Security and security controls in operating systems

A *quantitative* approach
2019-02-25

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 authorize 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

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

Some concepts and terms

Vulnerability

Exploit

0day exploit

Foreverday exploit

CVE

Stack smashing

Stack overflow

Heap overflow

Race conditions

Intro - *the basics*

- 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

Intro - *the basics*

- Bugs tend to get names (*heartbleed, ghost, shellshock*, etc) and logos

- also some bugs/vulnerabilities gets ”formal name”, i.e. CVE*, and a scoring CVSS**
  - e.g. CVE-2011-3172

* ”Common Vulnerabilities and Exposures;” https://cve.mitre.org/

** https://www.first.org/cvss/specification-document**

Some concepts and principles

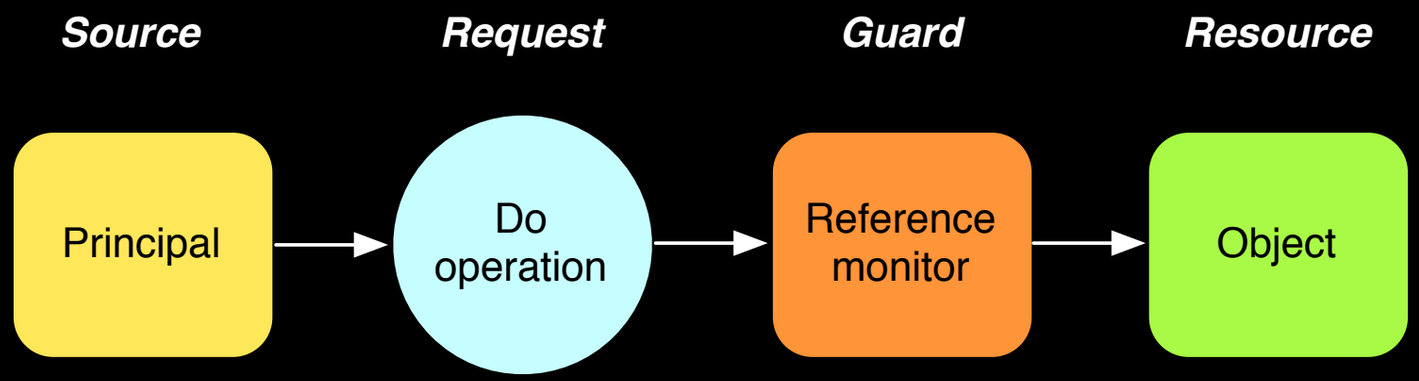
- Attack vector - Different paths to reach an vulnerability. One path might be closed by a vendor patch, but another might still be there, if the root cause is not identified and fixed.
- Reverse engineering - To re-create the original design by observing the final result, in computer science - to re-create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

Capabilities and requirements

Need	Description	Example
Protect a system resource	*Prohibit **malicious** or **unintentional** access to*	System tables, direct access to I/O-units,
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*	reference monitor
Separation of resources	*Physical, Logical, temporal or cryptographical separation*	separation in running time

Some important concept

- Reference monitor



- Trusted Computing Base, TCB

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

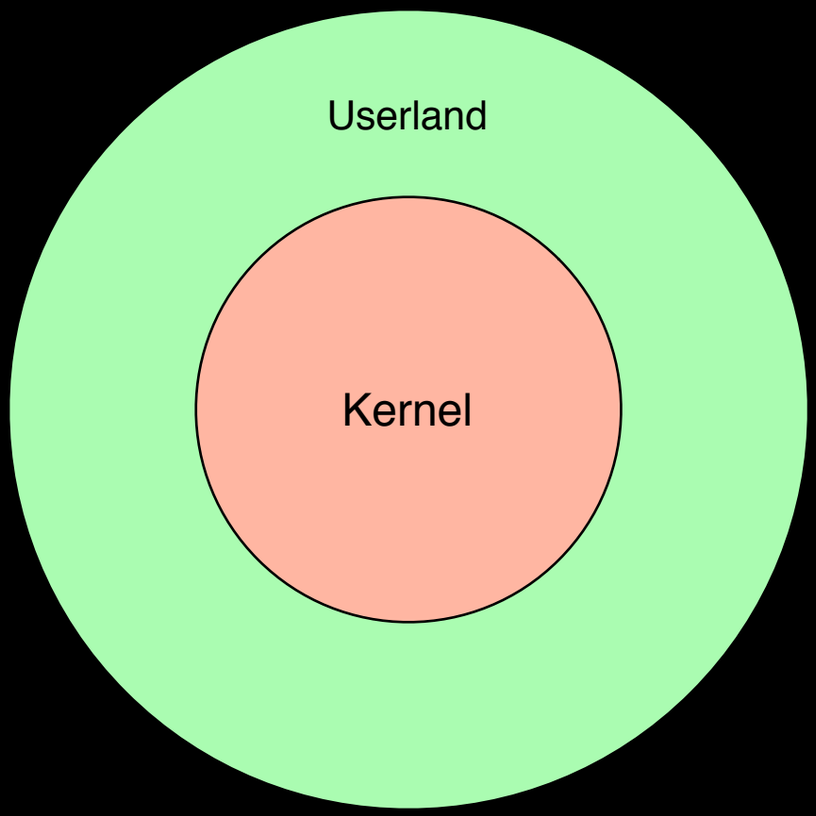
Principles for secure design*

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

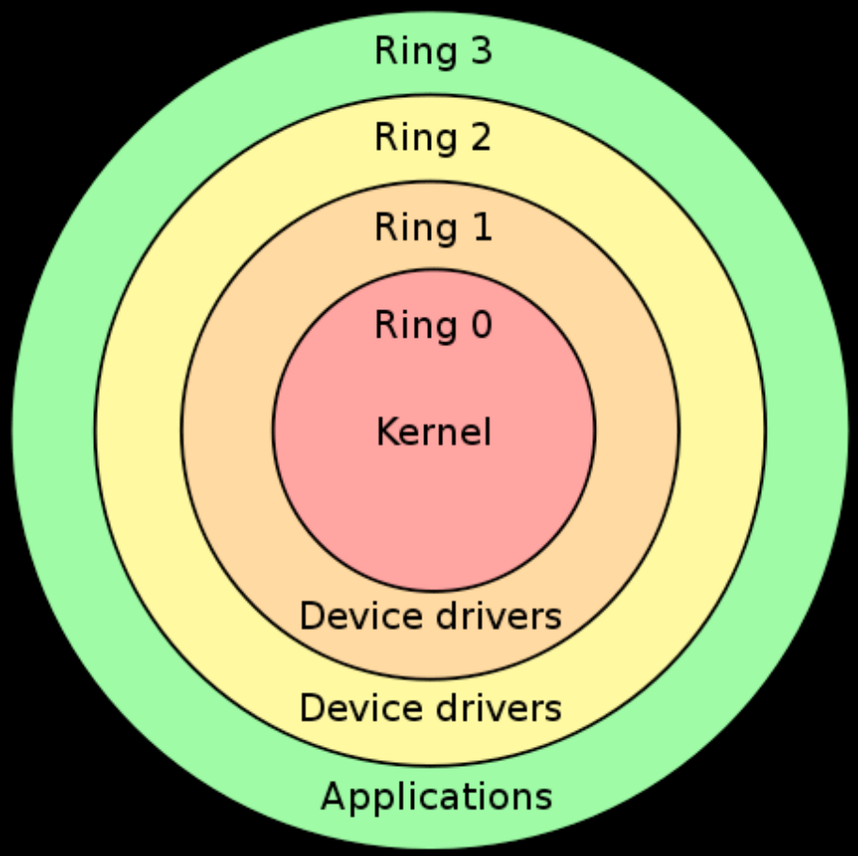
JEROME H. SALTZER et al *The Protection of Information in Computer Systems* http://www.cs.virginia.edu/~evans/cs551/saltzer/

The classical *ring model*

UNIX



x86

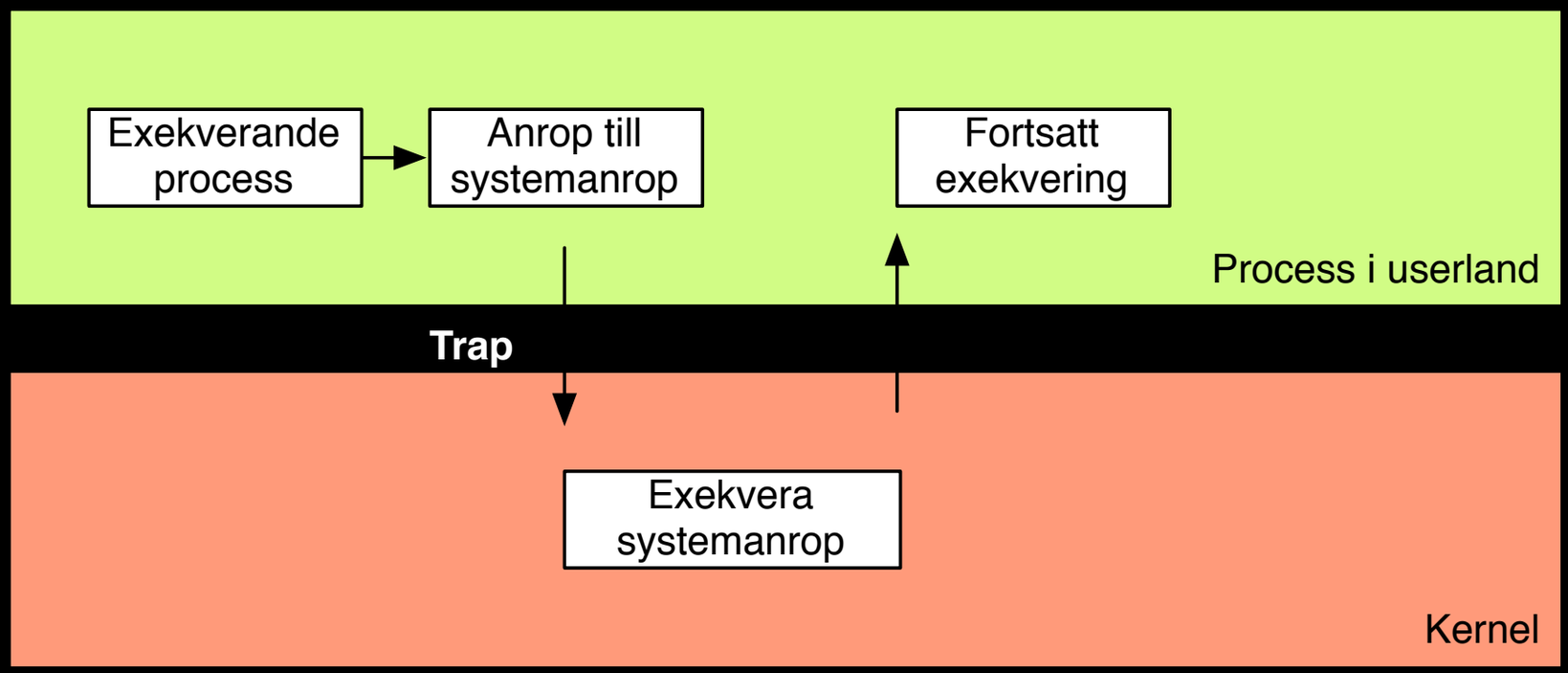


Least privileges

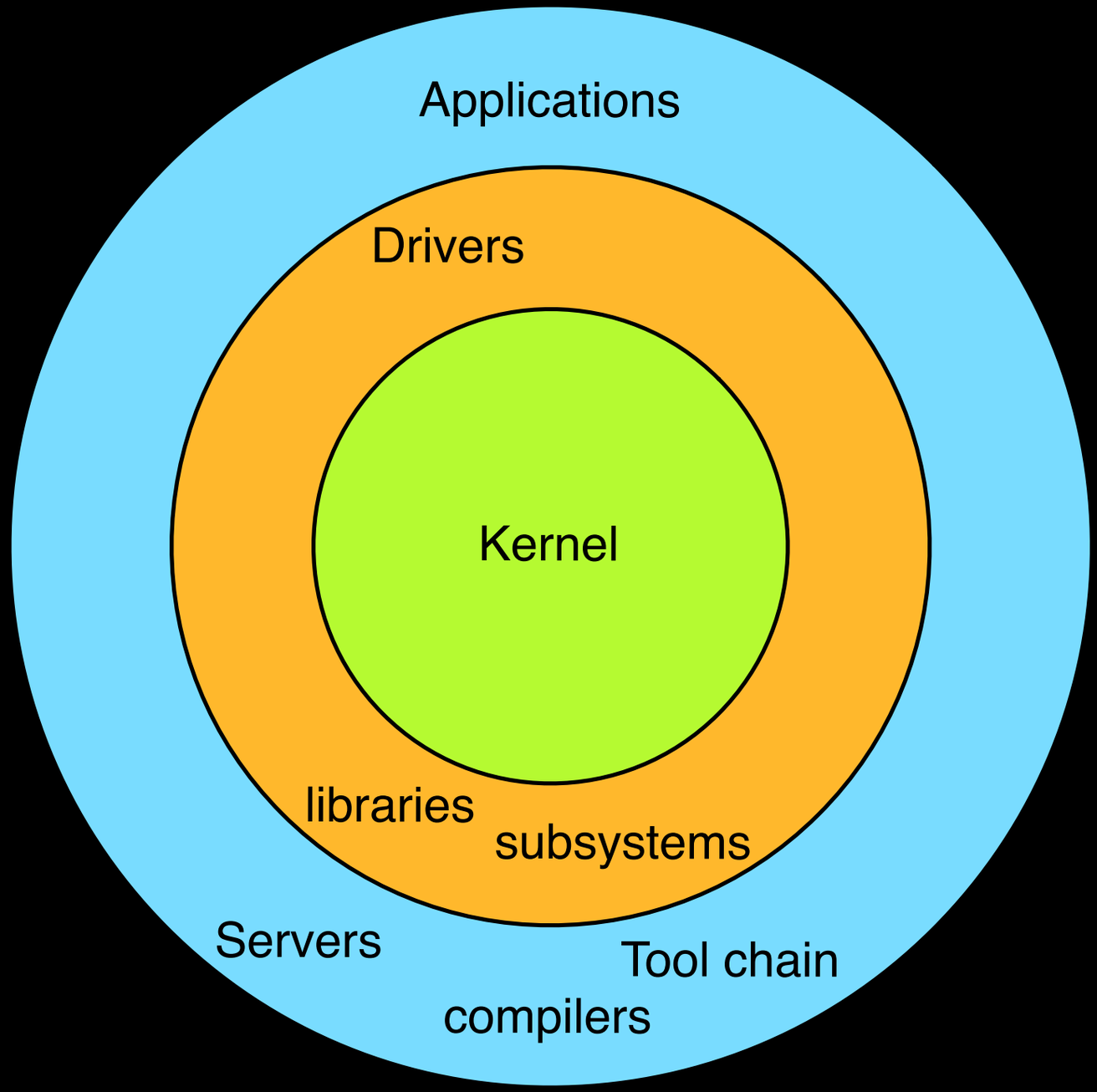
Highes privileges

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

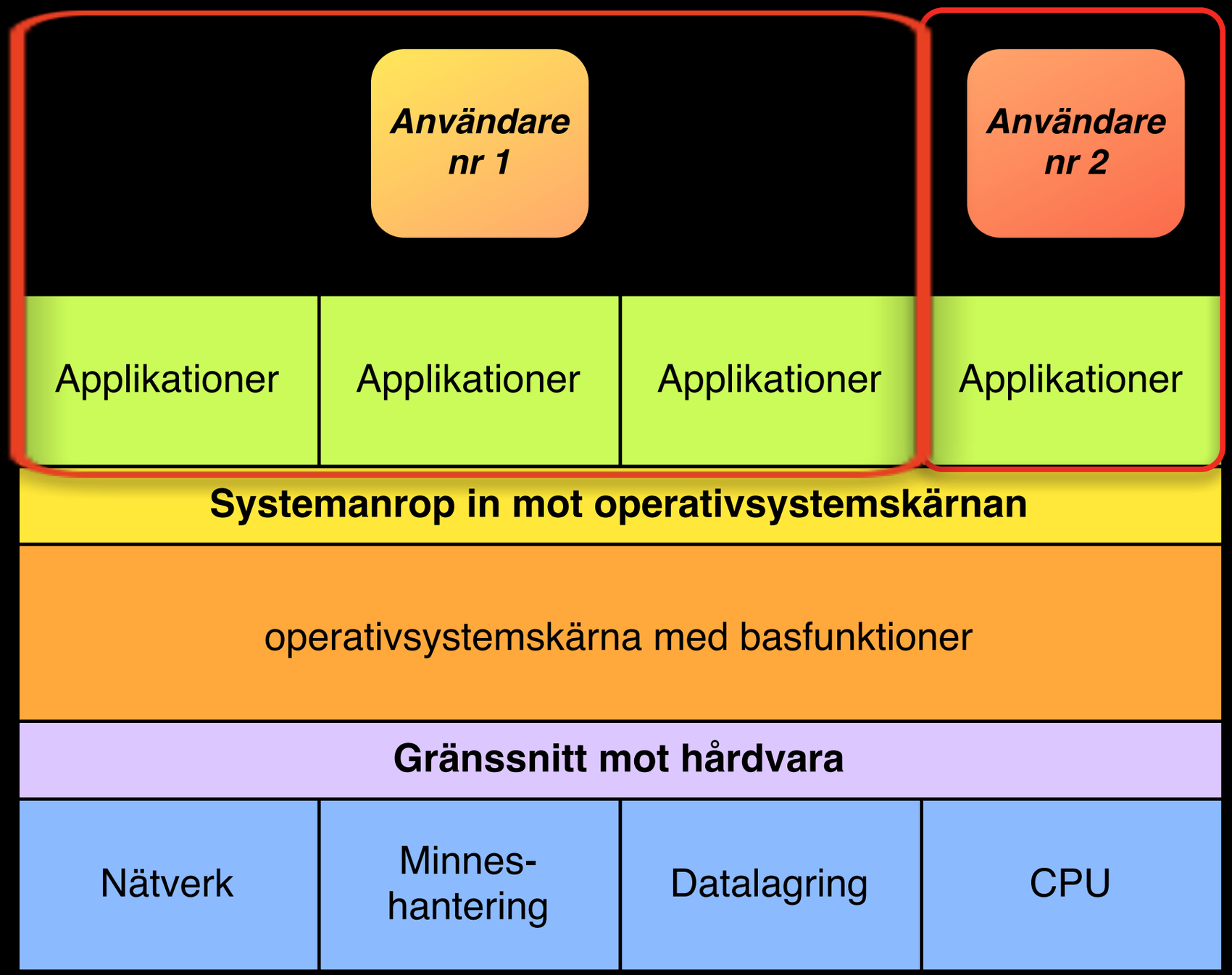
Interaction between application and OS



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 (1/5)

- 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 (2/5)

- Development methodologies
  - Open Source or Closed Source?

"Given enough eyeballs, all bugs are shallow"
- Linus' Law

- 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?

Comparing security in

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

2015	Windows 10	40-60

2018	Linux kernel 4.X	25

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

https://informationisbeautiful.net/visualizations/million-lines-of-code/

C y in Op 4/4)

https://www.cvedetails.com/top-50-products.php

CVE Details
The ultimate security vulnerability datasource

(e.g.: CVE-

Log In Register

Switch to https://
Home

Browse :
Vendors
Products
Vulnerabilities By Date
Vulnerabilities By Type

Reports :
CVSS Score Report
CVSS Score Distribution

Search :
Vendor Search
Product Search
Version Search
Vulnerability Search
By Microsoft References

Top 50 :
Vendors
Vendor Cvss Scores
Products
Product Cvss Scores
Versions

Other :
Microsoft Bulletins
Bugtraq Entries
CWE Definitions
About & Contact
Feedback
CVE Help
FAQ
Articles

External Links :
NVD Website
CWE Web Site

View CVE :
Go
(e.g.: CVE-2009-1234 or 2010-1234 or 20101234)

View BID :
Go
(e.g.: 12345)

Top 50 Products By Total Number Of "Distinct" Vulnerabilities

Go to year: 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2

	Product Name	Vendor Name	Product Type	Number of Vulnerabilities
1	Debian Linux	Debian	OS	2282
2	Linux Kernel	Linux	OS	2182
3	Android	Google	OS	2146
4	Mac Os X	Apple	OS	2107
5	Firefox	Mozilla	Application	1767
6	Chrome	Google	Application	1733
7	Iphone Os	Apple	OS	1514
8	Ubuntu Linux	Canonical	OS	1489
9	Windows Server 2008	Microsoft	OS	1187
10	Acrobat	Adobe	Application	1130
11	Flash Player	Adobe	Application	1070
12	Windows 7	Microsoft	OS	1047
13	Internet Explorer	Microsoft	Application	979
14	Safari	Apple	Application	968
15	Opensuse	Opensuse	OS	915
16	Acrobat Reader Dc	Adobe	Application	912
17	Acrobat Dc	Adobe	Application	912
18	Acrobat Reader	Adobe	Application	878
19	Thunderbird	Mozilla	Application	866
20	Windows Vista	Microsoft	OS	828
21	Windows Server 2012	Microsoft	OS	821
22	Enterprise Linux Desktop	Redhat	OS	785
23	Windows 10	Microsoft	OS	775
24	Windows 8.1	Microsoft	OS	757
25	Windows Xp	Microsoft	OS	740
26	Enterprise Linux Server	Redhat	OS	717
27	Seamonkey	Mozilla	Application	697
28	Enterprise Linux Workstation	Redhat	OS	684
29	Mac Os X Server	Apple	OS	640
30	Windows Rt 8.1	Microsoft	OS	631

Comparing security in Operating systems (5/5)

- 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**
Secure boot chain / Verified boot	*Make system startup sequence is secure*	Make sure that each step of boot is cryptographically signed to ensure code integrity, e.g. **UEFI vs BIOS**
Encryption	*Protection against eavesdropping or unauthorized access*	network traffic, file content, disk partitions, memory pages, swap files/ page 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**
Random numbers	*Make a resource non-deterministic*	File names, proccess ID,'s port numbers, sesssion keys, session id's, transaction numbers, DNS query ID's, execution time & timing
Constant numbers	*Make a resource non-deterministic*	execution time & timing

General example of control principles

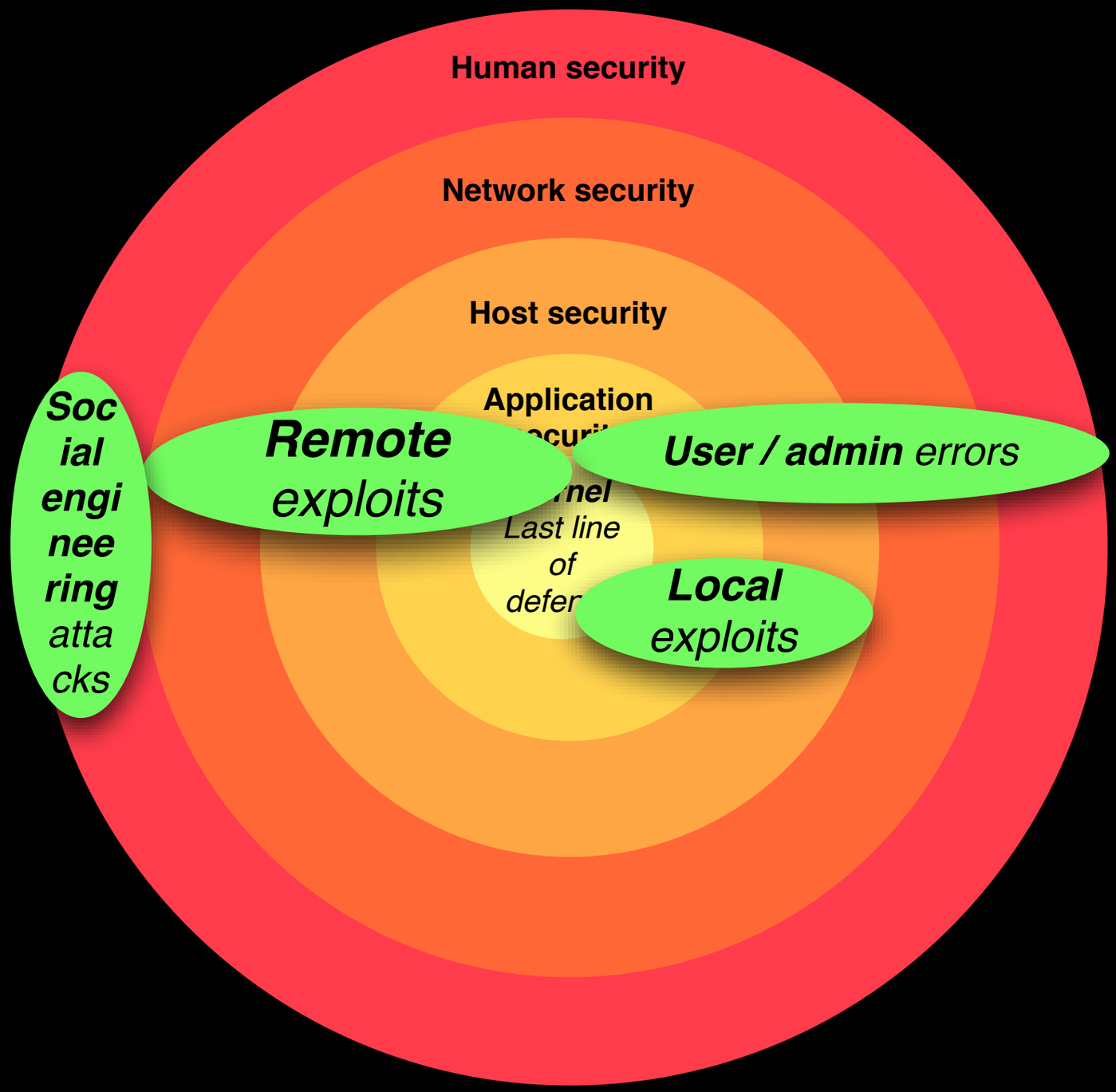
Security controls	**Description**	**Example**
Compiler generated airbag - canary	*Make sure buffer overflows dont gets undetected*	ProPolice, VisualStudio /GS
ASLR	*Randomize addresses used by applications. Make sure its hard to write code that knows of addresses. Where did that lib go?*	Android >4.0, iOS > 4.3, Windows >Vista, OpenBSD/NetNSD, Linux >2.6.12, MacOSX >10.5, Solaris >11.1, etc
KASLR	*Randomize addresses used by kernel*	Windows Vista, NetBSD, Linux >3.14, MacOSX 10.8, etc

General example of control principles

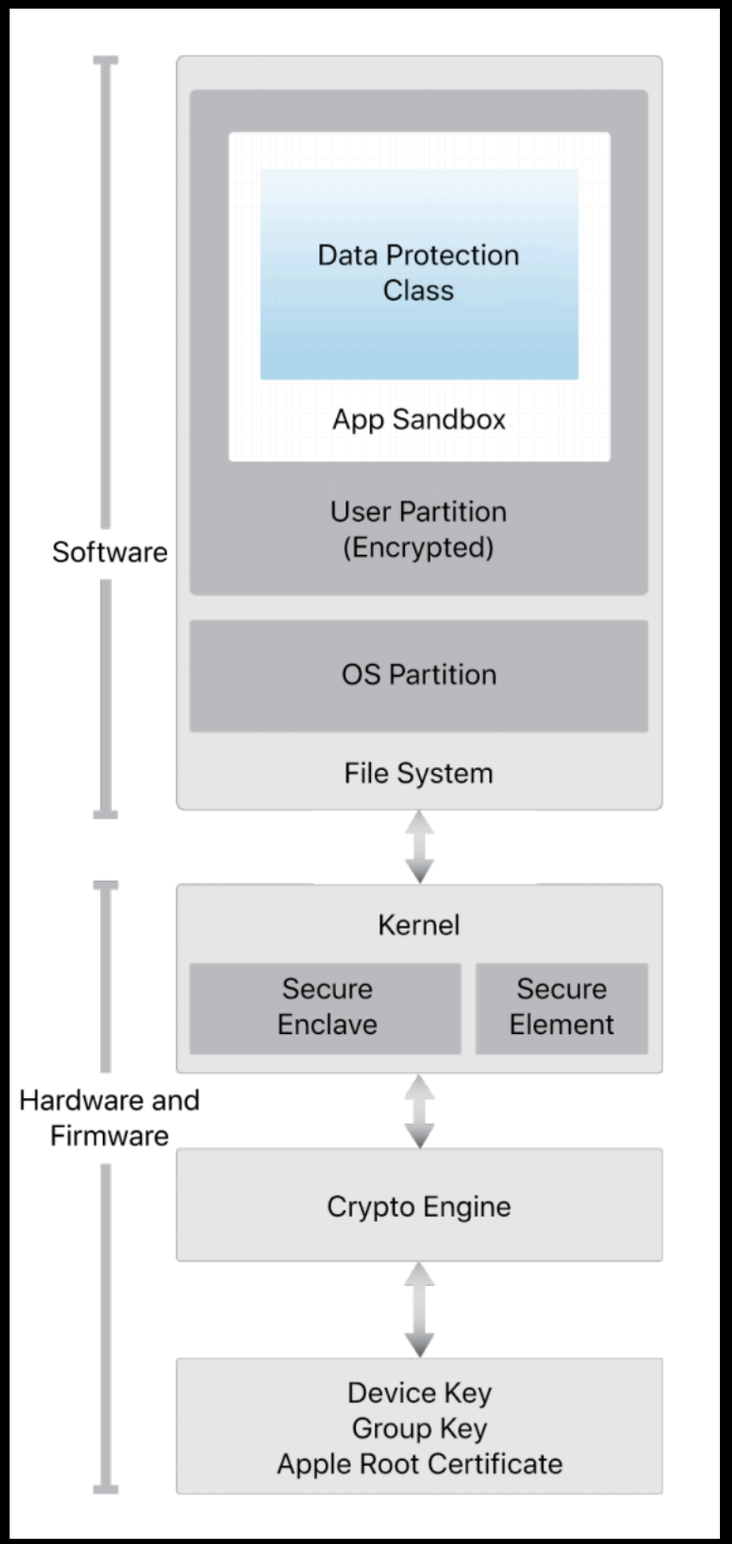
Security controls	Description	Example
DEP, NX, W^X	*Make sure memory is not executable*	IE on Windows Vista, Android >2.3, FreeBSD > 5.3, OpenBSD, Linux >2.6.8, MacOSX >10.5, etc
Scrubing, zeroing	*Make sure that old data areas are cleaned before usage or returned to system*	memory, file systems, VM system

Examples of vulnerabilities and attacks

Where do attacks occur?



Apple iOS device security



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

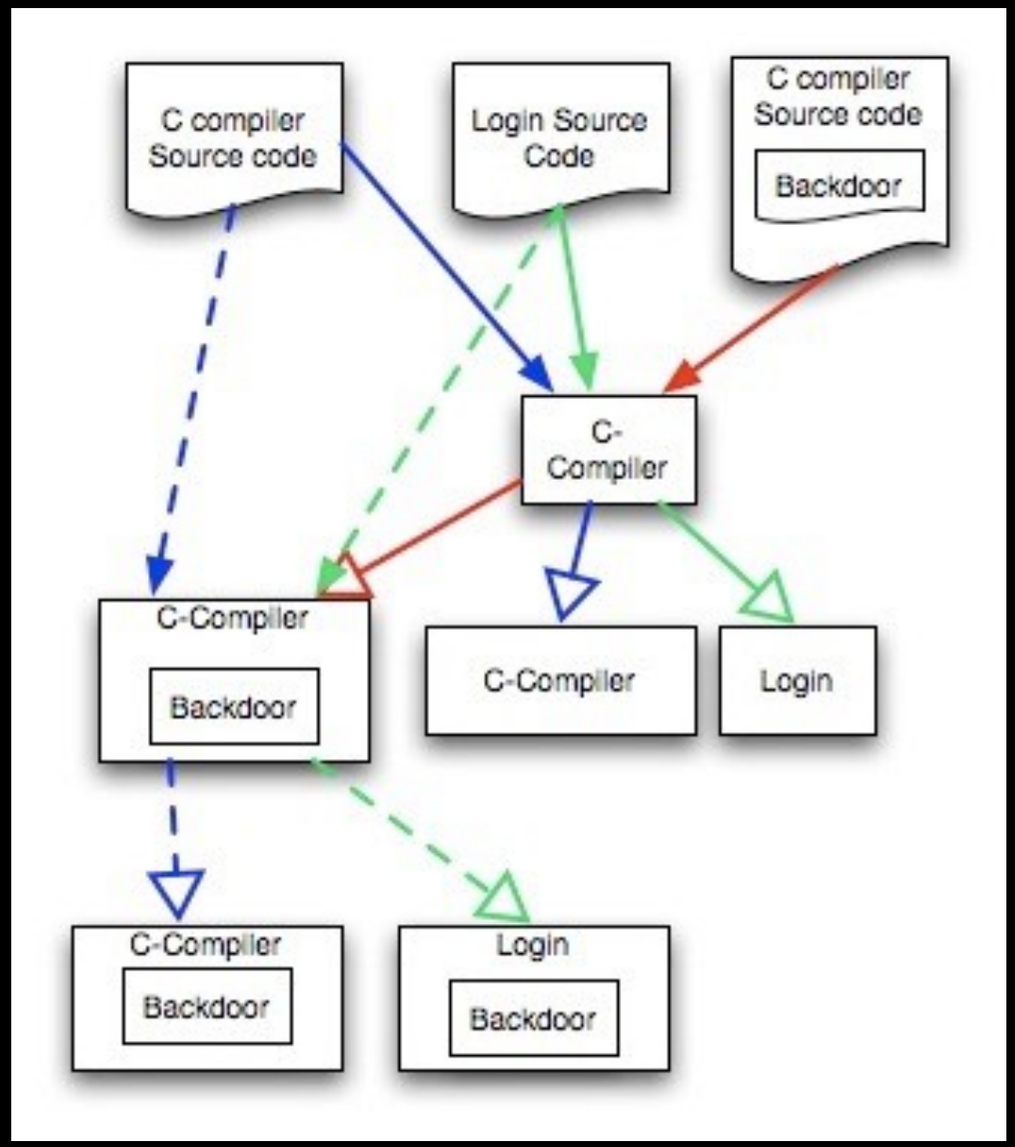
Manipulated user files

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

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*

Example of 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 ;-)

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, evil devices*)
  - UEFI attacks via Thunderbolt (*thunderstruck attack*)
  - Using JTAG interfaces to snoop & manipulate bus

Example of attacks

- Removal of, or direct attachment to, physical memory chips (*cold boot attacks*)

Example of attacks: *cold boot attacks*

F-secure ”The Chilling Reality of Cold Boot Attacks” https://www.youtube.com/watch?v=E6gzVVjW4yY

Example of attacks: *PCILeech*

Attacking
UEFI Runtime Services
and Linux

Ulf Frisk - ”Attacking UEFI Runtime Services and Linux” https://www.youtube.com/watch?v=PiUVRHYTDUg

Example of attacks: *HW implants*

(TS//SI//NF) Left: Intercepted packages are opened carefully; Right: A "load station" implants a beacon

https://arstechnica.com/tech-policy/2014/05/photos-of-an-nsa-upgrade-factory-show-cisco-router-getting-implant/

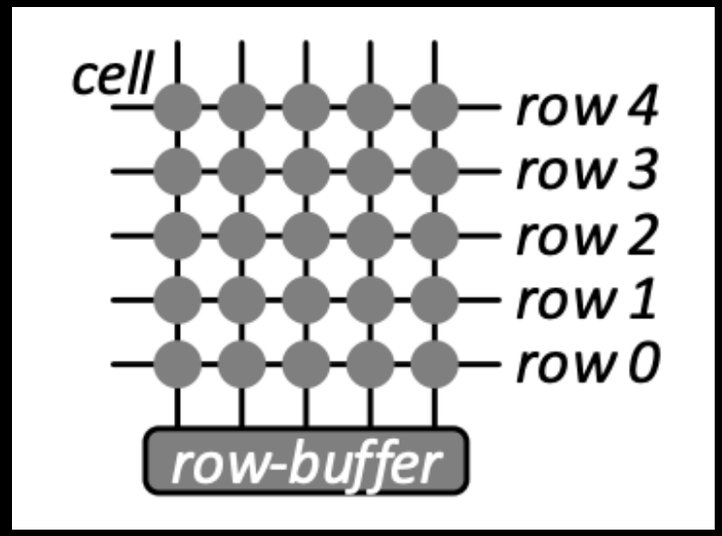
Advanced attacks

- *Rowhammer**
  - *Based on an unintended side effect in dynamic random-access memory (DRAM) that causes memory cells to leak their charges and interact electrically between themselves, possibly altering the contents of nearby memory rows that were not addressed in the original memory access.*
- *Flipping bits without accessing them*

"Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors"
— Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, Onur Mutlu, at CMU

Advanced attacks

- *Rowhammer**
  - *Method of reading writing memory cells so that memory cells in adjacent rows become changed*
  - *This circumvention of the isolation between DRAM memory cells*
    - *Memory leak == information leak*
- *Have been used to* ***Gain Kernel Privileges***
- *Can be used to attack* ***Virtual Machines***



* Kim et al ” Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors” https://users.ece.cmu.edu/~yoonguk/papers/kim-isca14.pdf

Advanced attacks

- *Rowhammer*
- *Have been implemented in JavaScript and runned in a browser*
- *Modern variants* have been used to* ***defeat ECC memory***

* ”Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks” https://cs.vu.nl/~lcr220/ecc/ecc-rh-paper-eccploit-press-preprint.pdf

Advanced attacks

- *Meltdown* & Spectre***
- *Low-level **cache** attacks, allow malicious READs*
- *Meltdown breaks isolation between **user land and kernel***
- *Spectre breaks isolation between applications in user land*

https://meltdownattack.com/

* Lipp et al ”Meltdown: Reading Kernel Memory from User Space” https://meltdownattack.com/meltdown.pdf

** Kocker et al ”Spectre Attacks: Exploiting Speculative Execution” https://spectreattack.com/spectre.pdf

Advanced attacks

- *Meltdown & Spectre*
  - *All modern CPUs are vulnerable (x86, AMD, ARM) in various degrees*

	Method	Attack	Spectre-PHT	Spectre-BTB	Spectre-RSB	Spectre-STL
Intel	same-address-space	in-place	● [52, 50]	★	● [62]	● [32]
		out-of-place	★	● [18]	● [62, 54]	○
	cross-address-space	in-place	★	● [52, 18]	● [62, 54]	○
		out-of-place	★	● [52]	● [54]	○
ARM	same-address-space	in-place	● [52, 50]	★	● [6]	● [6]
		out-of-place	★	☆	● [6]	○
	cross-address-space	in-place	★	● [6, 52]	☆	○
		out-of-place	★	☆	☆	○
AMD	same-address-space	in-place	● [52]	★	★	● [32]
		out-of-place	★	☆	★	○
	cross-address-space	in-place	★	● [52]	★	○
		out-of-place	★	☆	★	○

Symbols indicate whether an attack is possible and known (●), not possible and known (○), possible and previously unknown or not shown (★), or tested and did not work and previously unknown or not shown (☆). All tests performed with no defenses enabled.

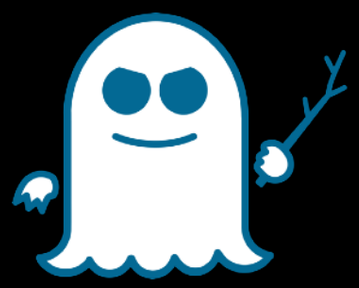
Vendor \ Attack	Meltdown-US [59]	Meltdown-P [90, 93]	Meltdown-GP [10, 40]	Meltdown-NM [83]	Meltdown-RW [50]	Meltdown-PK	Meltdown-BR	Meltdown-DE	Meltdown-AC	Meltdown-UD	Meltdown-SS	Meltdown-XD	Meltdown-SM
Intel	●	●	●	●	●	★	★	☆	☆	☆	☆	☆	☆
ARM	●	○	●	_	●	_	_	☆	☆	☆	_	☆	☆
AMD	○	○	○	○	○	_	★	☆	☆	☆	☆	☆	☆

Symbols indicate whether at least one CPU model is vulnerable (filled) vs. no CPU is known to be vulnerable (empty). Glossary: reproduced (● vs. ○), first showed in this paper (★ vs. ☆), not applicable (_). All tests performed without defenses enabled.

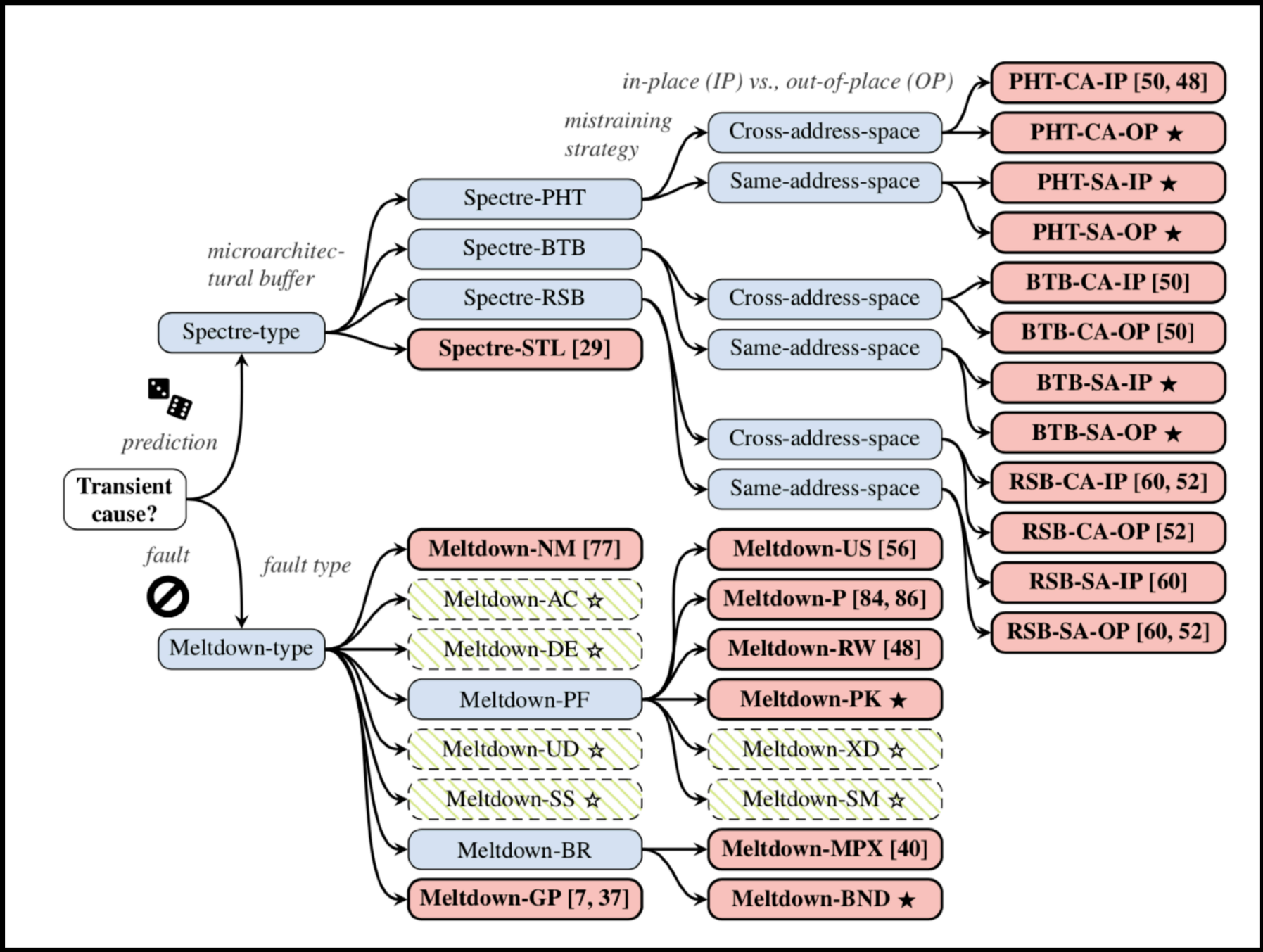
* Canello et al ”A Systematic Evaluation of Transient Execution Attacks and Defenses” https://arxiv.org/pdf/1811.05441.pdf

Advanced attacks

- *Meltdown & Spectre*
  - *work on personal computers, mobile devices, and in the cloud*
  - *Works on Windows, Linux, Android, etc*
  - *Works on containers: docker, LXC, OpenVZ etc*



Advanced attacks



* Canello et al ”A Systematic Evaluation of Transient Execution Attacks and Defenses” https://arxiv.org/pdf/1811.05441.pdf

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, *chang*e 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

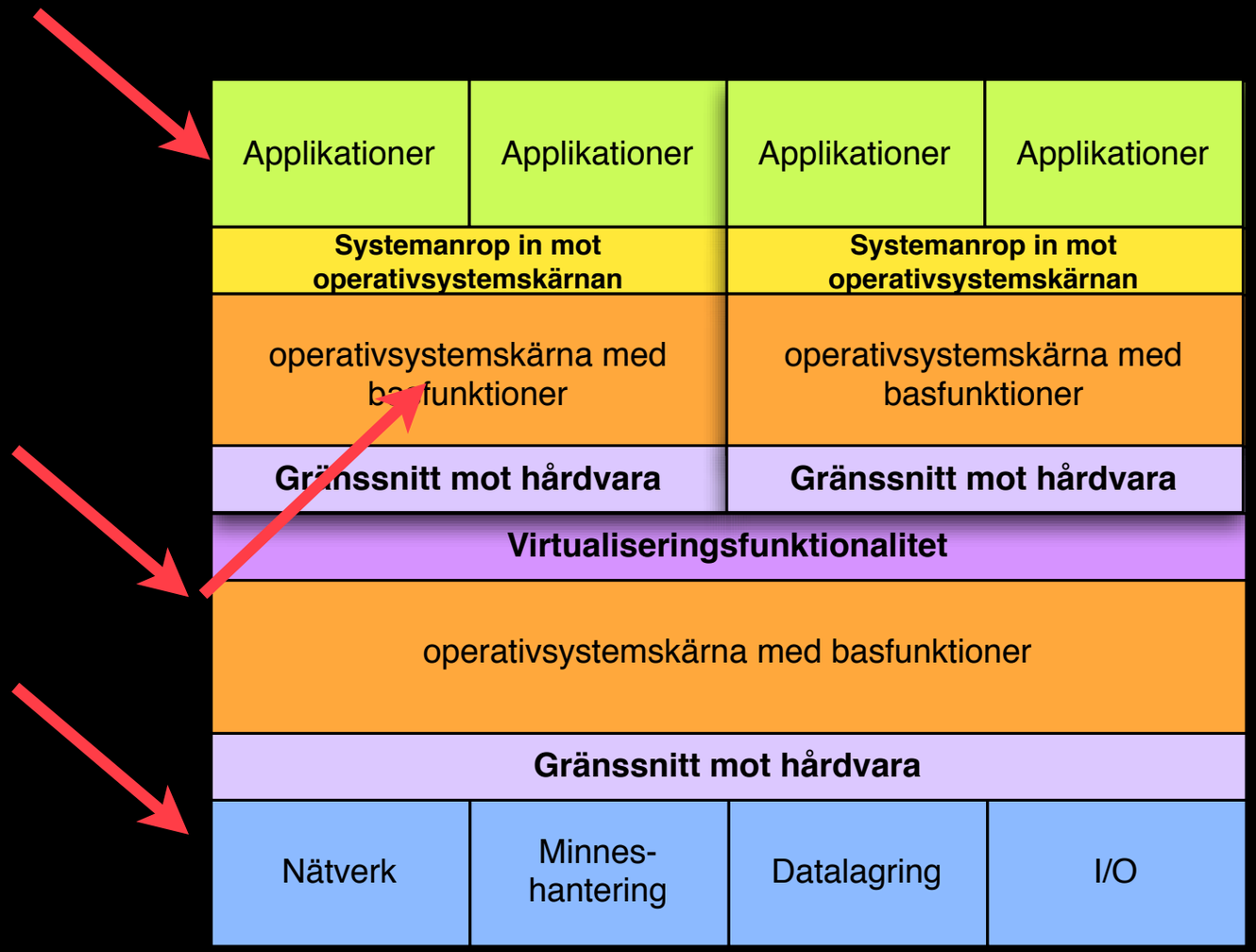
- Badly supported by Linux distributions
- Almost always require that one compile a custom kernel, which can have problems on it own
- Have support for RBAC through automatic rule generation

Virtualization and isolation

Isolation, separation and virtualization

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

Overview of virtualization



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
- 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

Sandboxing

- Various types of OS supported or application supported ***sandboxing*** is good as a way to get defense-in-depth
- Create temporary execution environments for certain tasks
  - test of exe files to lure out malicious code execution
  - perform certain tasks that is more prone to attacks
  - perform certain tasks that is more sensitive

Pro's and con's with virtualization

- Some sandbox and isolation technologies are not complete virutalization or separation
  - E.g. share name space (processes, file system, etc)
  - Share operating system kernel
  - Share drivers