone to an SELinux privilege-escalation vulnerability due to a flaw in its OpenSSH package.

Specifically, when remote users authenticate against a vulnerable OpenSSH server, their username can contain extra information, including the SELinux role they wish to use upon a successful login. Usernames containing a trailing ':/<role>' will be parsed as the user requesting the '<role>' SELinux role; the system will improperly grant the role privileges to the user. This reportedly occurs without proper validation or privilege checking.

Successfully exploiting this issue allows attackers who can successfully authenticate against affected OpenSSH servers to gain access to any configured SELinux role. This may allow them elevated privileges, facilitating the complete compromise of affected computers.

Note that OpenSSH must be configured with '--with-selinux' for this vulnerability to be exposed.

Information regarding specific affected packages of OpenSSH running on Debian Linux is not available. Other derivative versions and operating systems may also be affected.

Affected Products:

- Debian Linux 4.0
- Debian Linux 4.0 alpha
- Debian Linux 4.0 amd64
- Debian Linux 4.0 arm
- Debian Linux 4.0 hppa
- Debian Linux 4.0 ia-32
- Debian Linux 4.0 ia-64

Important note to remember is that security code can add new security bugs

SELinux

- Is distributed in COTS Linux distributions such as RedHat and Fedora
- Is actively maintained by RedHat, Tresys, NSA and others
- The company Tresys is the maintainer of the reference policy and several selinux userland program
  - also sell separate policys for more program, tex razor

SELinux

- The model used to grant rights is extremely granular and powerful
  - exec_heap, exec_mem are permissions in SELinux
- The SELinux advocate Russel Cooker have test boxes for anyone to use where root-login is allowed for anonymous users
  - http://www.coker.com.au/selinux/

SELinux

- Drawbacks with SELinux
  - To create a flawless SELinux policy from scracth is **very hard** - often it is a copy-and-paste work from some existing policy, and thus might not really implement your intended design
  - To maintain a SELinux policy is non-trivial, compare for example with apparmor
  - Dependencies on trusted programs as well as classic data validation errors can result in security errors, as usual

GRsecurity

- Brainchild of Brad Spengler
- *NOT* based on the LSM concept
  - Brad is a vocal critic of the LSM concept and have developed PoC attacks agains LSM based security solutions
- It is released as a separate, non official, patch cluster to the Linux Kernel
- Some see the non-official status and ”hack” type of solution as unacceptable, e.g. Xorg

GRsecurity

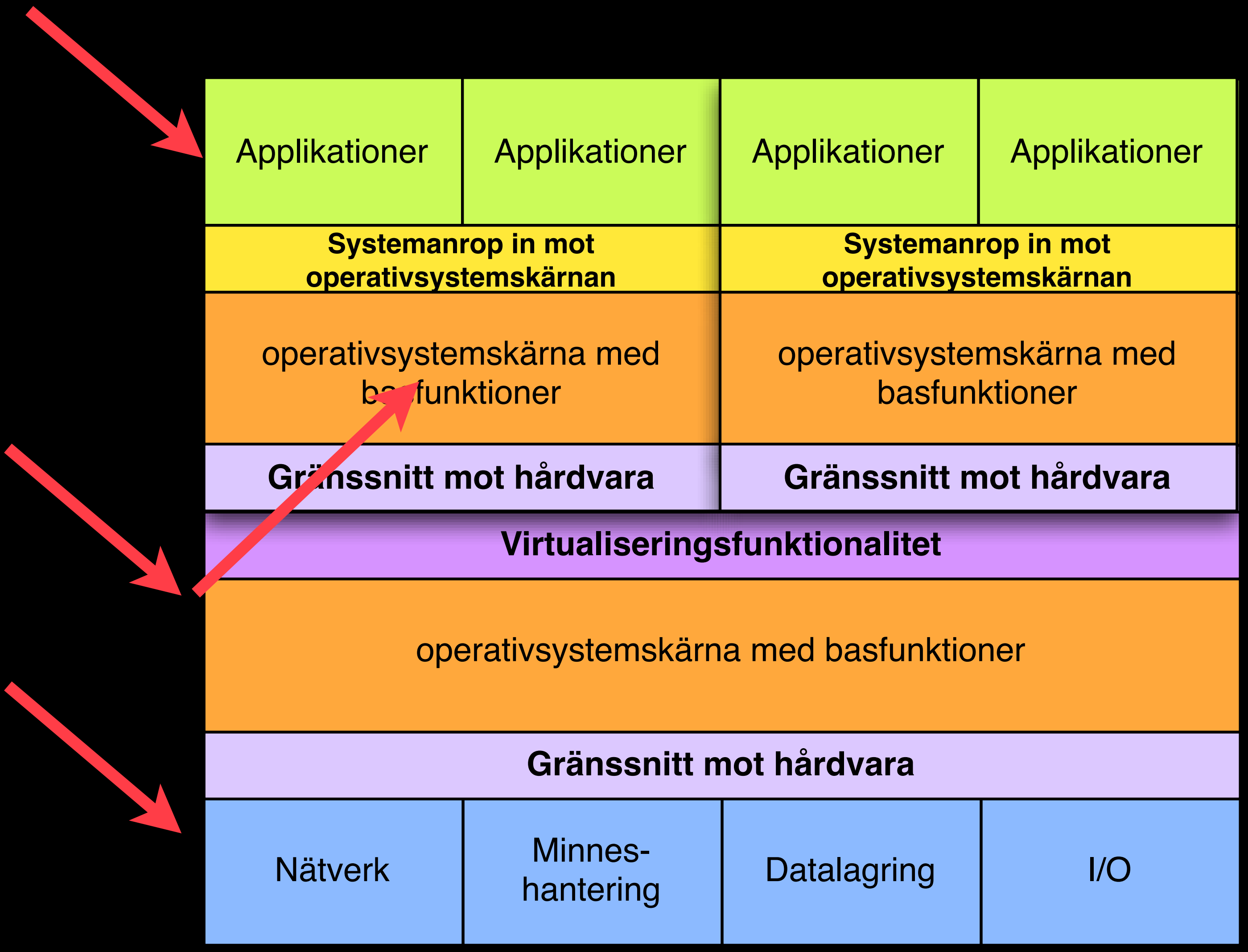
- Badly supported by Linux distributions
- Almost always require that one compile a custom kernel, which can have problems on it own
- Have support for RBAC through automatic rule generation

Virtualization and isolation

Isolation, separation and virtualization

- chroot (no virtualization, just isolation)
- jails
- user mode linux, uml
- Virtual machines: Vmware, MS Virtual Server, Containers
- Hardware partitioning: Sun LDOMs, IBM LPAR

Overview of virtualization


Applikationer
Applikationer
Applikationer
Applikationer
Systemanrop in mot operativsystemskärnan
Systemanrop in mot operativsystemskärnan
operativsystemskärna med basfunktioner
operativsystemskärna med basfunktioner
Gränssnitt mot hårdvara
Gränssnitt mot hårdvara
Virtualiseringsfunktionalitet
operativsystemskärna med basfunktioner
Gränssnitt mot hårdvara
Nätverk
Minnes-hantering
Datalagring
I/O

Pro’s and con’s with virtualization

- Isolation, and to have hardened and dedicated servers running specific services, are standard ways to minimize attack surface. Virtualization tools can help this
- Various types of OS supported or application supported *sandboxing* is good as a way to get defense-in-depth
- Its easy to believe that virtualization will automatically make things secure, and that there is no way to jump between guest os’, but exploits have shown this not hold true, e.g. cloudburst

http://www.immunityinc.com/documentation/cloudburst-vista.html

Some trends of interest

”real” OSs in new places

- Linux: phones & pads, embedded systems
- Windows: phones & pads, embedded systems
- MacOSX (iOS): phones, pads, embedded systems

- …and embedded systems are used in really critical places, like industry or utility companies.

”real” OSs in new places

- Gaming consoles: WII, Xbox, PS3 uses ”real OSs”
    - Alot focus on security - they know they will be attacked
    - Built in low-level hardware security

- Continous study of #fail
    - Hacked, hacked again, completely broken

”real” OSs in new places

```
220-<<<<<<>==< Haxed by A¦0n3 >==<>>>>>>
220- ¸,ø¤°°^°°¤ø,¸¸,ø¤°°^°°¤ø,¸¸,ø¤°°^°°¤ø,¸¸,ø¤°°^°°¤ø,¸
220-/
220-|    Welcome  to this fine str0
220-|    Today is: Thursday 12 January, 2006
220-|
220-|    Current througput: 0.000 Kb/sec
220-|    Space For Rent: 5858.57 Mb
220-|
220-|    Running: 0 days, 10 hours, 31 min. and 31 sec.
220-|    Users Connected : 1 Total : 15
220-|
220. ^°°¤ø,¸¸,ø¤°°^°°¤ø,¸¸,ø¤°°^°°¤ø,¸¸,ø¤°°^°°¤ø,¸¸,ø¤°°^
```

2006: Hacked “LeCroy” oscilloscope at CERN (running Win XP SP2)

”real” OSs in new places

- ”Same” attacks works in lots of places
- Not full OS’s on the new hosts, e.g. less protection makes it easier
- Always on, always reachable
- Device moves around, connects to unknown/rogue places, and easy to get physical access to them

On the high-level side

- When operating systems vendors and FOSS projects built-in better security and have more quality on their code, other low-hanging fruits are picked
  - Attacks are moving toward applications, e.g. Media players, flash framework, rich-text-document-format-viewers
  - ....and subsystems, e.g. database or integration software

On the low-level side

- More security critical functions are moved to hardware components
  - But the cat-and-mouse game continues, e.g. succesful hacking of TPM chips
  - bus snooping with JTAGs, goodfet, facedance, salea logic, etc

http://www.theregister.co.uk/2010/02/17/infineon_tpm_crack/

Summary

Important experience (1/3)

- OS-security is composed by several distinct parts. Failure in any of these parts with result in a security compromise
- There are many ways to circumvent security controls
- “Right tools for the right job”
  - E.g. to use MLS on a packet filtering firewall can be something that one should consider if that is the right thing to do...

Important experience (2/3)

- Old and new security functions often co-exists in a system
  - Can be complexed to understand the full consequense of a setup
  - Often hard to use
  - Is the counterpart to KISS

- RBAC is easy to understand, but hard to implement
  - 10 or 250 roles - wide or narrow privileges?
  - Often templates and profiler that is copied or referenced - changes to original definitions are not properly propagated, or in some cases errounesly propagated when they should not

Important experience (3/3)

- New trends challenge existing security concept or controls
  - E.g. attacks goes from server -> client
- Unfortunately, many good models are not properly used in practise since they are very had to understand, work with and to administer

Questions?

Additional information (1/3)

- Heap spraying as an attack method to do buffer overruns. Common attacks include Javascript based implementations to be used in web browsers, or to attack Adobe flash
  - http://en.wikipedia.org/wiki/Heap_spraying

Additional information (2/3)

- Security in Cloud Computing
  - US NIST government agency on cloud security
    - http://csrc.nist.gov/groups/SNS/cloud-computing/
  - Enisa report on Cloud security with recommendations
    - http://www.enisa.europa.eu/act/rm/files/deliverables/cloud-computing-risk-assessment/at_download/fullReport
  - http://www.infoworld.com/d/security-central/gartner-seven-cloud-computing-security-risks-853

Additional information (3/3)

- Contact me if you're interested in any of the work we have done with
  - Asterisk running contained with SELinux
  - FreeBSD w BIBA for different network services, e.g. bind
  - Free sandboxing, e.g. ironfox - our sandboxed firefox for MacOSX, ironadium - sandbox Adium, etc. See https://www.romab.com/ironsuite

Concepts mentioned during the class

- Kerkhoff’s principle - a ”rule of thumb” in crypto design - security must not rely on keeping the design/ machine/source code secret
- SDL - software development life cycle, MS model for developing ”more secure code”
- PRNG - Pseudo Random Number Generator. The rand() function is not entirely ”random”, and an external observer can recreate the series of number created by rand if the ”seed value” used by rand is somehow extracted/observed/etc

Concepts mentioned during the class

- Vulnerability - some property of a piece of software that can be manipulated or used in uninteded ways by an attacker, a.k.a. security bugs
- Exploits - usage of a vulnerability. Exploit code is software snippets to use the vulnerability
- Zero day exploits (0day). A previously ”unknown” (or at least publicly not published and wellknown) exploit for a security vulnerability

Concepts mentioned during the class

- Attack vector - Different paths to reach an vulnerability. One path might be closed by a vendor patch, but another might still be there, if the root cause is not identified and fixed.
- Reverse engineering. To re-create the original design by observing the final result, in computer science - to re-create some source code by examing a binary.

Tools mentioned during the class

- IDA pro - Disassembler
- Hexray - Decompiler
- Ollydbg, windbg - Other disassemblers
- Bindiff - Advanced tool from zynamics to compare binaries, with call graphs etc. Not same as built-in windows tool with same name.