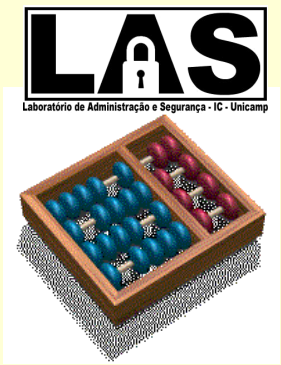
Immunological security systems

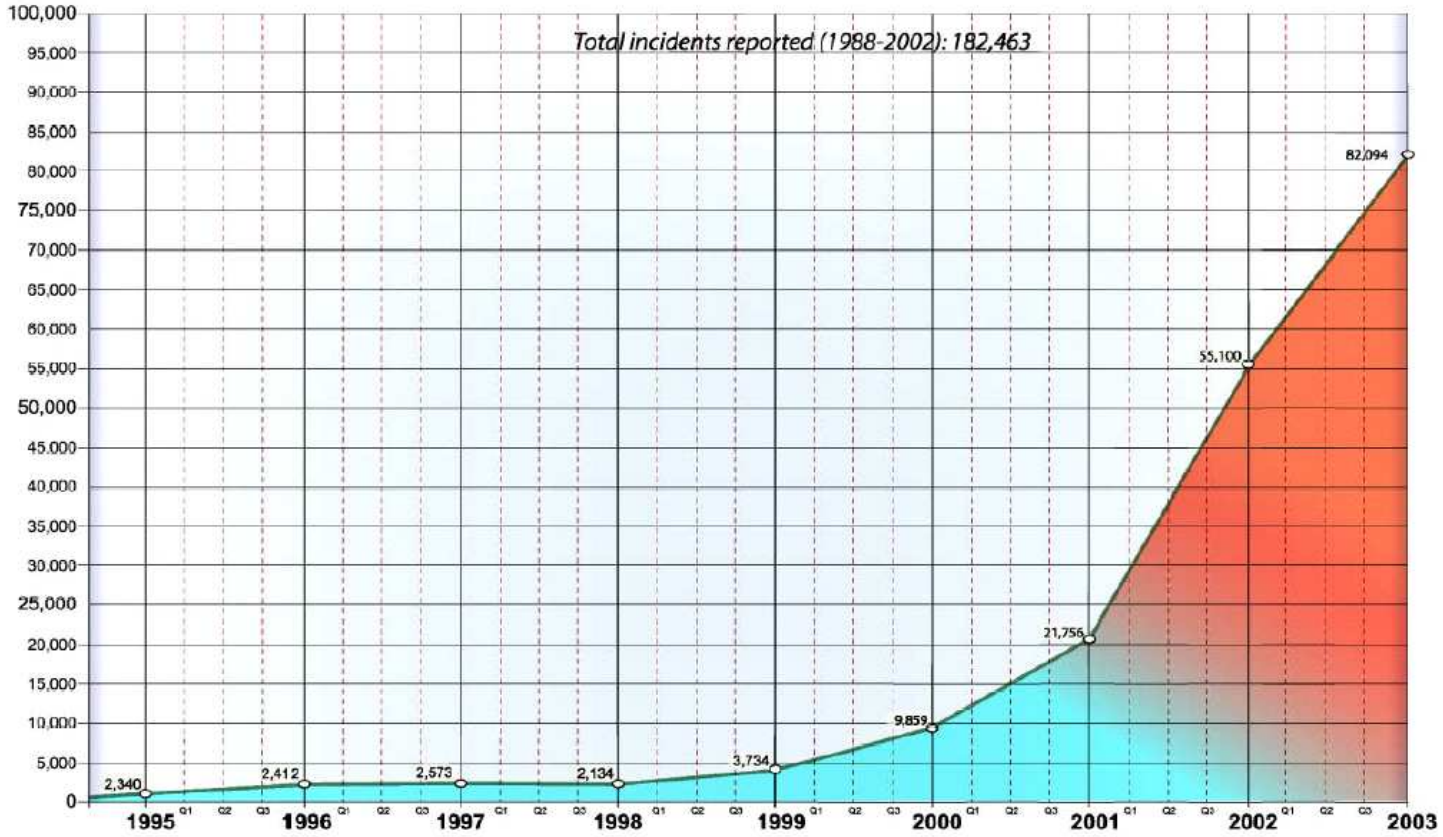
Overview

- Scenario (traditional security approaches)
- Human Immune System, Danger Model
- Imuno Project
- Components
- Analogies
- Prototype
- Some results
- Framework for future development



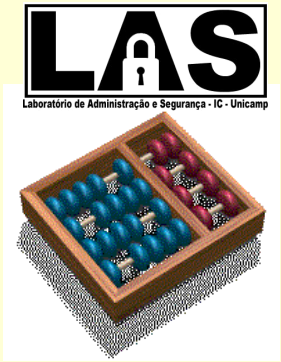
Scenario

- nature of security vulnerabilities
  - » configuration, weak specifications, sloppy programming (culture!)
- security incident rates



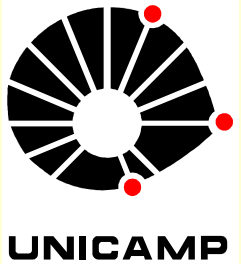
Scenario

- continually increasing
  - despite software writers' efforts—why?
- complex programming environments
  - Web servers + "intelligent" browsers
    - » can you a spot a "guru" on a given set of subsystems?
  - who can program securely, these days?
- security administrators
  - ➔ must be "online" (bug and patch awareness)



Scenario

- hard to maintain a security policy
- inadequacy/absence of native OS tools
  - Windows...
- flexibility demanded internally to the organization



Breaking the vicious cycle

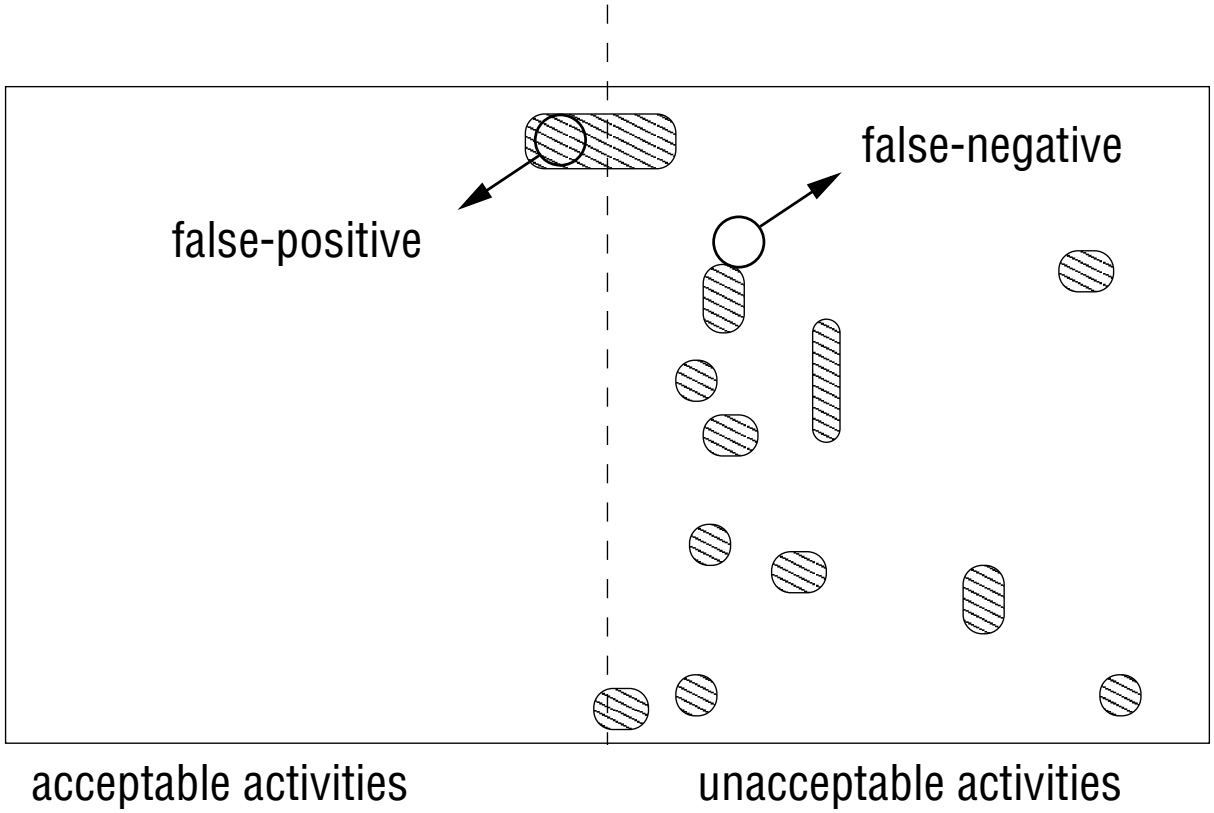
- we are very far from proved-secure OSs
- security must be intrinsic to the playing entities
- autonomous security systems
  - administrators: 'Thank, God!'

IDS

- IDS - Intrusion Detection Systems
  - provide warnings of possibly successful attacks
  - help covering limitations
    - » in the security policy process
    - » in the tools available to enforce policy
  - limited to known attacks
- IPS - Intrusion Prevention Systems
  - currently only help against known attacks
    - » some antiviruses manage some generality in signatures
- databases must be constantly updated
  - administrators still forced to be online…

IDS

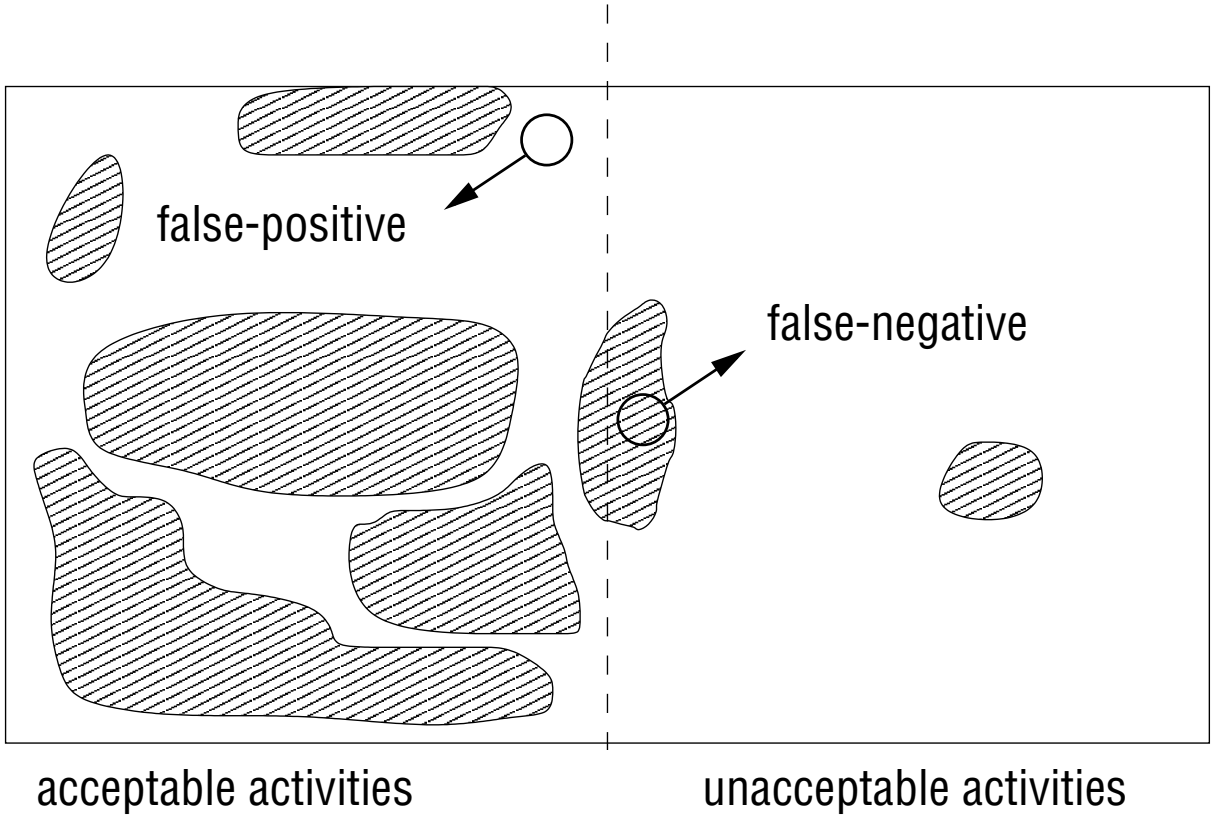
- knowledge-based detection



  - only known attacks (in database)

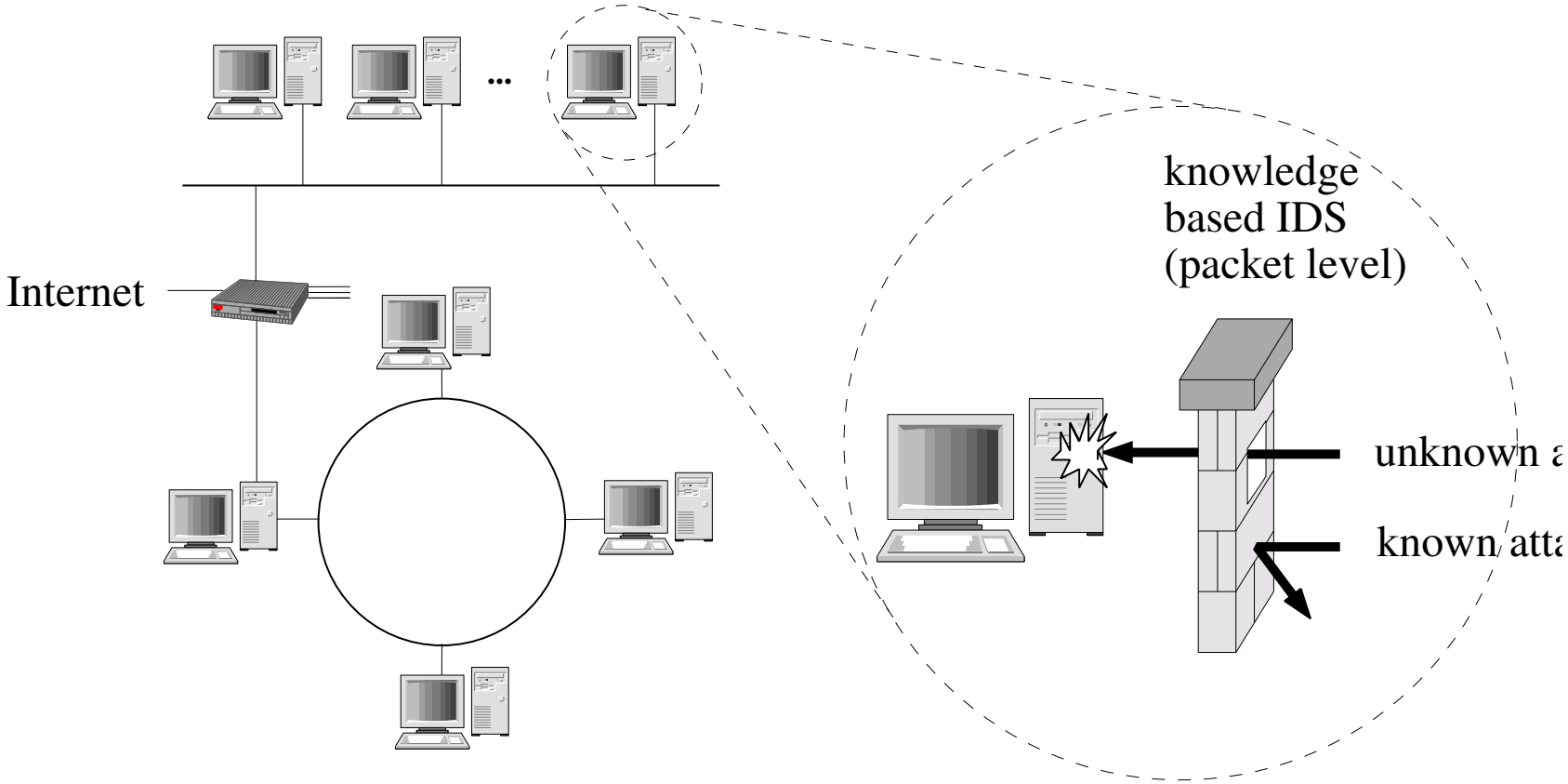
IDS

- behaviour-based detection



  - training required
  - either annoyance to users or highly ineffective

IDS



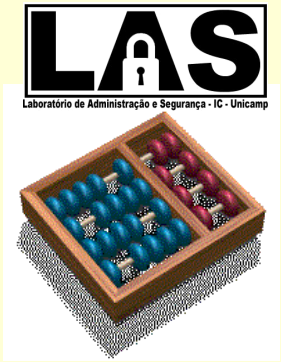
Human Immune System

(or Biological Immune System)

- pioneers (1994 onwards):
  - Stephanie Forrest
  - Jeffrey Kephart
  - Dipankar Dasgupta
  - Steven Hofmeyr

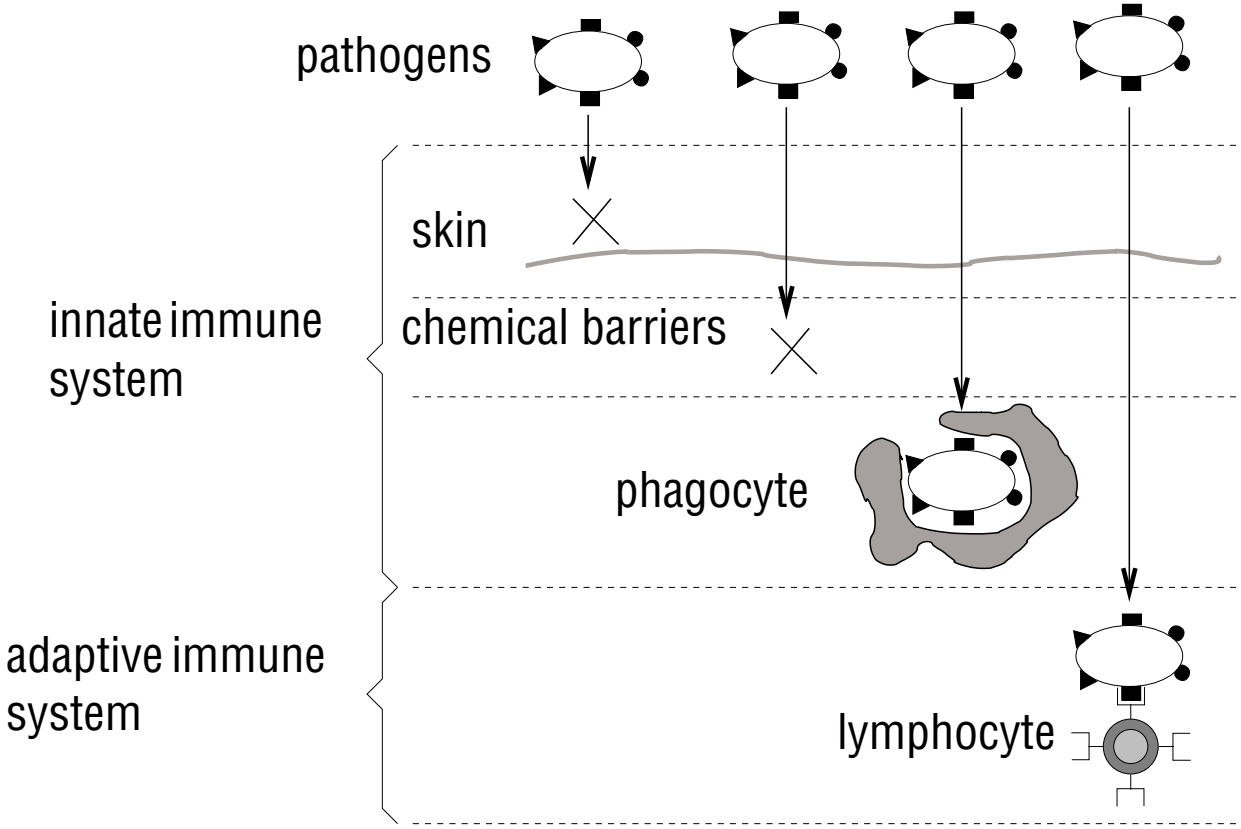
Human Immune System

- innate systems
  - behaviour-based detection, normally inherited
- adaptive systems
  - knowledge-based detection, acquired through exposure
- imperfect (people die!)
  - but very successful (we're around for quite some time...)



Human Immune System

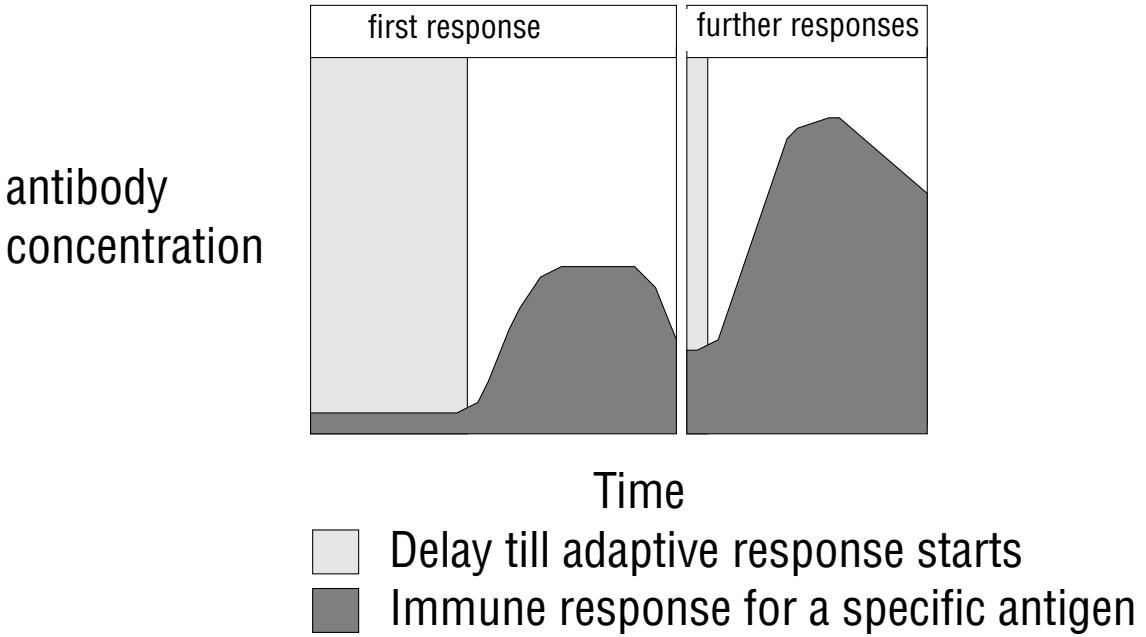
- layered approach (innate, adaptive)



  - very complex mechanism
    - » science has not yet mastered it all...

Human Immune System

- self-regulating response
  - time to build specific antibodies
  - response strength proportional…
    - » to existing number of antigens and antibodies

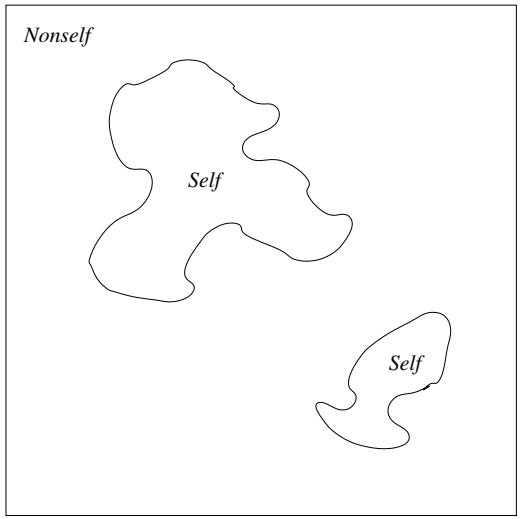


Analogies with computer security

human immune system	computer security
skin: basic physical isolation	packet filter
virus DNA insertion inside the cell	buffer/stack overflow changes in config files
lymphatic system and its memory of past infections	log file analysis, forensics, IPS?
filtering system: mouth, stomach (intestines benefit)	packet filtering
behaviour-based detection (phagocytes)	process and syscall monitoring
counter-attack by phagocytes and related ones	process killing?
digestive system: processing of organic matter to retrieve nutrients; remains are disposed of	proxying, application filtering, data conformance sanitization
tonsils (some languages: amygdalae)	escape goat, booby traps, honeypot
behaviour-based detection (phagocytes)	resource monitoring: mem, I/O, CPU
virus scope → limited to cell (on first analysis)	buffer overflow scope → limited to host process (on first analysis)
knowledge-based detection: creation of lymphocytes	**ideal solution**

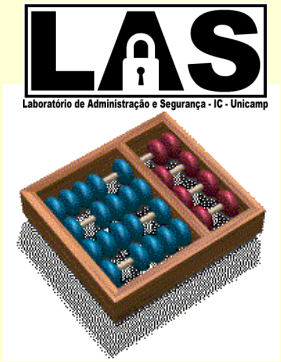
Problems

- telling apart self and nonself
  - Thymus
    - » random generation and maturation process (self/nonself)
  - that *is* the hard part!



Problems

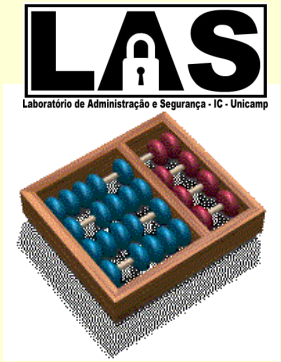
- data patterns in computer attacks - signatures
- domain of possible combinations
- what is “self” in computer application data?
  - some approaches monitor syscall activities



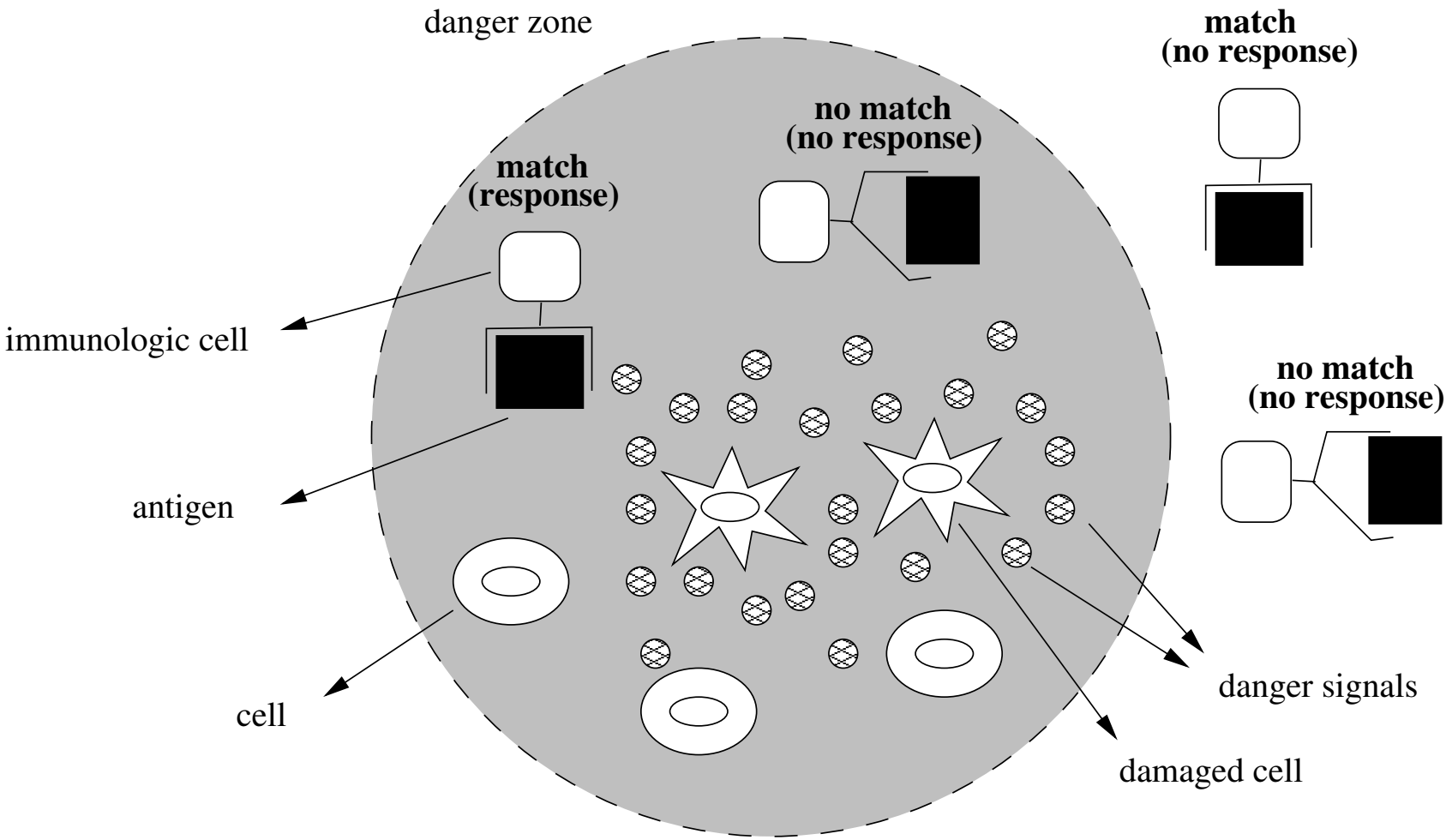
Alternate view - Danger Model

- Thymus function itself does not explain all
  - immune system's selective actions
- controversial theory says…

  detection of damage essential to trigger immune response
- lots of harmless foreign material
  - for which the immune system does not react!
    - » no reaction against useful bacteria in the digestive tract
    - » no response to food ingested
    - » no response either to invading virus that do not cause damage
    - » death of normal cells do not trigger reaction either
    - » the self is slowly, but constantly, changing

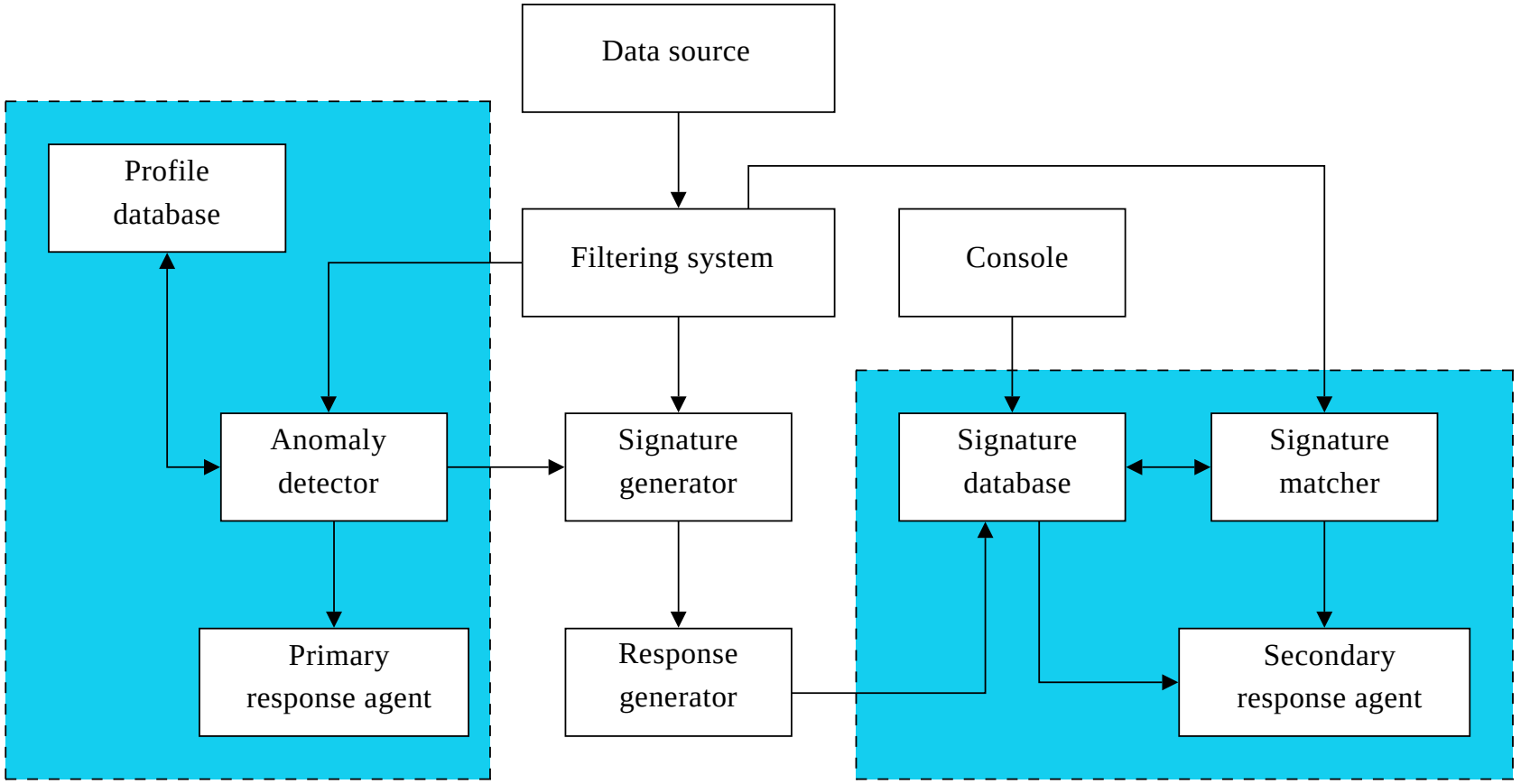


Alternate view - Danger Model



Imuno Project

- Model



Imuno Project

- detection of known attacks and efficient response
  - through signatures for IPS
    - » blocking packets on-the-fly, for example
- detection of unknown attacks
  - evidence analysis (host's realm)
    - » countermeasures -> restoration to acceptable levels
    - » learning from intrusion -> building suitable signature
    - » storing valuable information
      - ➔ manual "forensic" analysis, partial automation
    - » restoration of affected parts (undofs, process restarting)
    - » precise detection/blocking at next exposure (signature obtained)

Components

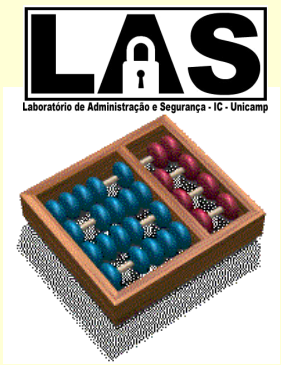
- Console - administrator's interface
  - » config and log files
- Data source - data collection and delivery
  - » network traffic, application and OS events etc
- Knowledge-Based Detector
  - » performed by an available IPS (prevention)

Components

- Adaptive Response Agent
  - » executes pre-specified, precise countermeasures
- Innate Response Agent
  - » smooth, reversible contention measures
  - » system restoration (irreversible actions)
  - » filesystem restoration, restarting processes, reboot
- Response Generator
  - » performs specific measures (new signature)
  - » blocking connections, killing processes etc
- Forensic Support Repository
  - » data collected for further manual inspection

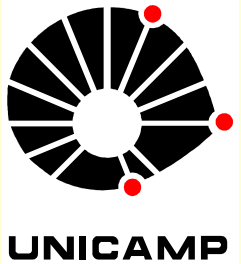
Components

- Behaviour-Based Detector
  - » enables identification of probable attacks
  - » provides moderate, unspecific reactions
  - » identifies unusual events related with unknown attacks
    - ➔ set of candidate attack signatures



Components

- Evidence-Based Detector
  - » search for clear signs of intrusion (successful attack)
  - – attackers almost always perform:
    - » subversion of process -> unexpected accesses
    - » execution of non-authorized services
    - » editing of config files and system logs
    - » establishment of non-authorized, easier communication link
    - » alteration of information in unusual ways
    - » performing of related violations
  - – security policies established beforehand to detect...
    - » indisputable evidences of successful attacks
    - » specific filesystem changes, network connections
    - » execution of new processes, kernel module activities



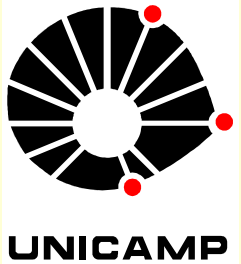
Components

- Signature Extractor
  - » elects data signatures ➔ good candidates for attack signatures
  - » simple algorithm

1. Restore the computer system to a safe state.
2. Select a set C of events to be candidate signatures, where $C \subseteq E$.
3. $progress \leftarrow 0$.
4. While $progress < \left\lceil \frac{|C|}{p} \right\rceil$ do:
   - 4.1. Get a new event $n \in N$ during normal computer system functioning.
   - 4.2. For all $c_i \in C$, if c_i matches n, then $C \leftarrow C \setminus \{c_i\}$.
   - 4.3. $progress \leftarrow progress + 1$.
5. Return each signature in C. If $|C| = 0$, return null.

Similarities with the biological system

- » precise detection of known attacks
- » efficient elimination of known attacks
- » ability to identify unknown attacks in similar/generic way
- » provision for generic, but valuable response to unknown attacks
- » filesystem restoration and elimination of intrusion processes
- » ability to learn and memorize unknown attacks
  - ➔ further precise detection and response.



Similarities with the Danger Model

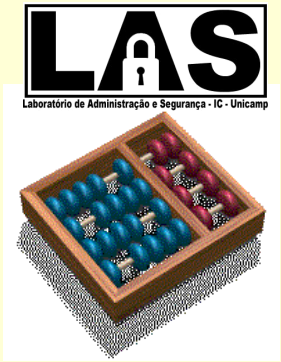
» candidate events limited by proximity in time

➔ to be considered concrete evidence

» no restrictions for closely related events

Example of a policy

```
policy_name[/fully/qualified/application/pathname]{
    fs_acl{ list of pathnames and access permissions }
    can_exec{ list of programs which can be executed }
    max_children = maximum number of children processes
    can_send_signal = yes | no
    can_manip_modules{ list of kernel modules which can be loaded and
unloaded }
    connect_using_tcp = yes | no
    send_using_udp = yes | no
    accept_conn_on_ports{ list of port ranges to listen to }
}
```



Prototype

(1) Precise detection of known attacks: could use Snort;

(2) Precise response of known attacks: some, but not integrated;

(3) Detection through evidence analysis: yes;

  (a) Provision for countermeasures: some, but not integrated;

  (b) Attack-related packet extraction: yes;

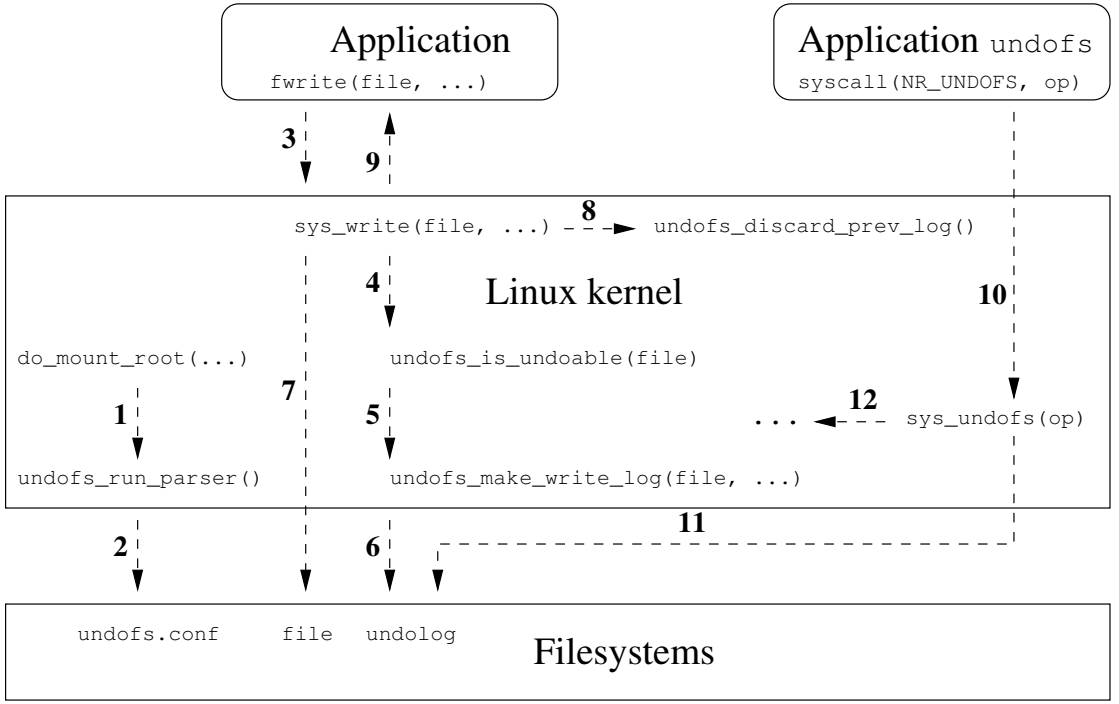
  (c) Filesystem forensics support: yes;

  (d) Filesystem/process restoration: yes.

(4) Precise detection/response at next exposure: partially implemented.

Prototype

- Case studied: buffer overflow
- Filesystem restoration (undofs)

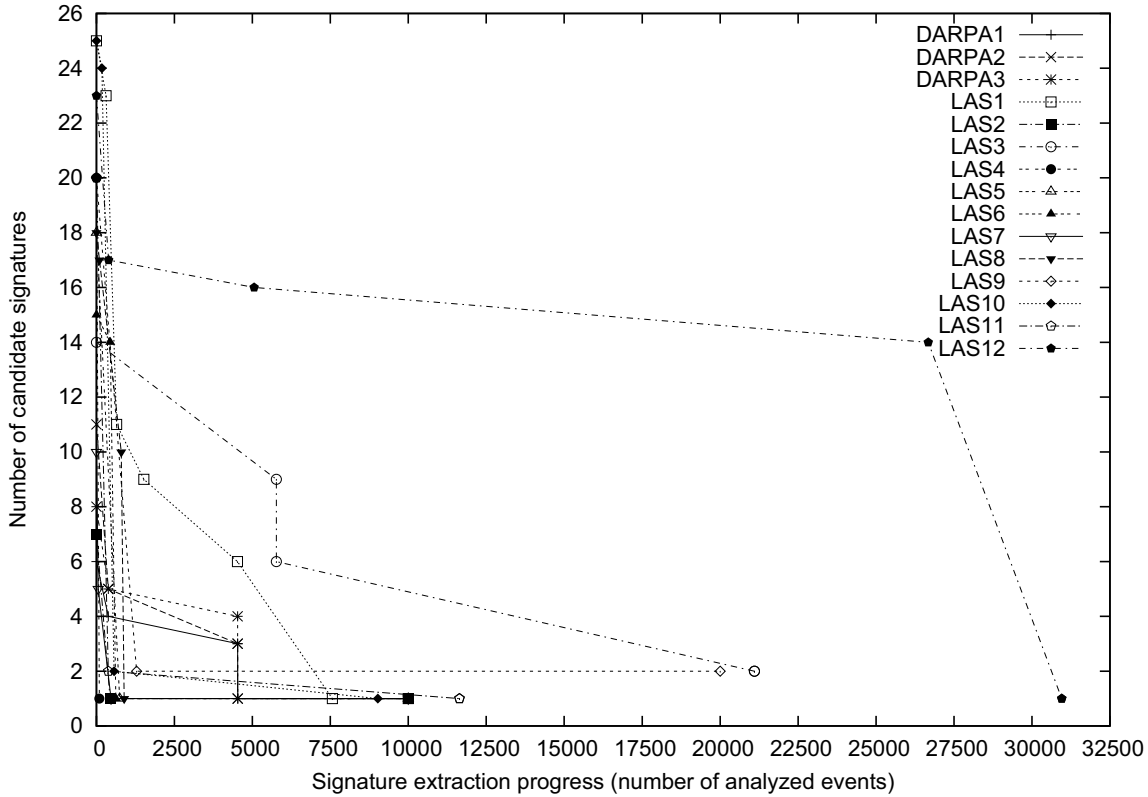


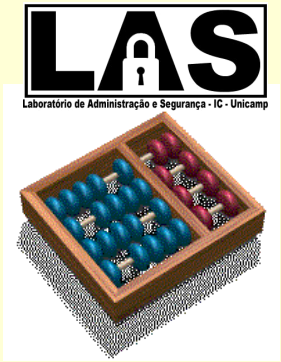
Results

- Datasets: DARPA and locally collected LAS

Results

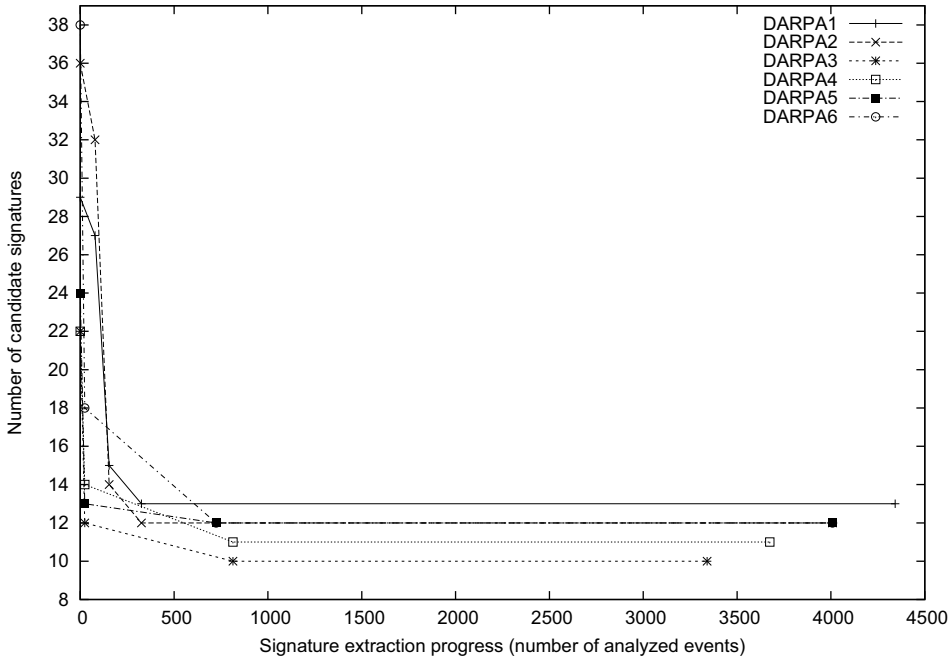
- named: 3 tests with DARPA and 12 with LAS
  - » real attack signature found in all tests
  - » but 2 LAS tests produced one false-positive each

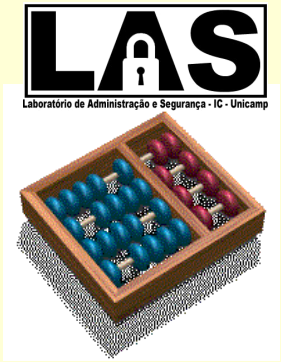




Results

- `wu-ftpd`: 6 tests with DARPA
  - » real attack signature found in all tests
  - » but some false-negative signatures were produced too



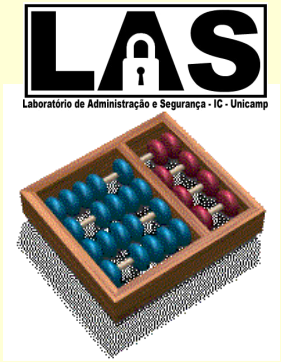


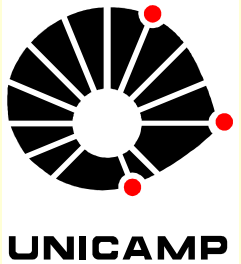
Problem

- Follow-up work
  - single kernel implementation used throughout (2.4.19)
  - very hard to bring to current kernel line (2.6)
  - continuous job of keeping up with kernel (sub)releases

Imuno framework requirements

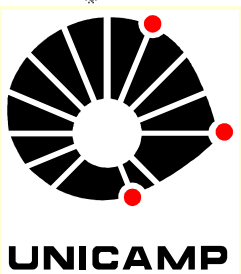
- Prevention
- Detection
  - Host level
  - Network level
- Response
  - Primary (innate)
  - Forensic analysis
  - Secondary (specific)
- Self-protection
- Administration



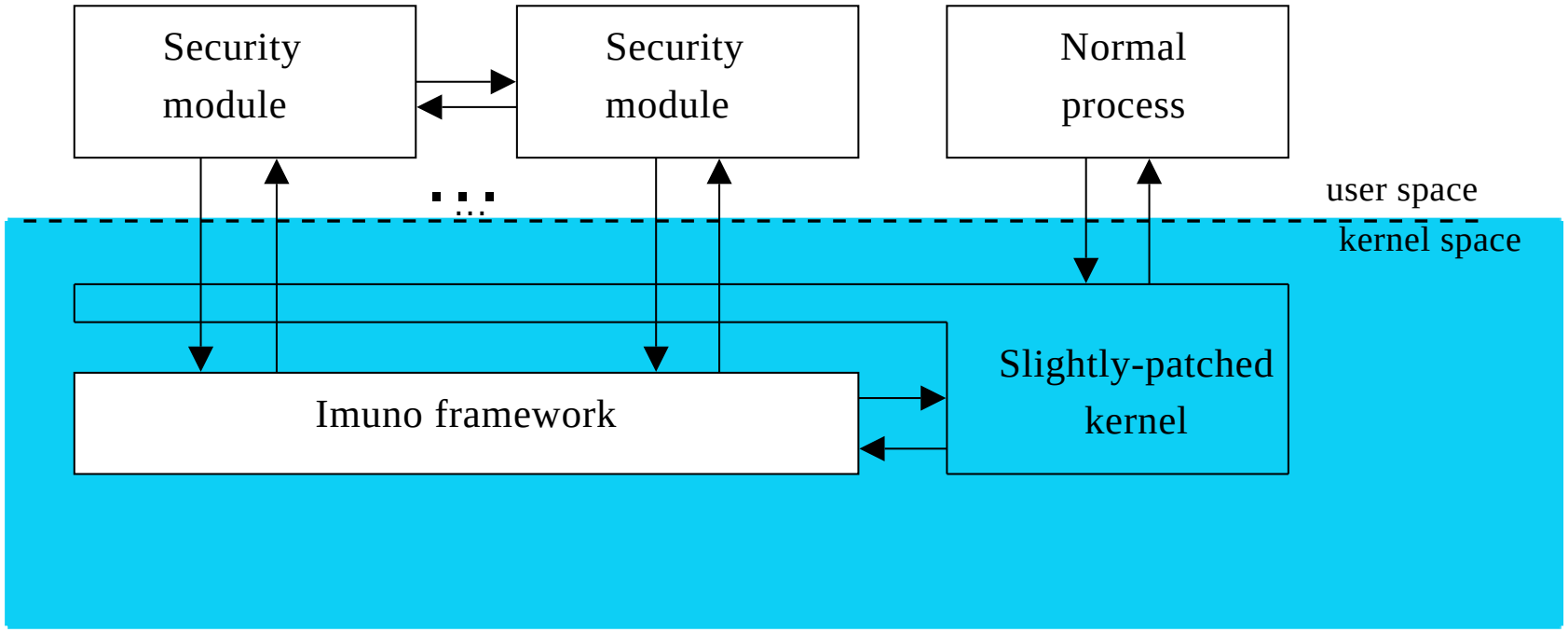


Subsystems used in Imuno

- Linux kernel, 2.6 series
- LSM (Linux Security Modules)
- Netfilter
- CKRM (Class-based kernel resource management)
- BSD Secure Levels



Abstract view



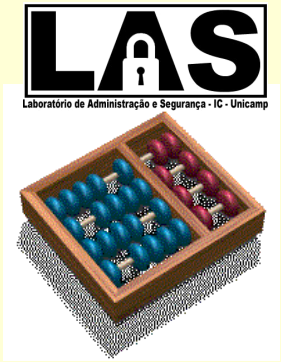
Key features

Multi-functional hooks

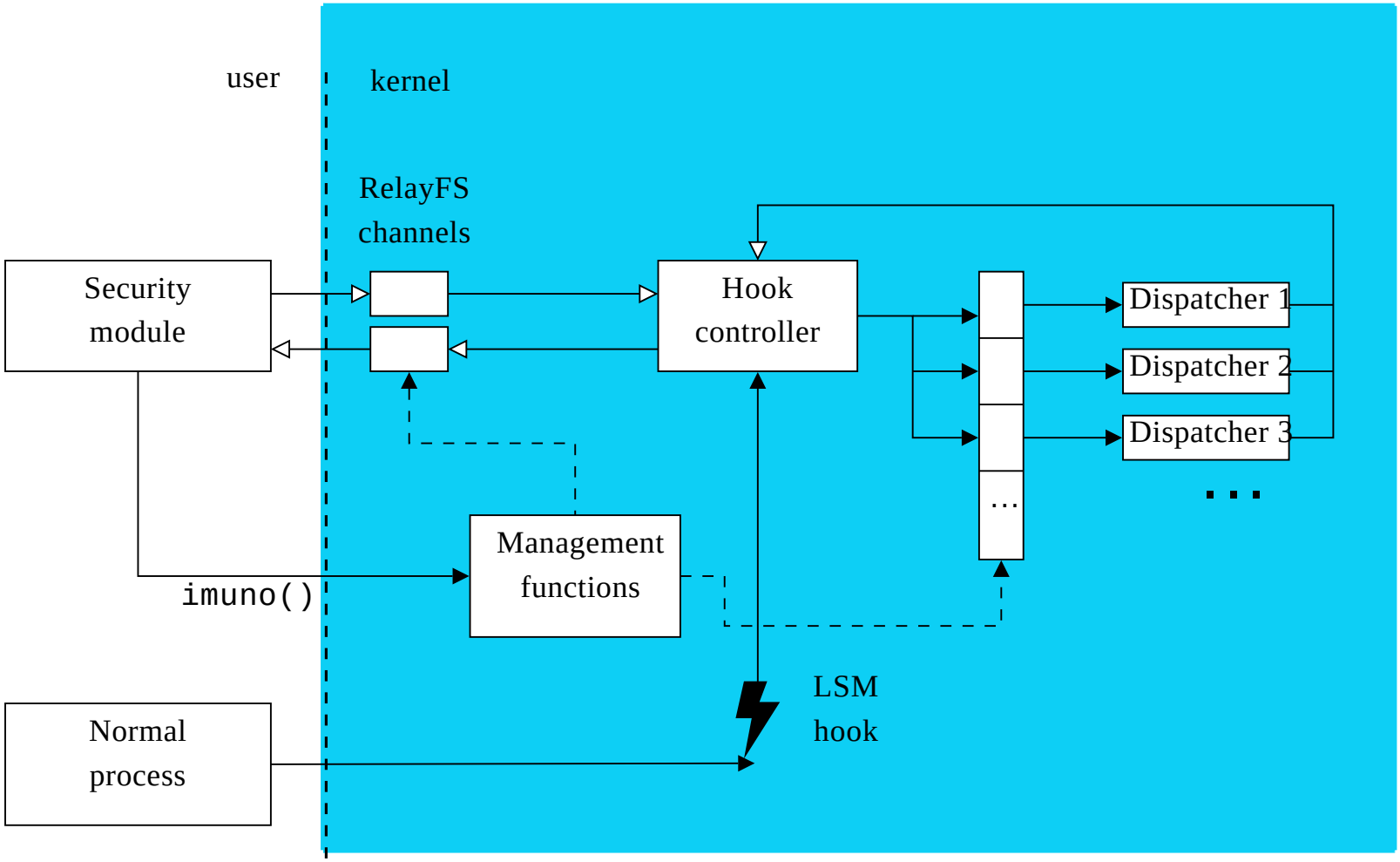
- LSM hook infra-structure → expanded and more flexible
- more freedom in the execution mode
- user-space interaction
- real-time dynamic control

Interface

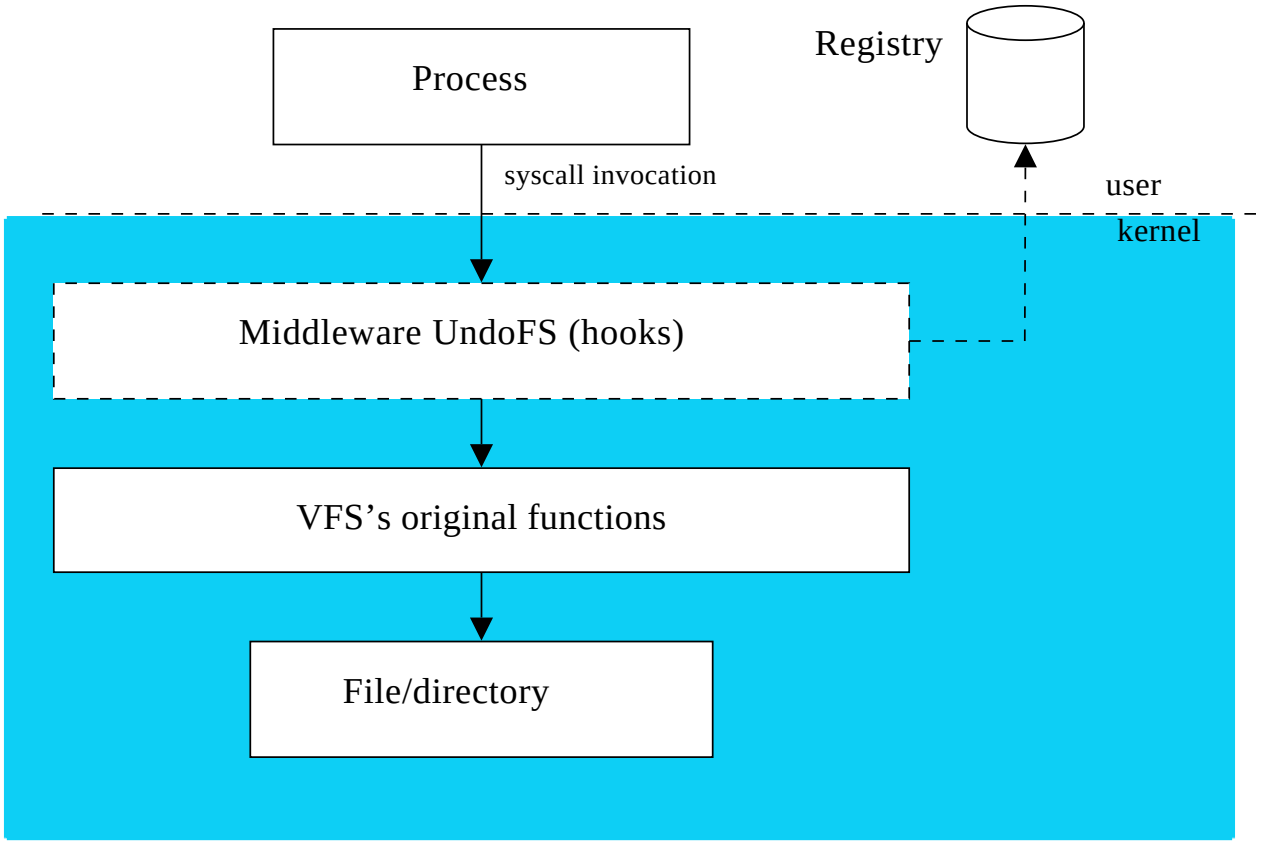
- use of RelayFS pseudo-filesystem
- new syscall added → `imuno()`



Multi-functional hook scheme



UndoFS hooks



- special hooks intercept file-modifying syscalls
- transactions registered on disc
  - ➔ restoration and forensic analysis of filesystems

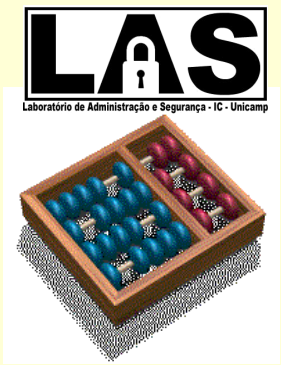
Other subsystems used

Netfilter

- does the job at the network level
- QoS features can also be used

CKRM

- resource control requirements
  - RCFS (Resource Control FileSystem)
- other features not used…



Self-protecting mechanism

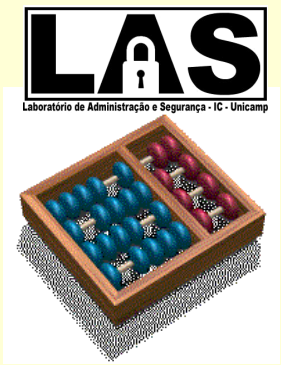
What needs to be protected?

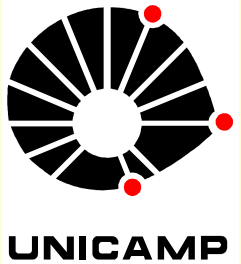
- kernel components, Imuno framework
- user-space processes involved with security
  - those that can implement a immune security system
- vital components of filesystems
- loading chain of the security system

Self-protecting mechanism

Policy that was implemented

- kernel isolation
- other administrative restrictions (BSD SEClvl)
- normal process cannot touch protected object
- protection of the `imuno()` syscall

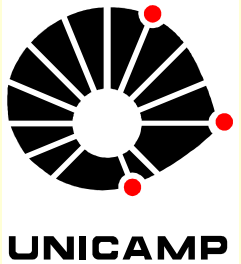




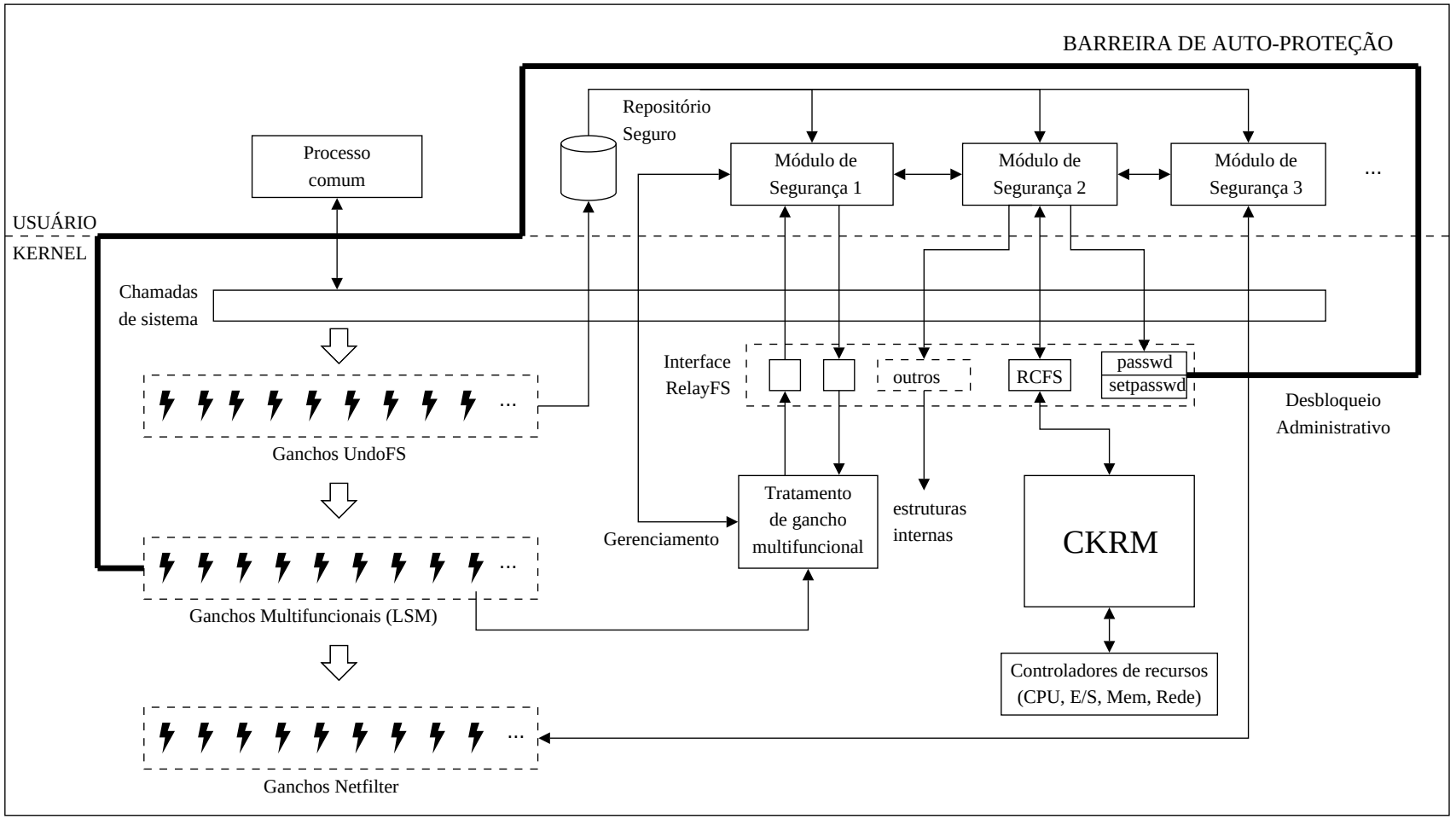
Self-protecting mechanism

Administrative unblocking

- disables self-protection
- password controlled (BSD SEC lvl)
- Interface ➔ 2 RelayFS files
  - `/imuno/interface/setpasswd`
  - `/imuno/interface/passwd`
- can be changed to use TPM
  - Trusted Platform Modules employ hardware crypto



Complete view



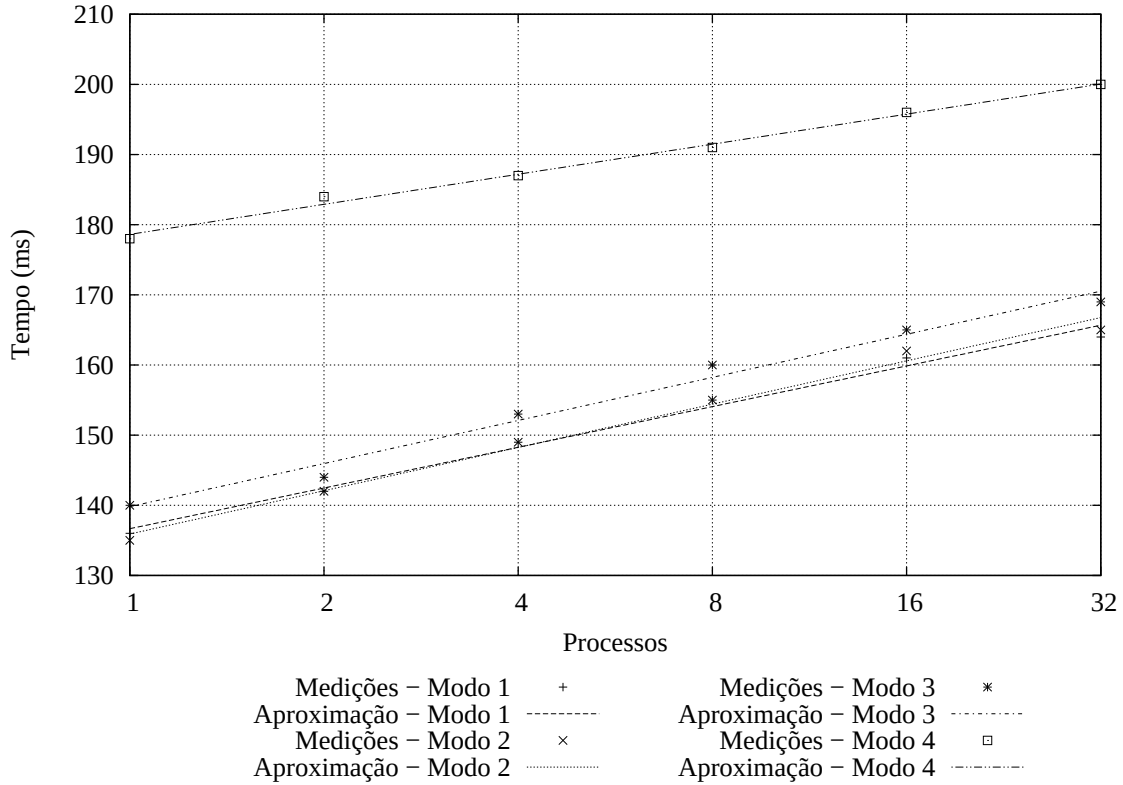
Results of micro-tests

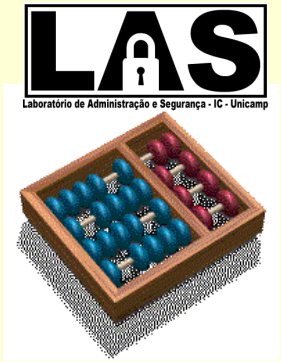
- chosen hook → `security_task_create()`
- benchmark → 1000 loop invocations of `fork()`
- 1, 2, 4, 8, 16 and 32 concurrent processes

Test modes

- standard LSM (Imuno disabled)
- multi-functional, but no registered dispatchers
- multi-functional, 5 dispatchers in sequence mode
- multi-functional, 5 dispatchers in interactive mode
  - controlled by a user-space process

Results of micro-tests





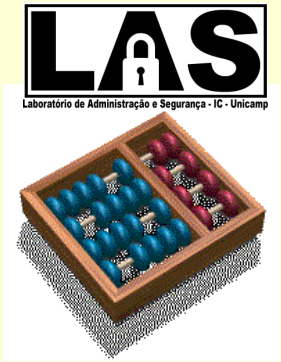
Results of macro-tests

Benchmark

- decompression of `linux-2.6.12.tar.bz2`
- kernel compilation, default configuration
- removal of the whole kernel tree

Hooks exercised (same four previous modes)

- `security_inode_setattr()`
- `security_inode_unlink()`
- `security_inode_rmdir()`
- `security_inode_create()`
- `security_inode_mkdir()`



Results of macro-tests