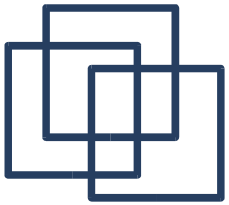
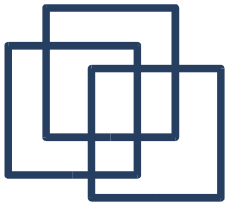


Umbrella Security Framework

Emmanuel Fleury
fleury@cs.aau.dk

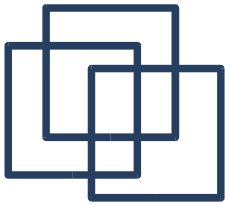
Kristian Sørensen
ks@cs.aau.dk

Aalborg University
Department of Computer Science
Denmark

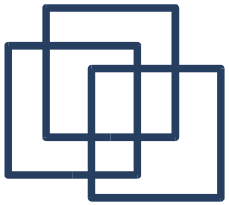


Outline

- Motivations
- Umbrella Security Framework
- Process-Based Access Control (PBAC)
- Digitally Signed Binaries (DSB)
- Conclusions & Further Work
- Live Demonstration (Kristian Sørensen)



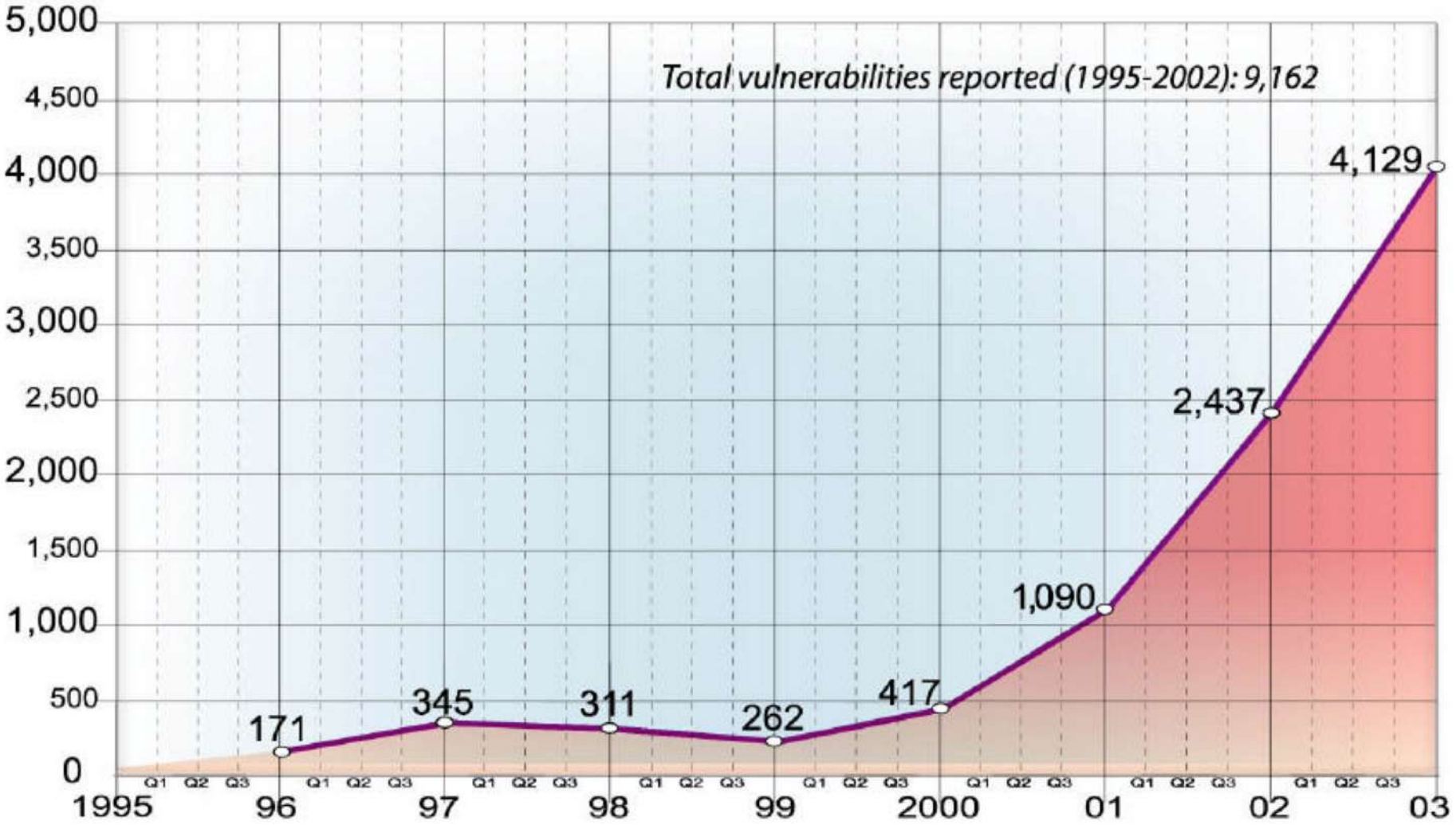
Motivations



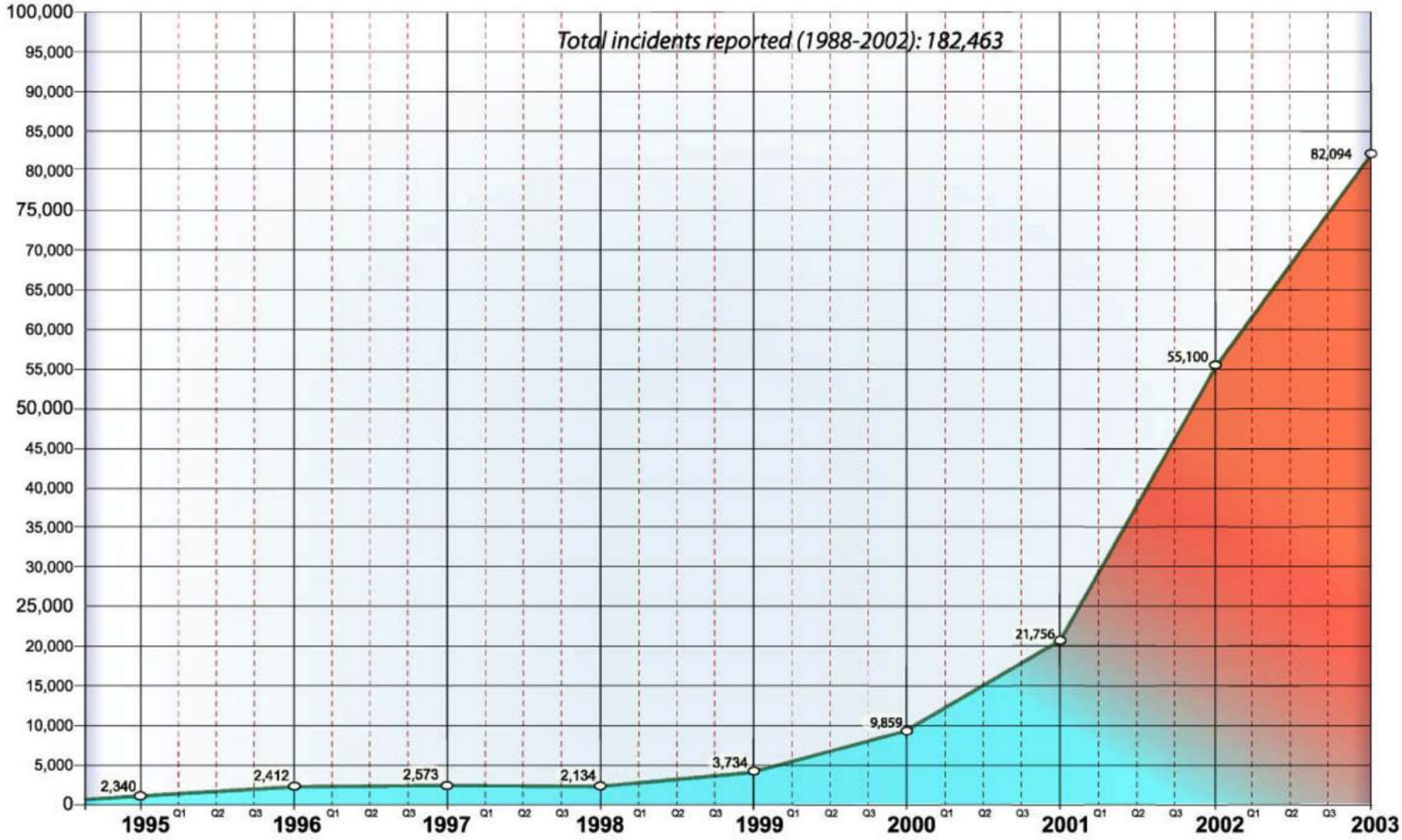
Security Threats over Internet

- **Complexity** of Internet, Protocols and Applications are all **increasing**
- Source Code isn't required to **find flaws**
  (i.e. Microsoft Windows & Internet Explorer)
- **Attacker tools** are:
  - Increasingly sophisticated
  - Easy to use
  - Designed to perform large scale attacks

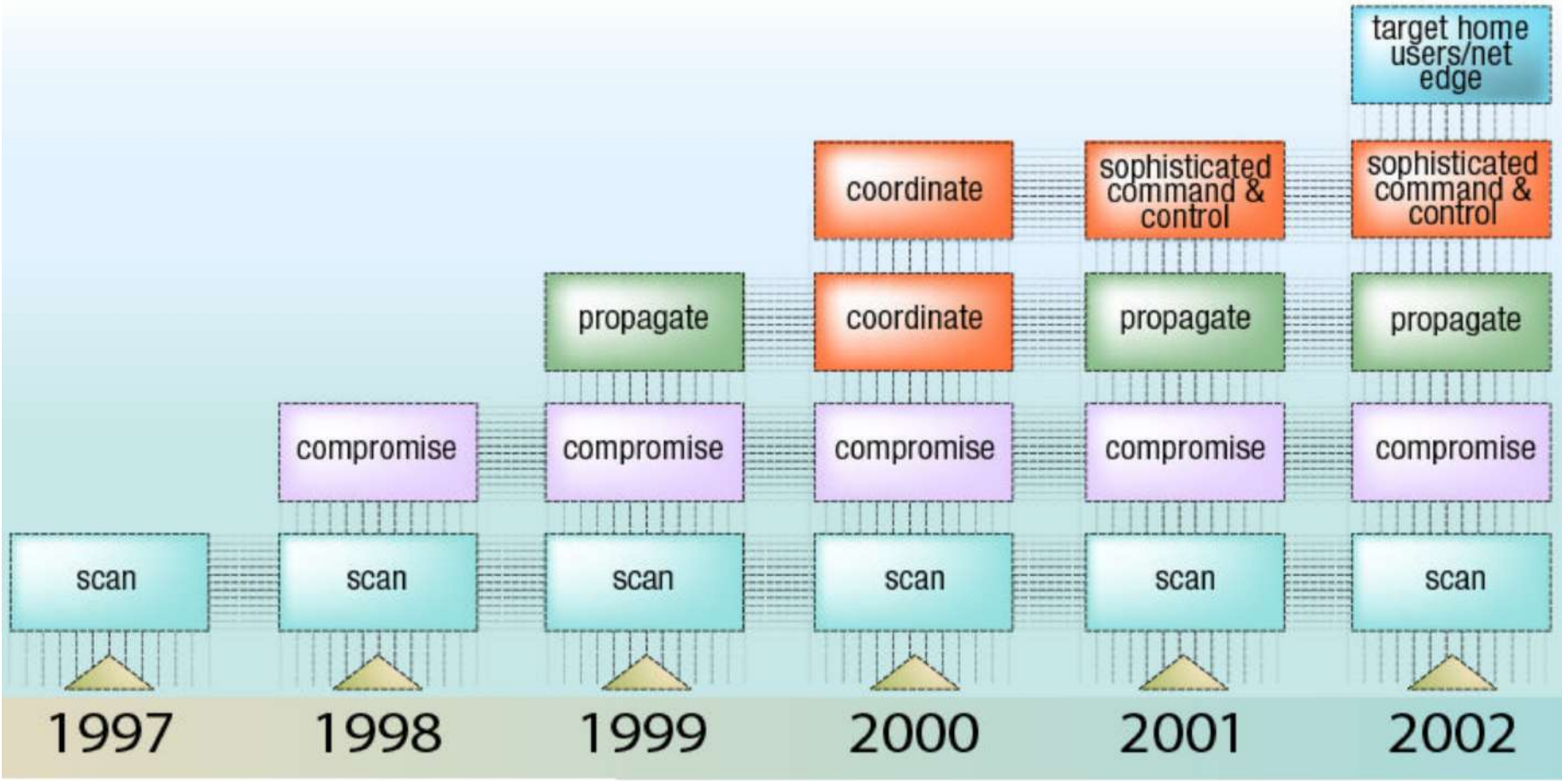
Total Vulnerabilities Reported



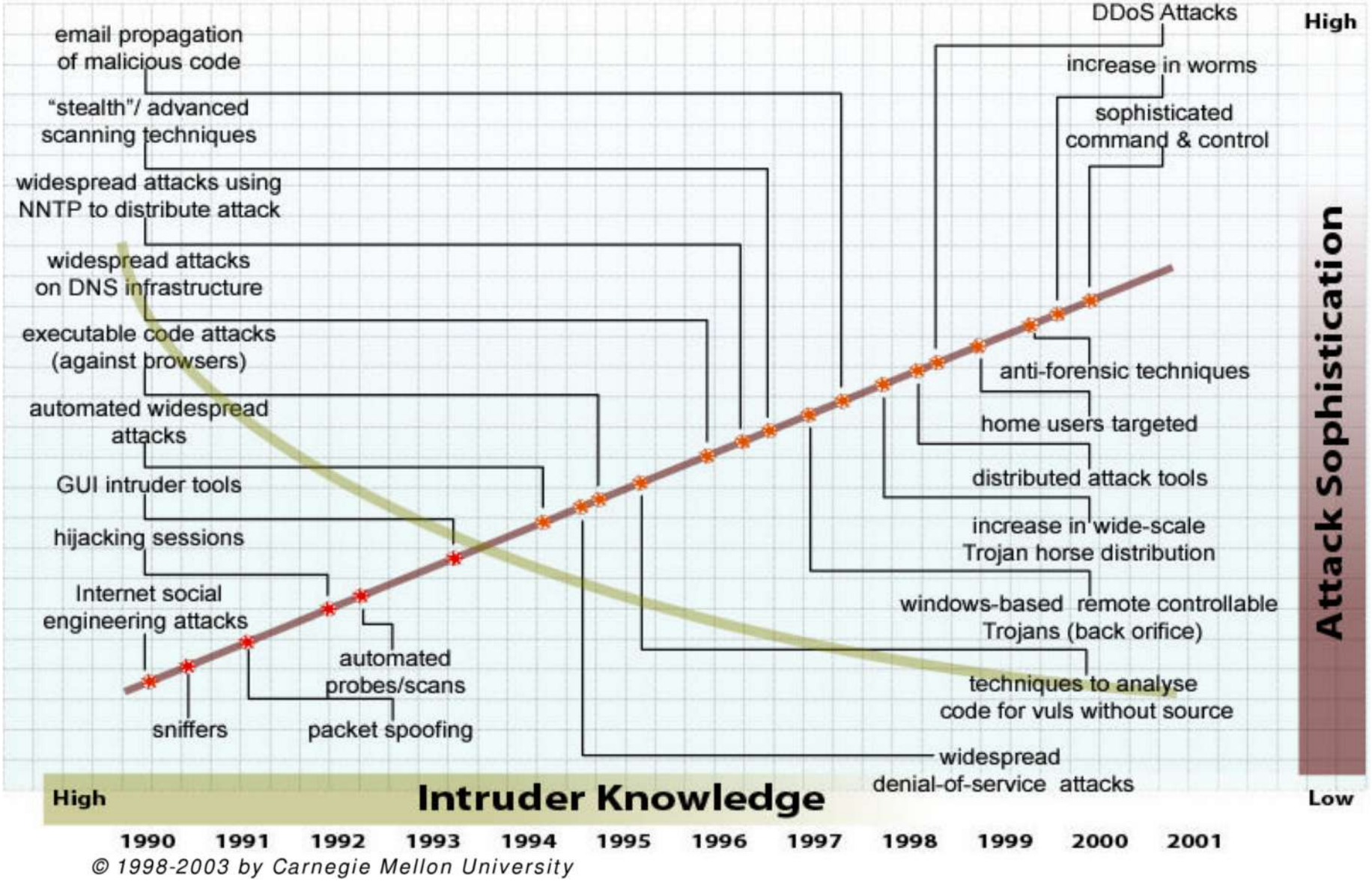
Total Incidents Reported

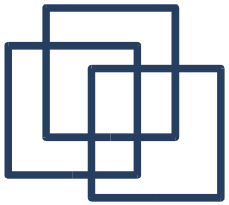


Attacker Technology



Attacks Evolution





Future Of the Security Over Internet ?

- Expertise of Attackers is increasing
- Sophistication of Attacks is increa
- Ergonomy of Attack To
- Experti on Internet is decreasing
- ation of Counter-measures is increasing
- Ergonomy of Counter-measure Tools is stable

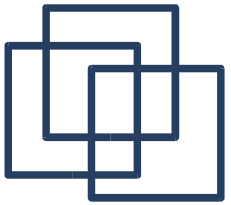
Highly Explosive Situation !!!

Threats over Consumer Electronics

What does all these things have in common ???

So, they are sensitive to the same risks as a real computer !!!

Caribe: A Virus for Mobiles



Caribe: Technical Facts

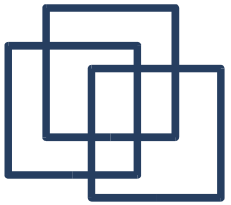
- Released in June 2004 by the group 29A
- Just a Proof of Concept
- Innovations ?

None !

- Flaws Exploited ?

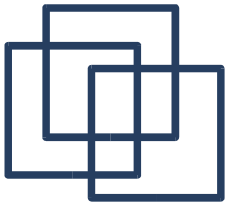
None !

Where is the Challenge, then ???



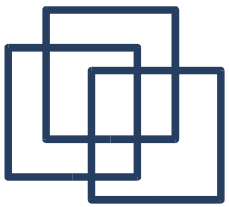
Why is it Challenging ?

- Need Specific Hardware
- Need Highly Technical Documentations
- Need Reverse-Engineering Work
- Need Cross-compilers
- There Are No Standards
- No Open Community of Coders
- …

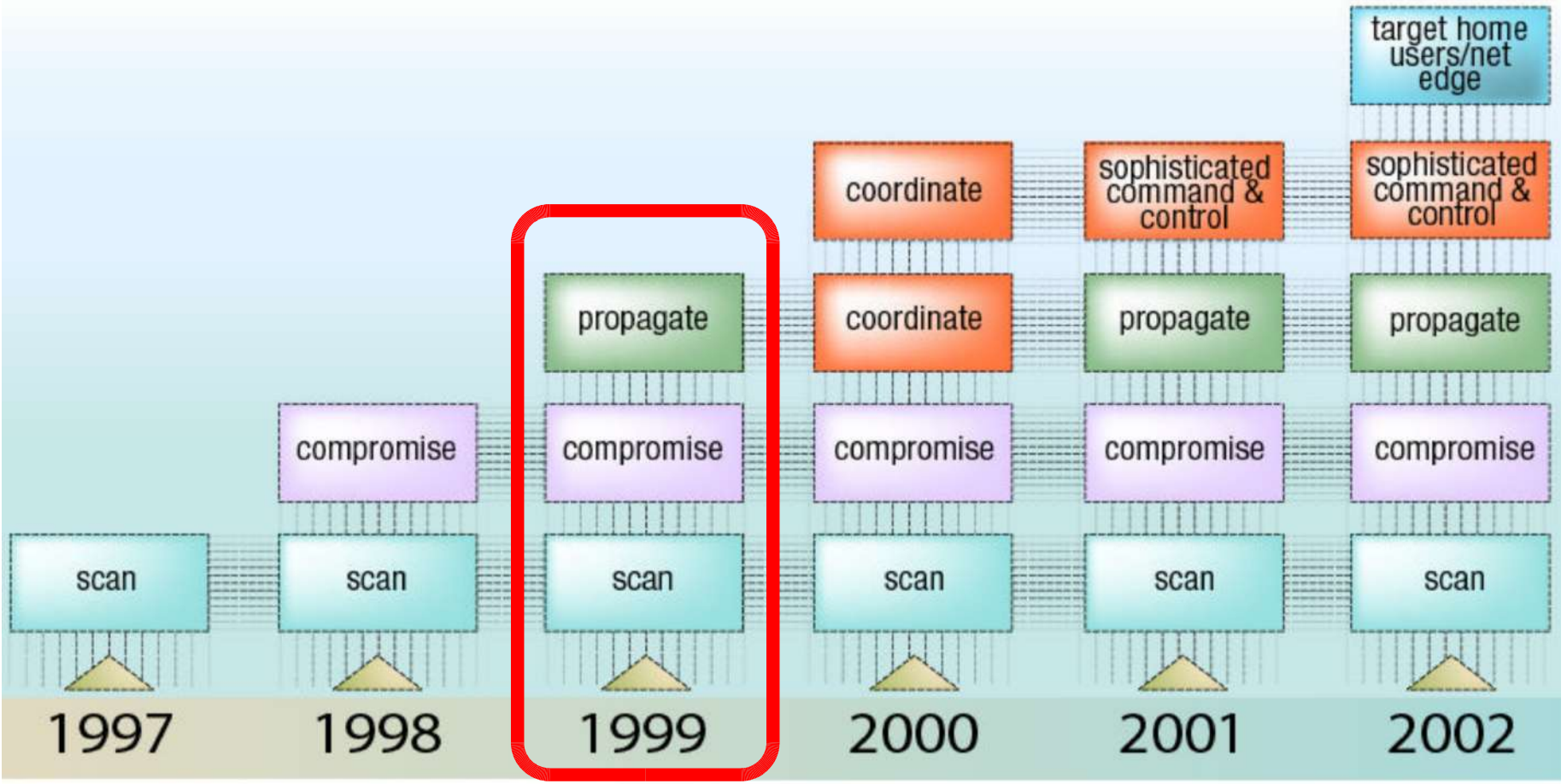


In the Future ?

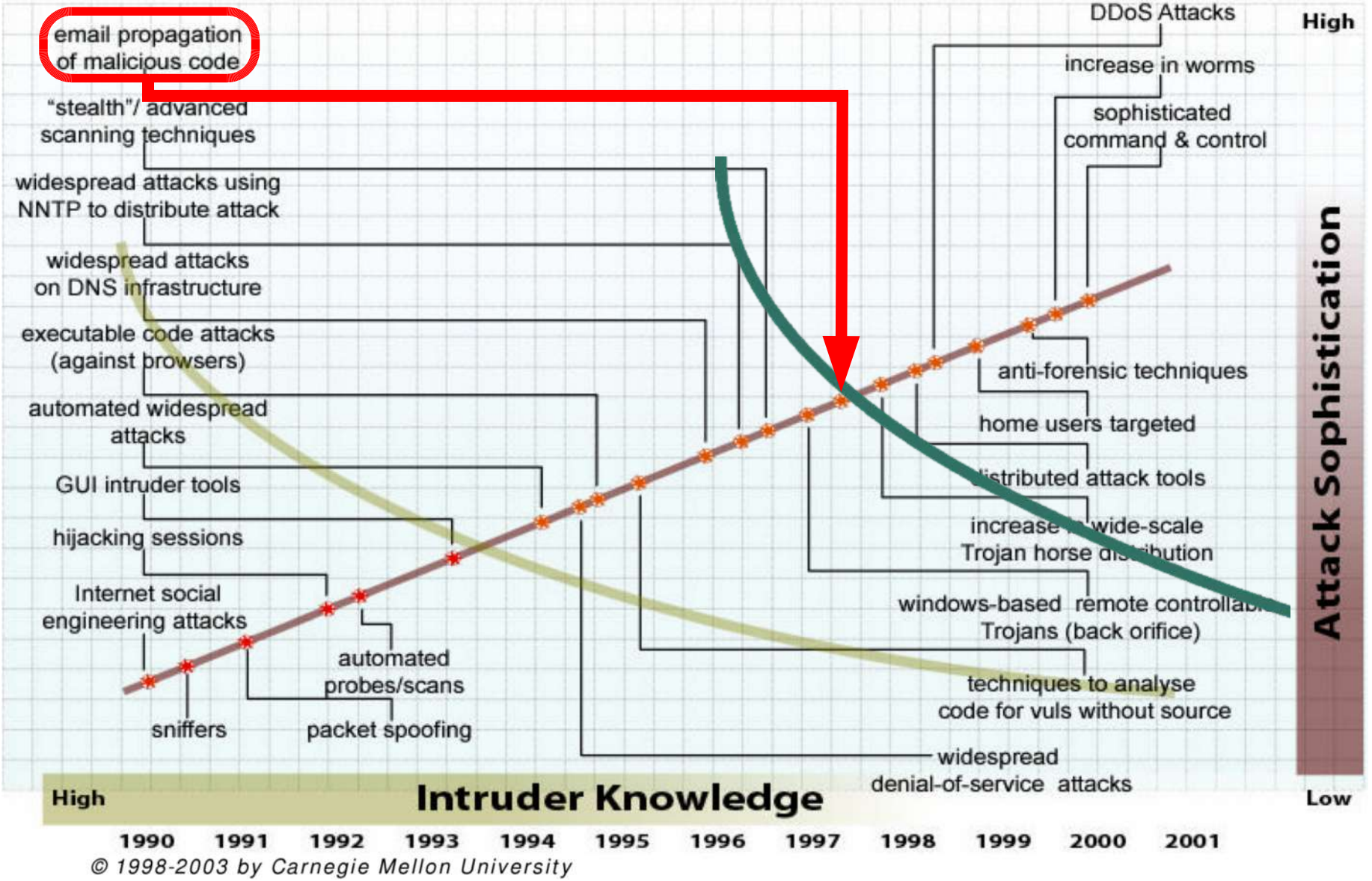
- Hardware will be Standardized
- Documentation will be Simplified
- Reverse-Engineering will be Published
- Cross-compilers will become easy to use
- Standards will come (have to !)
- Open Community of Coders will be formed
- ...

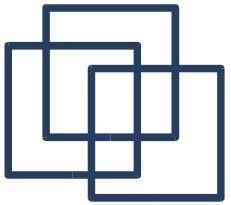


Where Are We Now ? (1)



Where Are We Now ? (2)

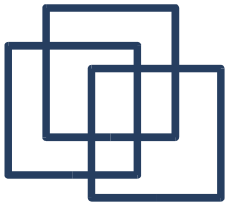




Future Of the Security Over CE ?

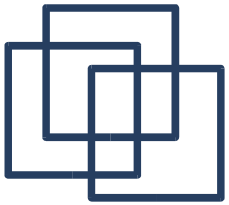
- Expertise of Attackers will increase
- Sophistication of Attacks will incr
- Ergonomy of Attack To
- People
hone are computer illiterates
- ion of Counter-measures will increase
- Ergonomy of Counter-measure Tools will be stable

Highly Explosive Situation !!!



What OS would suit ?

- Multi-users (Fine Grained Access Control)
- Multi-tasking (Advanced Scheduler)
- Standardization of the Programming Interface (POSIX-like ?)
- Separation of Kernel and User memory
- Strictly Follow the Standards For Network Protocols
- **Hey !!! It's UNIX !!!**



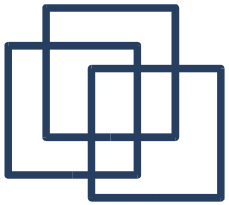
Final Thought ?

You have to understand what the primary objective of an OS is:

To create a virtual environment that is simple and sane to program against....

Have you learned nothing at all from DOS and Windows ?

-- Linus Torvalds



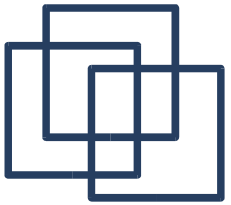
Umbrella Security Framework

Umbrella: Don't get wet !

The Umbrella Team:

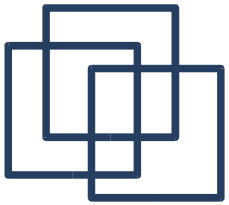
- Søren Nøhr Christensen (student)
- Emmanuel Fleury (assistant professor)
- Kristian Sørensen (student)
- Michel Thrysøe (student)

http://umbrella.sourceforge.net/



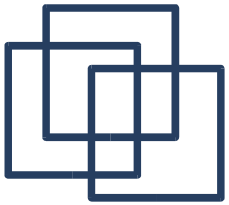
Project Background

- 2003
  - Project start in September at Aalborg (Denmark)
  - Goal: Improve security on handhelds
- 2004
  - Umbrella launched in February
  - Master's Thesis completed in June
  - Continued in September with TDC (Denmark Telecom)
- 2005
  - Continue with Panasonic Research



Project Partners

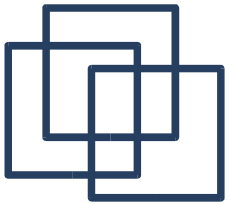
- **September – December 2004**
  - CISS (Center for Embedded Software Systems)
  - TDC (Denmark Telecom)
    - Prototype for alarm box
- **January – June 2005**
  - CISS (Center for Embedded Software Systems)
  - Panasonic
    - Implement kernel keyring, Testing and optimization
    - Other features



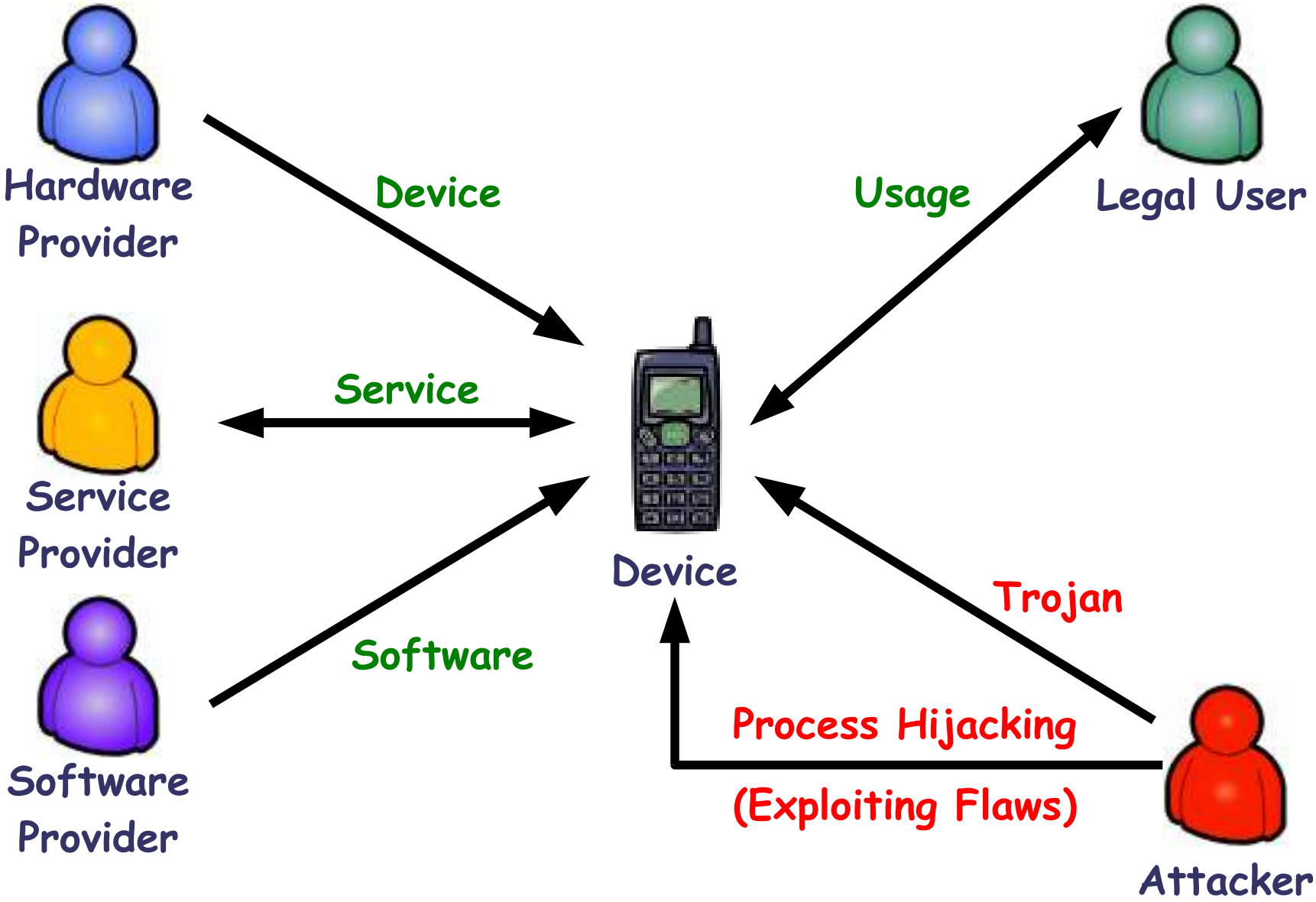
Umbrella Goals

A combination of process-based access control and signing of binaries targeting Consumer Electronics

- Easy to deploy and to maintain
- Transparent to the user
- Avoid global configuration of the security policy
- Can handle the restrictions process by process



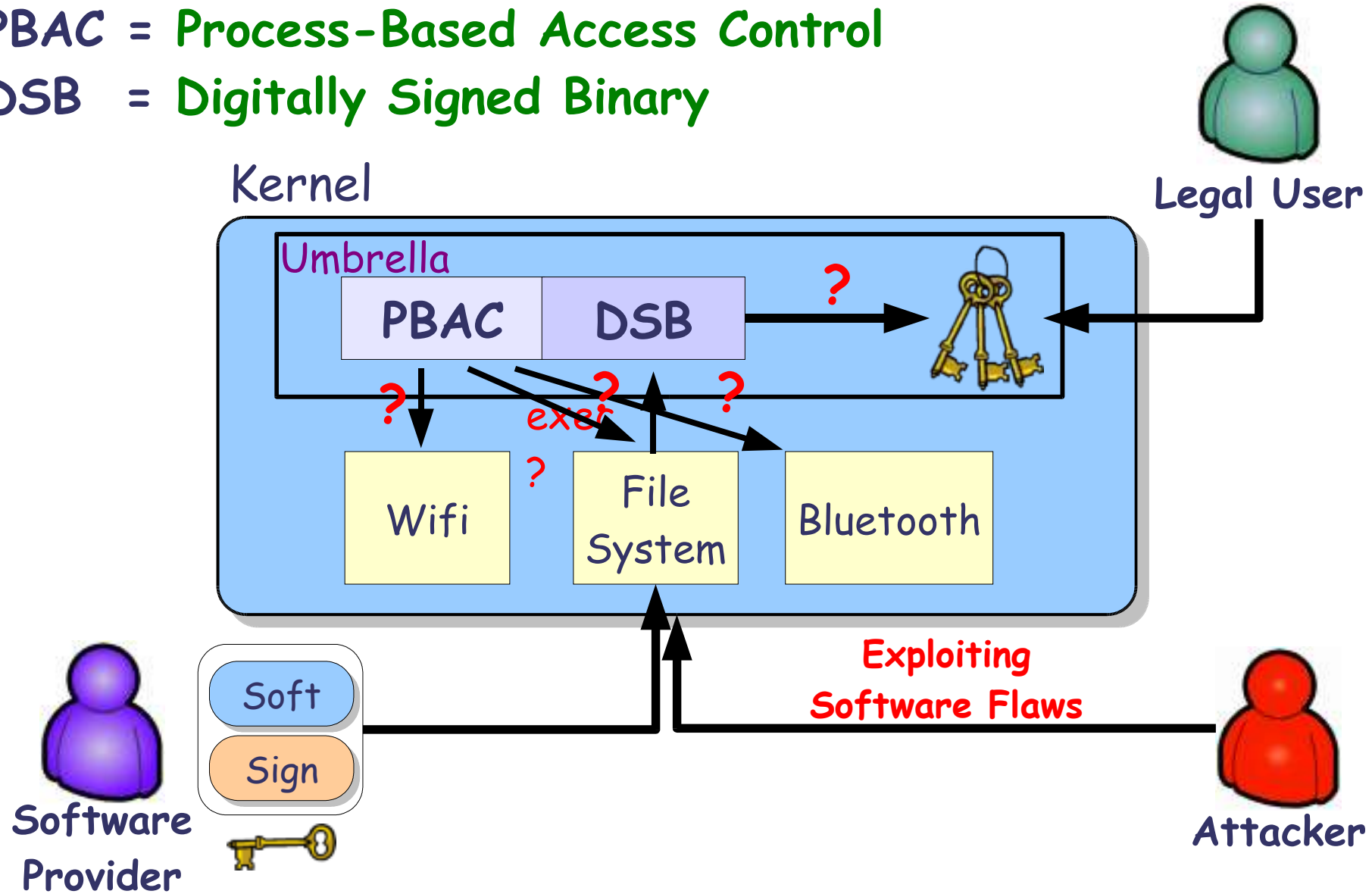
Umbrella Context



Top Level Design

PBAC = Process-Based Access Control

DSB = Digitally Signed Binary

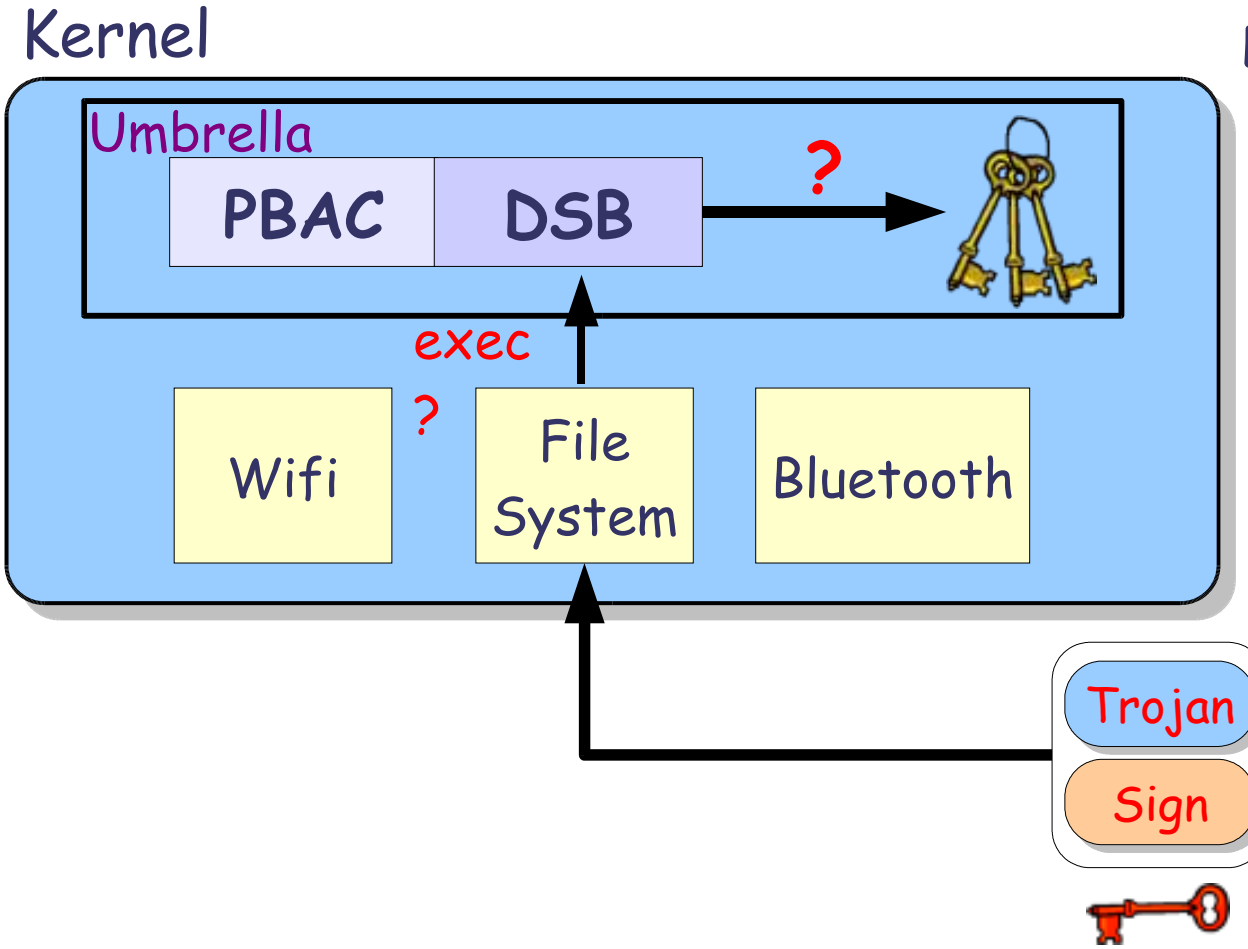


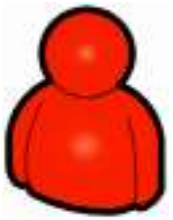
Top Level Design

PBAC = Process-Based Access Control

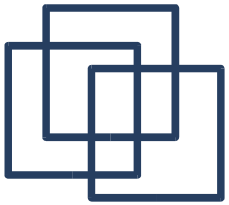
DSB = Digitally Signed Binary

Legal User



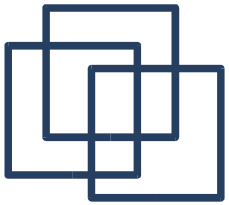


Attacker

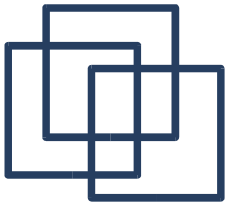


Roadmap

- 0.3 Process-based restrictions
  - Restrictions can be set using restricted fork
- 0.5 Execute restrictions
  - Restrictions can be embedded and applied when executing
- 0.6 Integration with GNU Privacy Guard
  - Authenticate binaries and check restrictions integrity
- 0.7 Implement keyring
  - Hold public keys of several vendors
- 0.8 Feature complete
- 0.9-1.0 Bug fixing and optimization

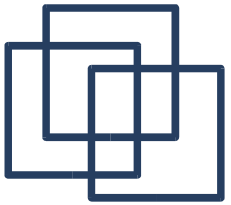


Process-Based Access Control (PBAC)



Related Projects

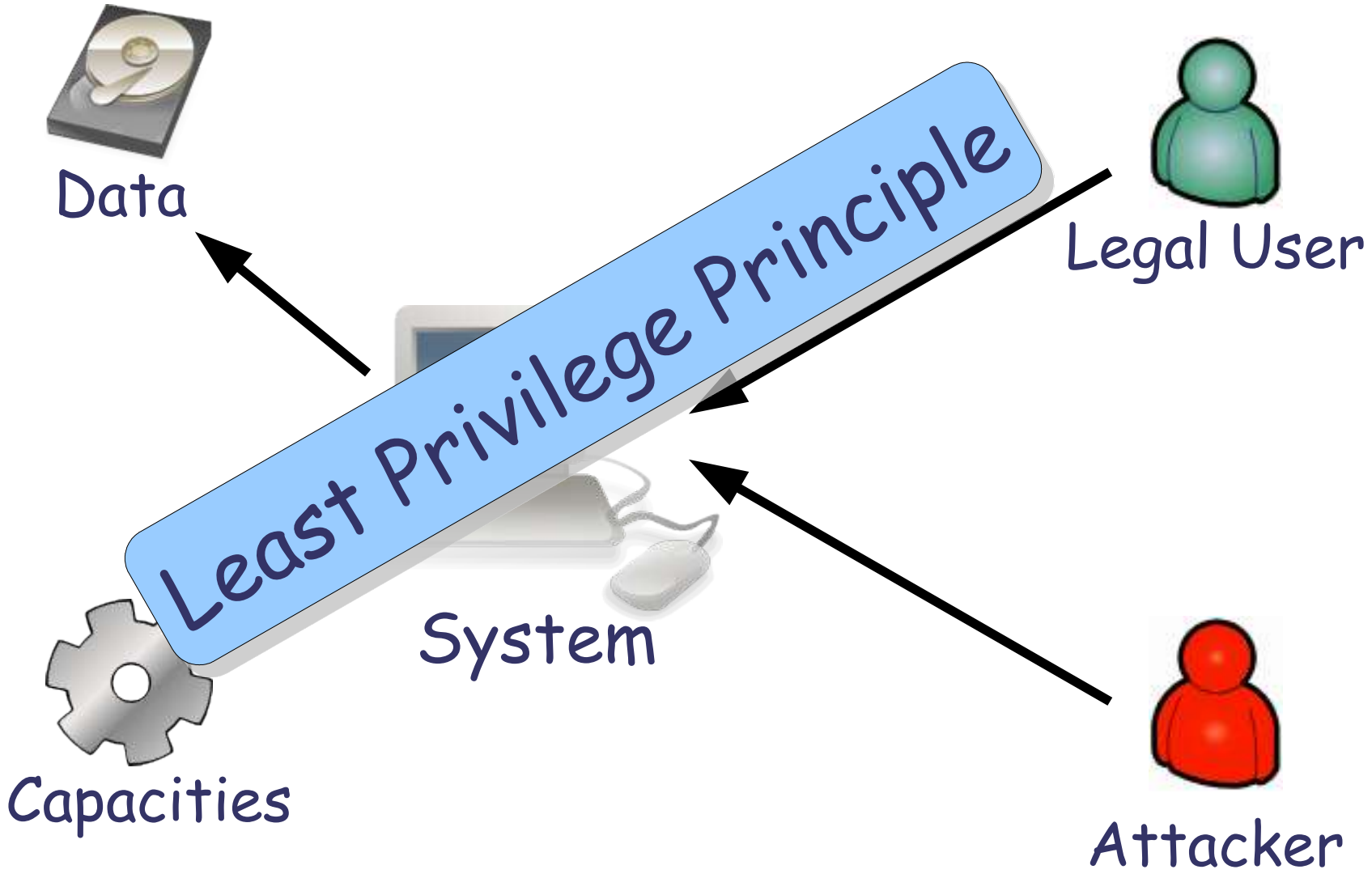
- **Security-Enhanced Linux (SE Linux)**
  - Combination of different security mechanisms
    - Role-Based access control
    - Type-enforcement
    - Multi-level security
  - Extreme fine granularity
    - Administrators can configure it extremely precisely
    - Complex to understand and maintain

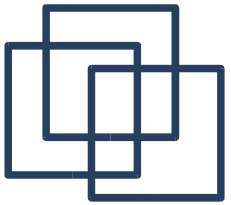


Other Related Projects

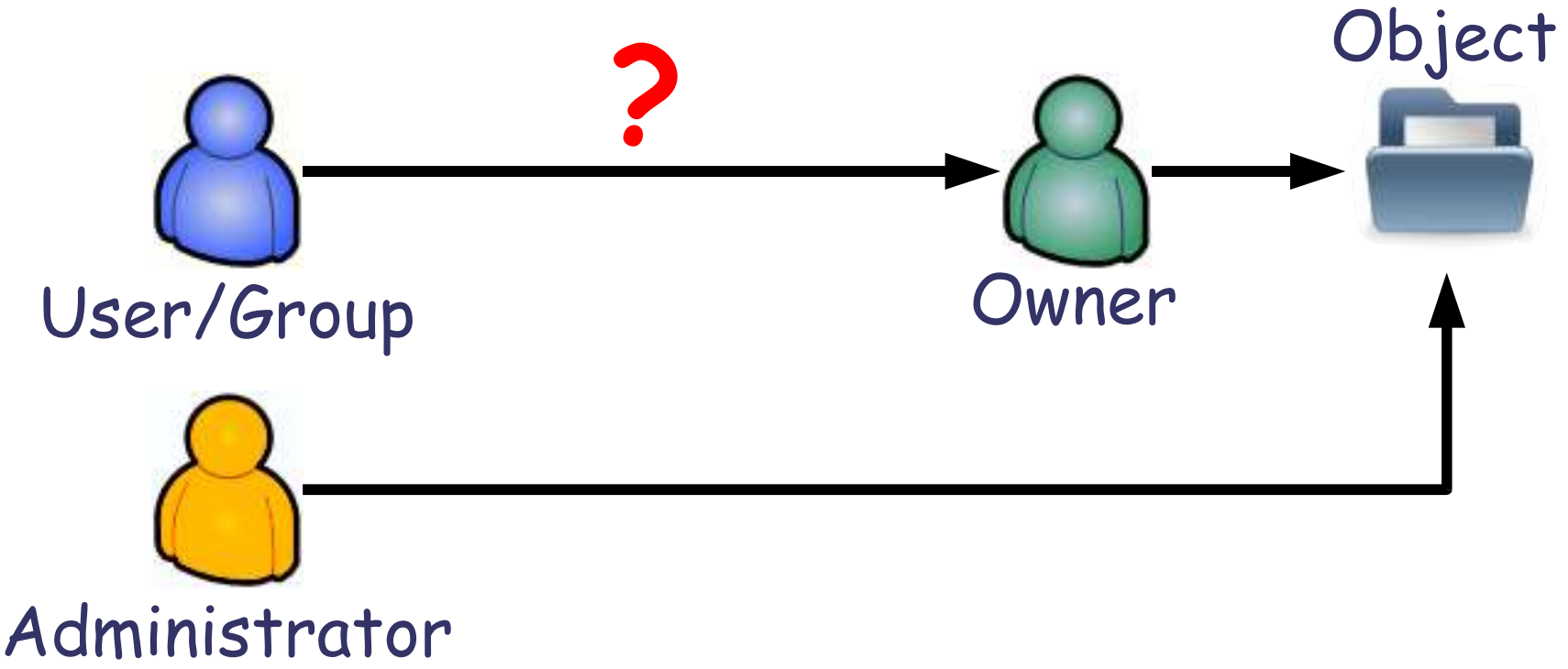
- **SubDomain**
  - Least privilege mechanism based on programs
  - Easy to understand Security-Enhanced Linux
  - Closed source owned by Immunix
- **Medusa DS9**
  - Virtual Space Model
  - Security decision center in user space
- **Rule Set Based Access Control** (RSBAC)
- **Linux Intrusion Detection System** (LIDS)
- **Grsecurity**
- ...

What is Access Control About ?

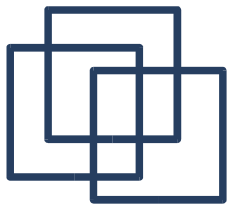




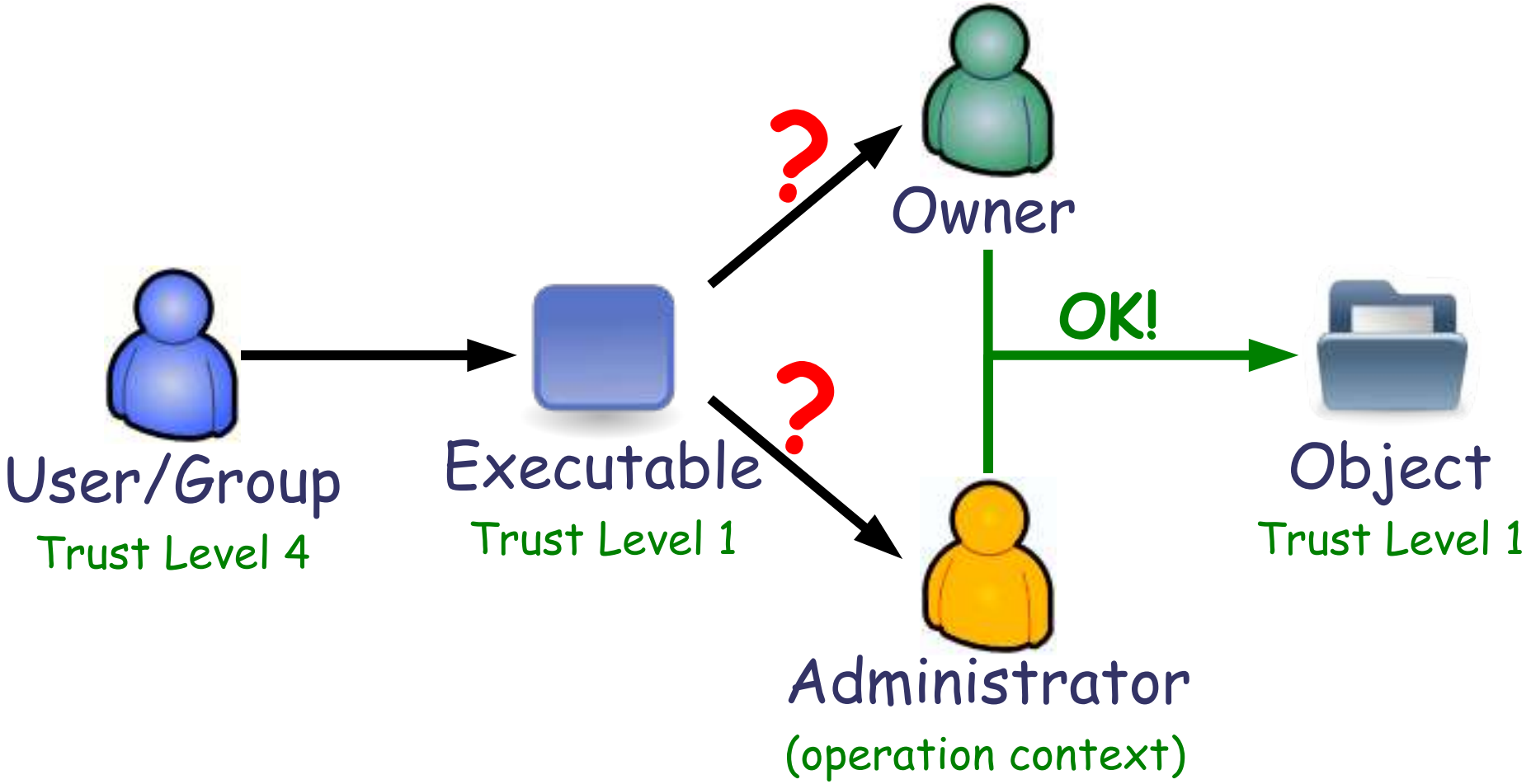
Discretionary Access Control



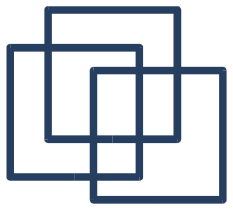
Access to an Object is left to the Discretion of the owner



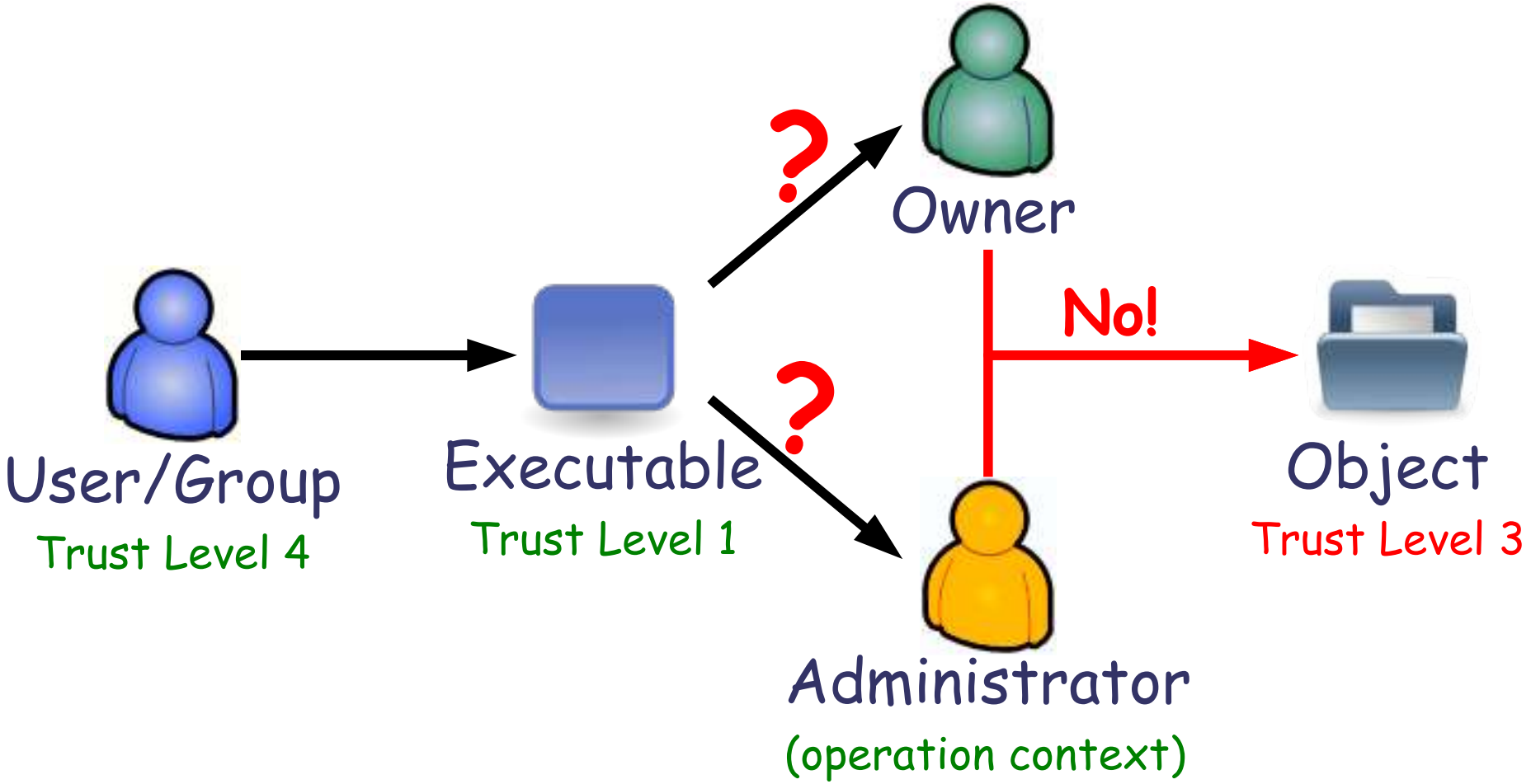
Mandatory Access Control



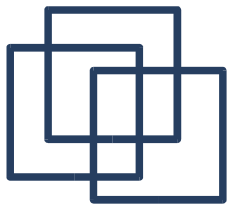
Access to an object is granted depending on the owner decision, the trust level of the subject accessing it and the operation context



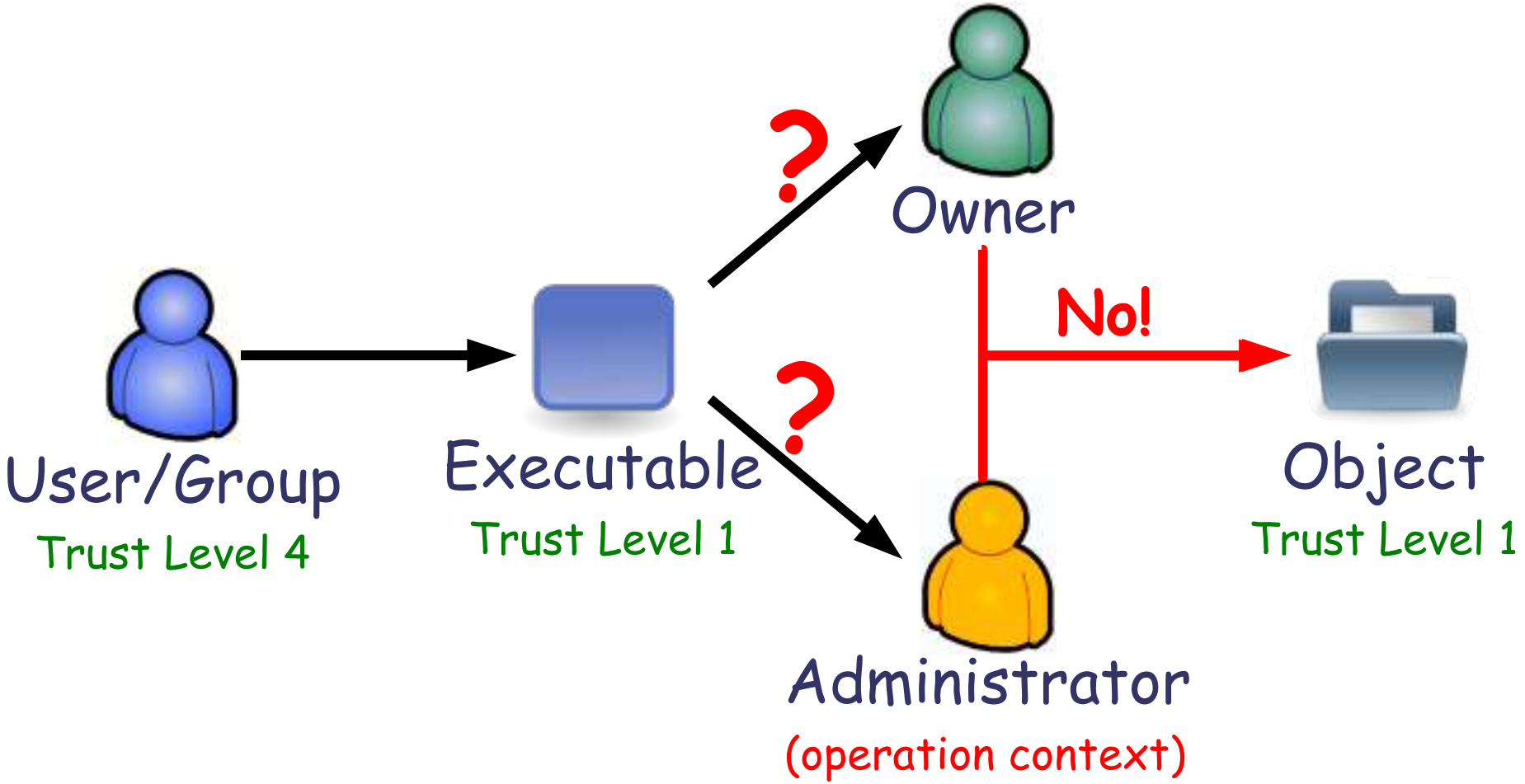
Mandatory Access Control



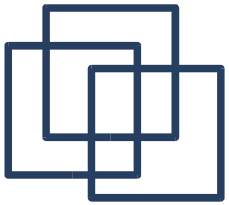
Access to an object is granted depending on the owner decision, the trust level of the subject accessing it and the operation context



Mandatory Access Control



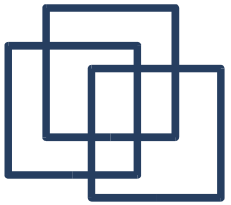
Access to an object is granted depending on the owner decision, the trust level of the subject accessing it and the operation context



Role-Based Access Control

A super-set of Mandatory Access Control where access is granted based on:

- Object's Owner decision
- User's Role(s) (lattice over roles)
- Object's Trust Level (lattice over objects)
- Operation Context (relations between objects)



What's Wrong ?

- **Discretionary Access Control:**

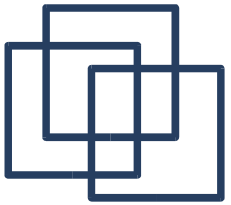
  Not fine grained enough
  (cannot stop trojans within the user environment)

- **Mandatory Access Control:**

  Operation's context is complex to configure
  (An end-user cannot deal with this)

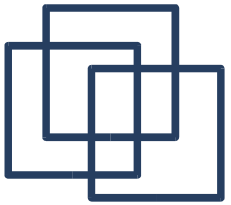
- **Role-Based Access Control:**

  If MAC was already too complex, RBAC is as well.



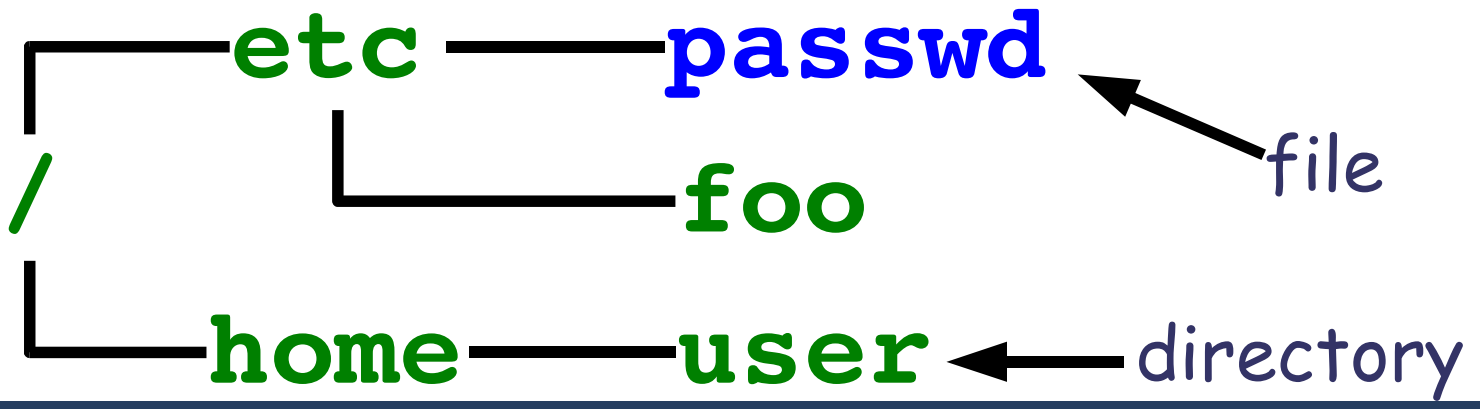
Process-Based Access Control

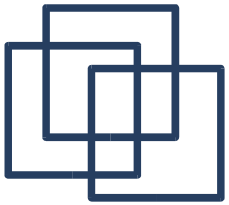
- Combined with **Discretionary Access Control**
- **File-system** & **Capacity** restrictions:
  - Access to /home/john/addressbook
  - Access to the Network
- Restrictions at **Process Level**
  (use process hierarchy to define a global security policy)
- Setting new restrictions through syscalls:
  - **`exec()`** (embedded restrictions)
  - **`rfork()`** (restricted fork)



File-system Restrictions

- Overlap Discretionary Access Control
- Binary restrictions (Access/No Access)
- You can only add restrictions
- Mimic **dentry** data-structure
- Restrictions stored in a tree masking the file-system



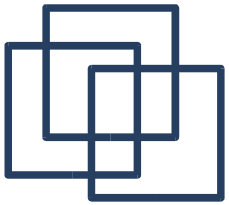


Capacity Restrictions

- Binary restrictions (Access/No Access)
- You can only add restrictions
- Implemented as a 32 bit binary vector
- Checks are performed by masking

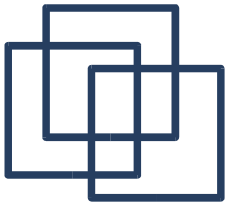
1	0	1	1	0	0	1	0	0	0	0	0	Bit vector
0	0	0	1	0	0	0	0	0	0	0	0	Mask

Restricted !



Capacity Restrictions

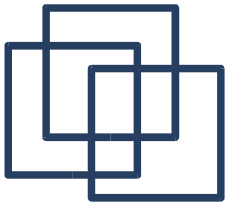
Restrictions	Mediation
SIGKILL	Kill signal
SIGTERM	Termination signal
SIGQUIT	Quit signal
SIGHUP	Hangup signal
SIGTRAP	Trap signal
SIGALRM	Alarm signal
SIGCHLD	Child stopped signal
IPNET	IP socket creation
IRDA	Infra-red device
BLUETOOTH	Bluetooth device
FORK	Fork new process



Global Security Policy

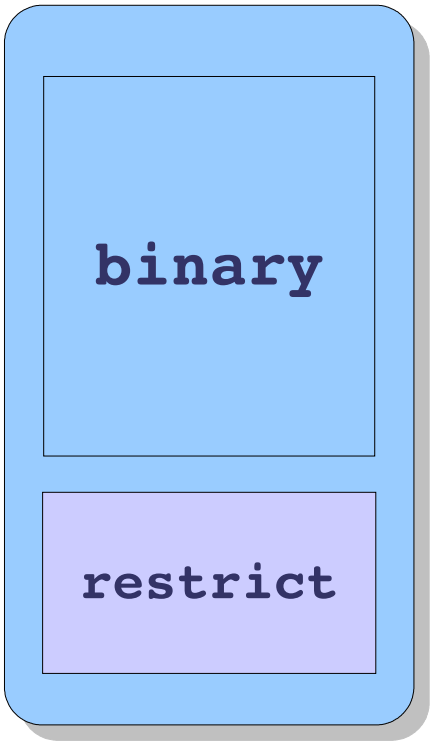
Every Child is at least as restricted as its father!

- Use the process hierarchy
- init is the least restricted process
  (Umbrella can't ensure anything before **`init`**. For this, see TCG)
- Change ownership (setuid) does not help
  (PBAC restrictions are still increasing)
- **`exec()`** can restrict further (see next slide)
- **`fork()`** duplicate the restrictions
- **`rfork()`** restrict further within a program

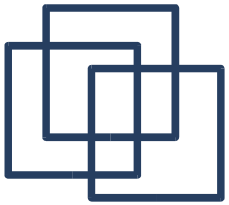


Embedded Restrictions

- Every executable has its restrictions embedded in the ELF format
- When a process call exec():
  - Restrictions from the calling process are added to the restrictions of the executable
  - A new process is spawned and given these new restrictions



ELF Format

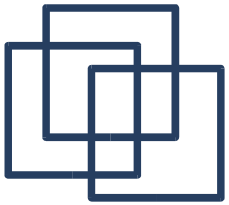


Restricted Fork

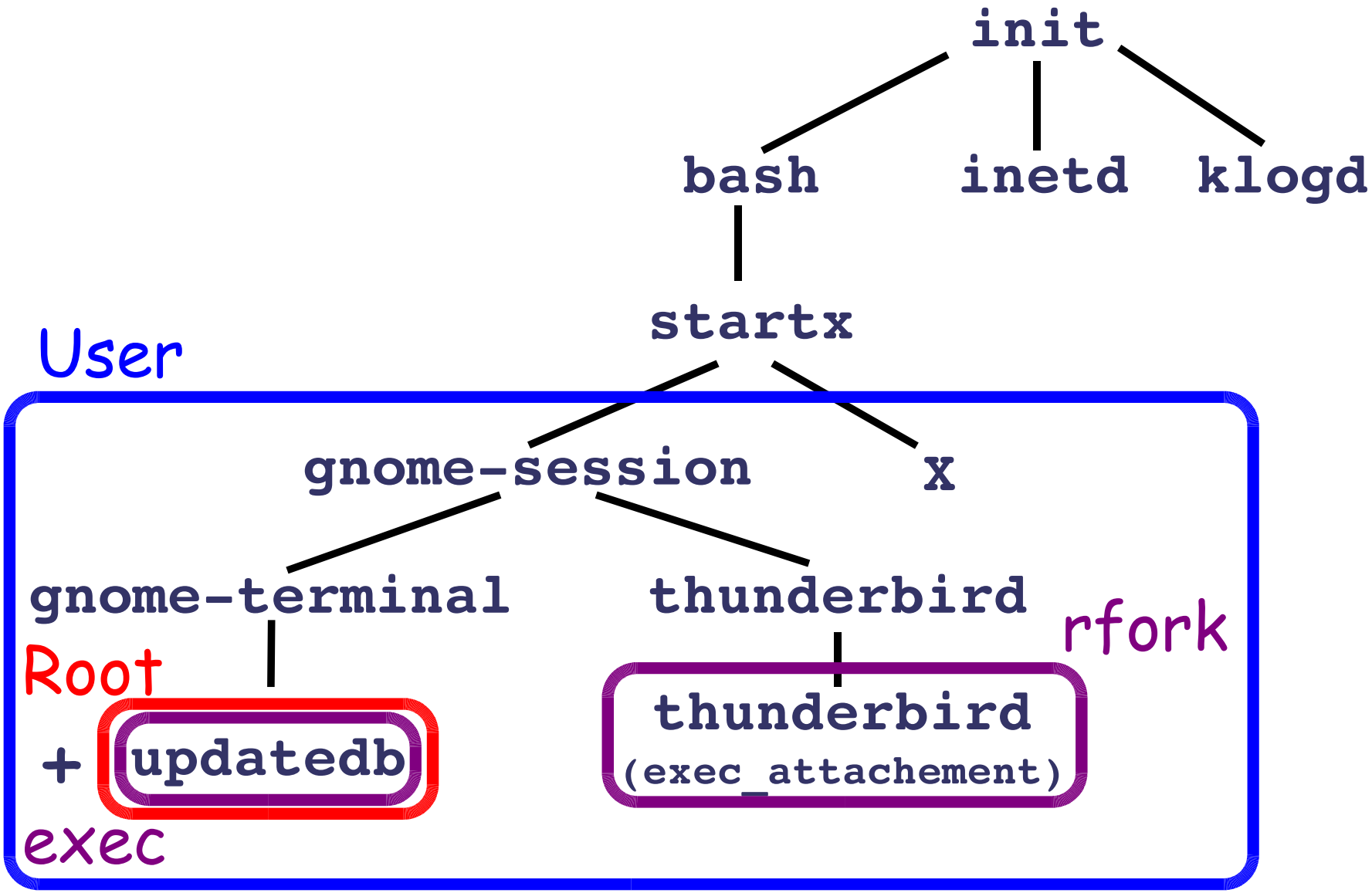
- Within a program a coder can restrict a child process by using the syscall rfork():

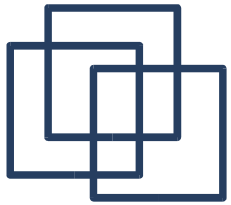
```
rfork(capacity_restrictions,
      file-system_restrictions);

Example:
rfork({IPNET, BLUETOOTH},
      {"/etc/", "/protected/area"});
```

- When called rfork() spawn a process with the restrictions specified in the rfork() added to the restrictions of its father

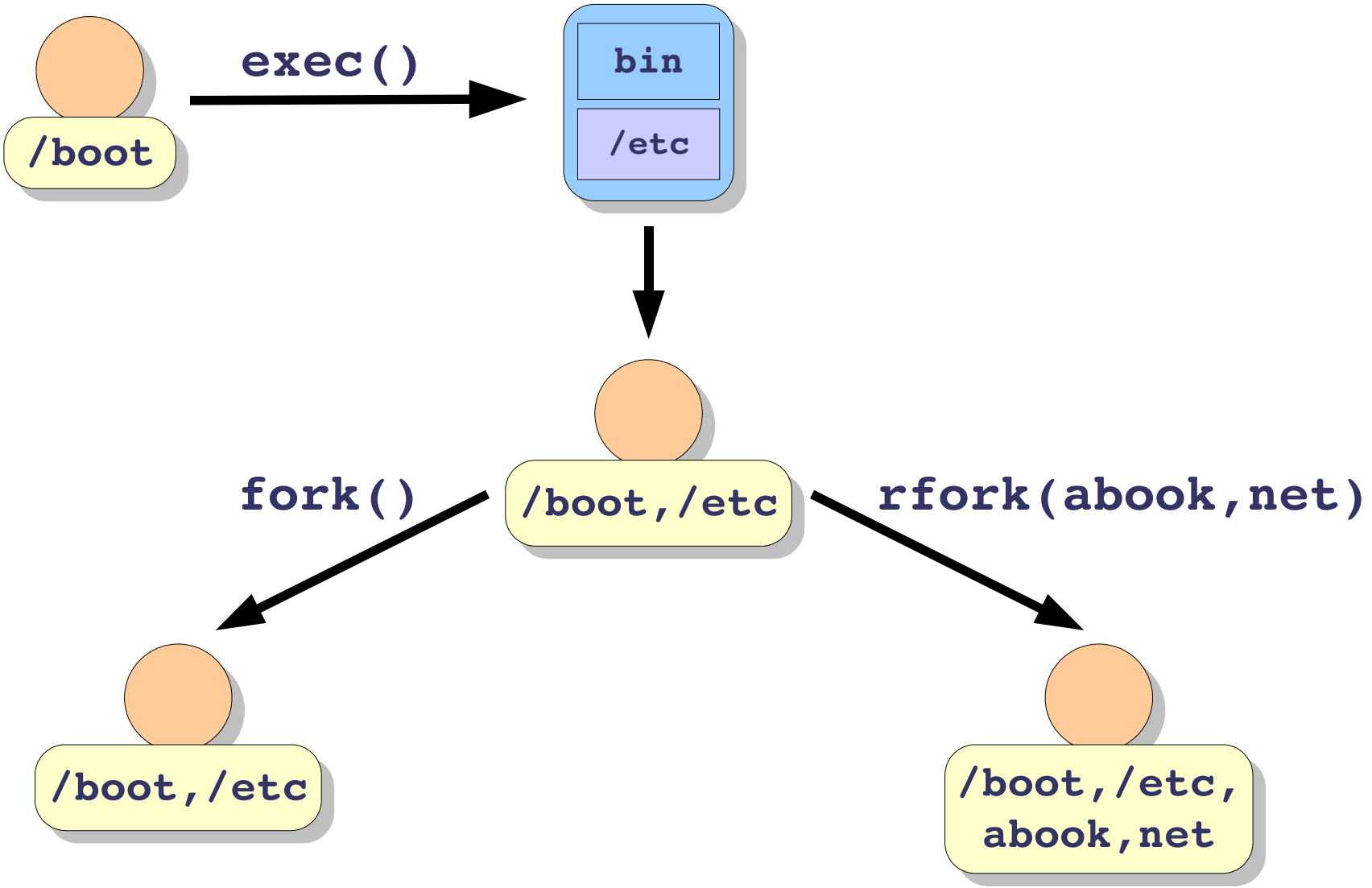


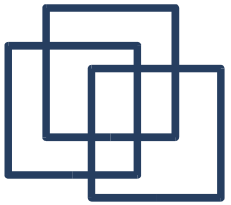
Restrictions & Ownership





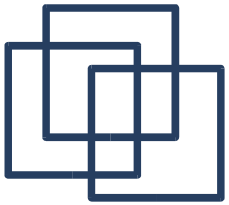
Restrictions Inheritance



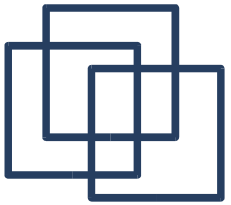


Mediation Through LSM

- PBAC as LSM based module
- Mediating creation of a process through:
  - `task_create()`
  - `task_alloc_security()`
- Mediating access to files through:
  - `inode_permissions()`
  - `inode_unlink()`
  - …
- Mediating access to network through:
  - `socket_create()`

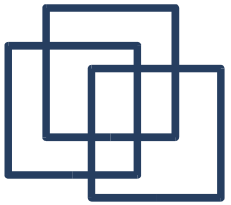


Digitally Signed Binaries (DSB)



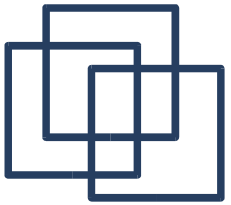
Related Projects

- **Bsign (Debian)**
  - Signed SHA1 inserted into ELF header
- **DigSig (Ericsson Research Lab)**
  - Kernel module for checking BSign signatures
- **Tripwire (Tripwire Inc.)**
  - Intrusion detection with file system hashes

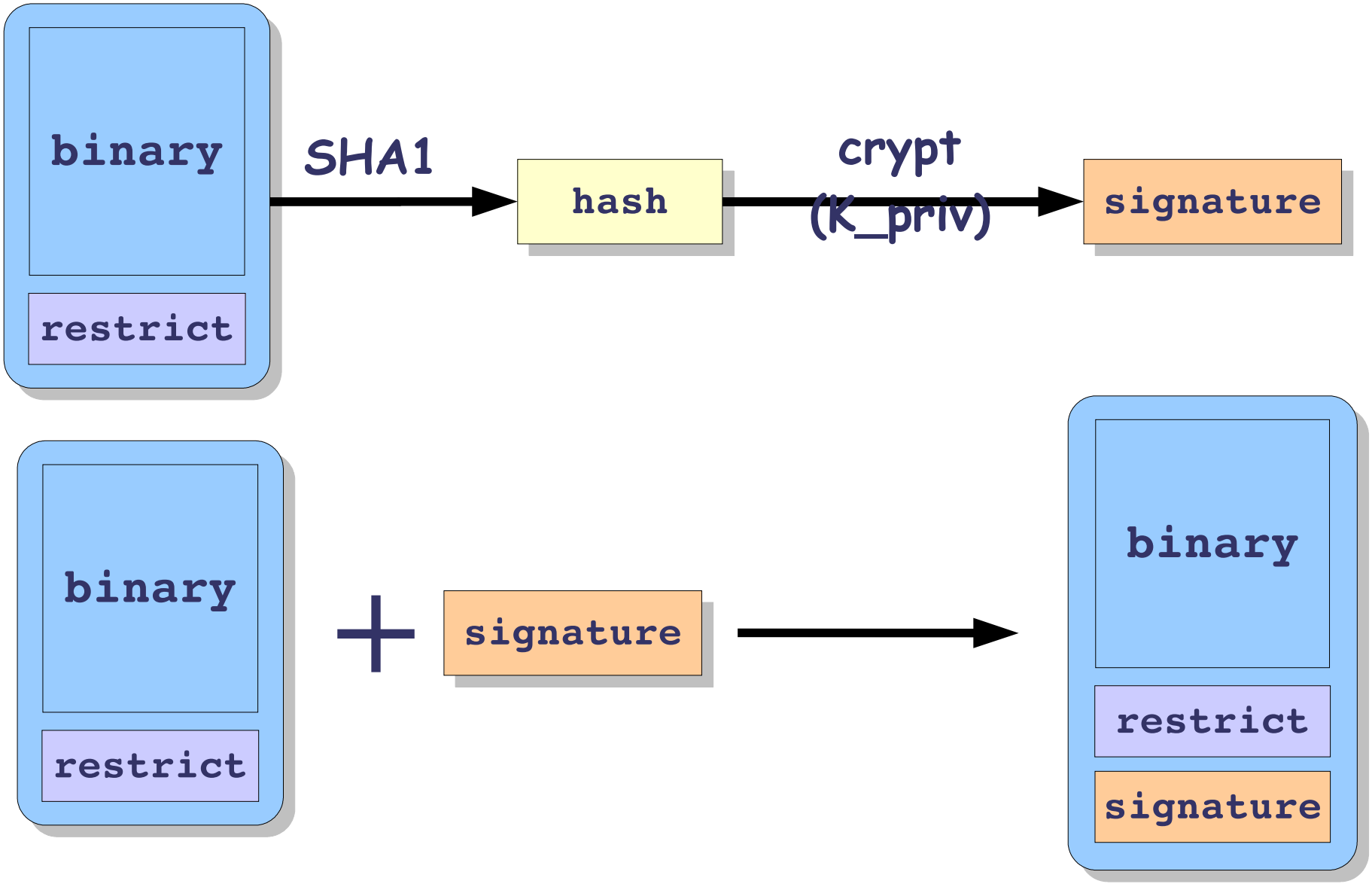


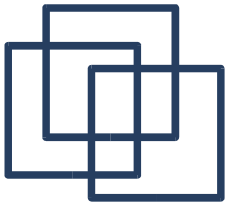
Why Signing Files ?

- How to ensure only trusted binaries are executed ?
- How to ensure integrity of the attached restrictions ?



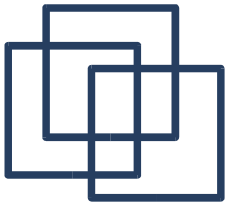
Signing Executables





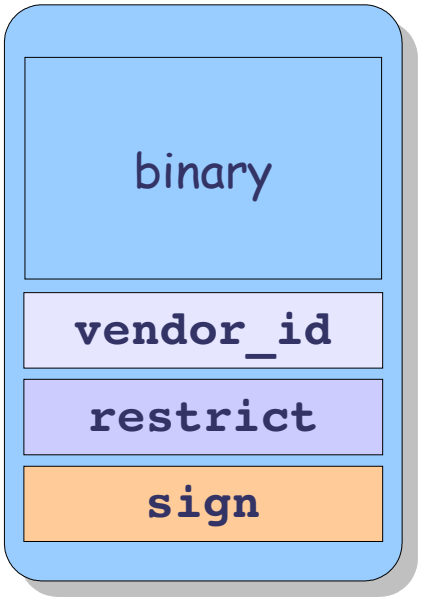
Digitally Signed Binary Format

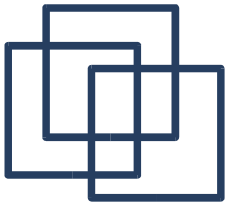
- Append the needed data at the end of the executable file
- Offset to the signature is stored in the ELF header
- Keep track of:
  - Vendor ID
  - Restrictions of the executable
  - Signature of the file



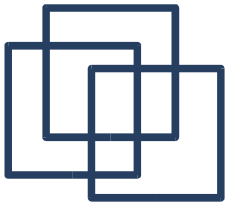
Verification of Executables

1. Get `vendor_id` and fetch the vendor public key
2. If the key is not found go to 7
3. Decrypt the signature with the public key
4. Perform the hash of (binary+vendor_id+restrictions)
5. Compare the two hashes
6. If they match
    1. Add restrictions to the new process
    2. Run the executable and exit
7. Deny execution or sand-box



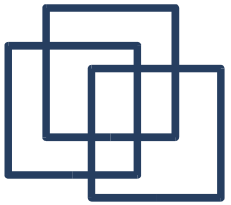


Conclusion
&
Further Work



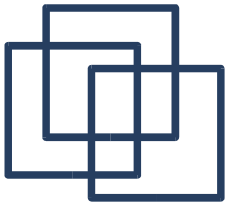
Conclusion

- Goals achieved
  - Simple API ensures easy deployment
  - Almost maintenance free
  - Signed files provide transparency
  - No global security policy to define
- Umbrella is a patch to Linux 2.6.x
- Umbrella is GPL



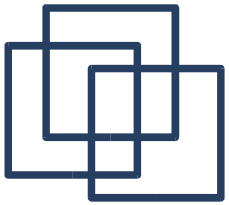
Further Work

- Finish the Digitally Signed Binary
- Design a secure way to handle the key-ring from user-space
- Work on optimization of PBAC
- Try to tackle other problems ? (Trusted paths, Stack scrambling, ...)



Live Demonstration

by Kristian Sørensen



Questions ?