Counter-Measures&Attacks

Yannick Chevalier
Université de Toulouse
CSA M1, Security

Historical background

Before the 1970s

- Computers only in physically secured environments
- Computers rebooted between each computation

Origin of security concerns

- Time-sharing : organisations with different needs use a same computer for economical purposes (Multics, Unix)
- Security goal : Time-sharing should be as secure as non-Time-sharing systems

Examples

- University and DoE/DoD use the same super-computer for simulation
- Nowadays : exactly the same concern :
  - Avionics : RTCA DO-255
  - Cloud computing

Historical background

Before the 1970s

- Computers only in physically secured environments
- Computers rebooted between each computation

Origin of security concerns

- Time-sharing : organisations with different needs use a same computer for economical purposes (Multics, Unix)
- Security goal : Time-sharing should be as secure as non-Time-sharing systems

Examples

- University and DoE/DoD use the same super-computer for simulation
- Nowadays : exactly the same concern :
  - Avionics : RTCA DO-255
  - Cloud computing

Historical background

Before the 1970s

- Computers only in physically secured environments
- Computers rebooted between each computation

Origin of security concerns

- Time-sharing : organisations with different needs use a same computer for economical purposes (Multics, Unix)
- Security goal : Time-sharing should be as secure as non-Time-sharing systems

Examples

- University and DoE/DoD use the same super-computer for simulation
- Nowadays : exactly the same concern :
  - Avionics : RTCA DO-255
  - Cloud computing

SEPARATION

SEPARATION INFORMAL STATEMENT

The execution of one program cannot influence in any way the execution of another program

EXTENSIVE DEFINITION

- A program in an infinite loop cannot preempt the processor
- A program with a memory leak cannot hold all the available memory
- In TS OS, the processor state has to be cleaned at each process change

SEPARATION

SEPARATION INFORMAL STATEMENT

The execution of one program cannot influence in any way the execution of another program

EXTENSIVE DEFINITION

- A program in an infinite loop cannot preempt the processor
- A program with a memory leak cannot hold all the available memory
- In TS OS, the processor state has to be cleaned at each process change

SEPARATION

SEPARATION INFORMAL STATEMENT

The execution of one program cannot influence in any way the execution of another program

EXTENSIVE DEFINITION

- A program in an infinite loop cannot preempt the processor
- A program with a memory leak cannot hold all the available memory
- In TS OS, the processor state has to be cleaned at each process change

SEPARATION

SEPARATION INFORMAL STATEMENT

The execution of one program cannot influence in any way the execution of another program

EXTENSIVE DEFINITION

- A program in an infinite loop cannot preempt the processor
- A program with a memory leak cannot hold all the available memory
- In TS OS, the processor state has to be cleaned at each process change

TRUSTED COMPUTING BASE (1/2)

ASSUME/GUARANTEE

When adding a new component to a system

- Except at the lowest level, use of other components based on assumptions
  - OS on the hardware
  - Applications/libraries on the OS and other applications/libraries
  - ...
- The new component guarantees (at some level of certainty) some properties :
  - no memory leaks,
  - bounds on execution time,
  - ...

Trusted Computing Base (2/2)

Security in CS

- Formalisms to state and prove the guarantees provided by a component based on some assumptions on other components
- Trusted computing base : parts of a system that are not analysed and just assumed to provide the stated guarantees
- Example : processor state reset :
  - register reset not in early version of Unix (unused registers were left as is)
  - cache never resetted after a context change (leading to lot of security attacks, *e.g.* Spectre)
- problem compounded in current processors by pipeline, speculative execution, and on-the-fly code optimisation

Security Solution : Static Configuration

Principle

Ressources are statically allocated to consumers

Security implications : Access Control

- Access control policy : definition of the possible allocation of resources to consumers
- Must be defined before the start of the machine
- Cannot be changed while it is running

Static AC

- Example : bounds on execution time, memory available, etc.
- Unmutable policy :

Mandatory Access Control

- Allows for a thorough analysis of a system

Security Solution : Static Configuration

Principle

Ressources are statically allocated to consumers

Security implications : Access Control

- Access control policy : definition of the possible allocation of resources to consumers
- Must be defined before the start of the machine
- Cannot be changed while it is running

Static AC

- Example : bounds on execution time, memory available, etc.
- Unmutable policy :

Mandatory Access Control

- Allows for a thorough analysis of a system

Security Solution : Static Configuration

Principle

Ressources are statically allocated to consumers

Security implications : Access Control

- Access control policy : definition of the possible allocation of resources to consumers
- Must be defined before the start of the machine
- Cannot be changed while it is running

Static AC

- Example : bounds on execution time, memory available, etc.
- Unmutable policy :

Mandatory Access Control

- Allows for a thorough analysis of a system

Security Solution : Static Configuration

Principle

Ressources are statically allocated to consumers

Security implications : Access Control

- Access control policy : definition of the possible allocation of resources to consumers
- Must be defined before the start of the machine
- Cannot be changed while it is running

Static AC

- Example : bounds on execution time, memory available, etc.
- Unmutable policy :

Mandatory Access Control

- Allows for a thorough analysis of a system

Security Solution : Static Configuration

Principle

Ressources are statically allocated to consumers

Security implications : Access Control

- Access control policy : definition of the possible allocation of resources to consumers
- Must be defined before the start of the machine
- Cannot be changed while it is running

Static AC

- Example : bounds on execution time, memory available, etc.
- Unmutable policy :

Mandatory Access Control

- Allows for a thorough analysis of a system

Security Solution : Static Configuration

Principle

Ressources are statically allocated to consumers

Security implications : Access Control

- Access control policy : definition of the possible allocation of resources to consumers
- Must be defined before the start of the machine
- Cannot be changed while it is running

Static AC

- Example : bounds on execution time, memory available, etc.
- Unmutable policy :

Mandatory Access Control

- Allows for a thorough analysis of a system

TIME SEPARATION

STATEMENT

The AC policy must specify the computation time allocated to each program

- Hard real-time systems
- Allocates a fraction of the processor cycles to each process
- In practice :
  - implemented in embedded and mission-critical systems for safety

    the ABS functionality cannot be affected when you start a new Ariana Grande song

Spatial Separation

Statement

Memory available to each application has to be allocated statically and be disjoint

- *a.k.a.* partitioning
- Implies no direct communication between applications (IPC system V), communication through the OS still possible
- In practice
  - RAM,HDD, SDD are sets of pages, and that set is partitioned, and each application receives one subset
  - No malloc/mmap, or a secure variant
  - Bounded stack
  - No pointers unless the Processor/OS guarantees that pointers never access a wrong page
- Very hard in practice
  - Need for a specific language (*e.g.* Lustre) or compiler
  - Registers and cache still have to shared on current processors

Access Control and Spatial Separation

- TL ;DR : very hard, but with some tolerance, achieved by current systems like Linux CGroups
- Incidentally, development driven by Cloud-hosting companies
- Known tolerances :
  - shared libraries
  - bounds on max memory available, but no hard partition
  - `root` still can do whatever he wants

Access Control and Spatial Separation

- TL ;DR : very hard, but with some tolerance, achieved by current systems like Linux CGroups
- Incidentally, development driven by Cloud-hosting companies
- Known tolerances :
  - shared libraries
  - bounds on max memory available, but no hard partition
  - `root` still can do whatever he wants

Access Control and Spatial Separation

- TL;DR : very hard, but with some tolerance, achieved by current systems like Linux CGroups
- Incidentally, development driven by Cloud-hosting companies
- Known tolerances :
  - shared libraries
  - bounds on max memory available, but no hard partition
  - `root` still can do whatever he wants

ACCESS CONTROL AND SPATIAL SEPARATION

- TL ;DR : very hard, but with some tolerance, achieved by current systems like Linux CGroups
- Incidentally, development driven by Cloud-hosting companies
- Known tolerances :
  - shared libraries
  - bounds on max memory available, but no hard partition
  - `root` still can do whatever he wants

ACCESS CONTROL AND SPATIAL SEPARATION

- TL ;DR : very hard, but with some tolerance, achieved by current systems like Linux CGroups
- Incidentally, development driven by Cloud-hosting companies
- Known tolerances :
  - shared libraries
  - bounds on max memory available, but no hard partition
  - `root` still can do whatever he wants

Access Control and Spatial Separation

- TL ;DR : very hard, but with some tolerance, achieved by current systems like Linux CGroups
- Incidentally, development driven by Cloud-hosting companies
- Known tolerances :
  - shared libraries
  - bounds on max memory available, but no hard partition
  - `root` still can do whatever he wants

Partitioning in Information Systems

In a computer network

- Tasks to be performed by a system are allocated to different computers based on their sensitivity
- The network is partitioned into zones for computers of similar security concerns
- Access Control on inter-zone exchanges (*a.k.a.* firewall)
- Example :
  - DMZ at the border between a corporate network and the outside (3 zones),
  - Firewalls to control packets between these zones

Note

- Outside of critical systems, there's no way to prove that all applications are secure
- First glimpse at Layered Security/Defence-in-Depth

Good AC systems

Neat Security

- Non-bypassable : impossible to act outside of the bounds of the AC system
- Evaluatable : possible communications can be assessed before deployment
- Always invoked : no future rights, the AC system must always be queried
- Tamperproof : the AC system cannot be changed while the system is running

- These concerns are shared with safe systems
- Possible on systems dedicated to a task, not in general (*e.g.*, no new file and no new user)
- The CS Security dilemna :
  - we know how to secure a system
  - we know how to have a usable system
  - we know how to build a cheap system

but no one has ever (or ever hopes to have) more than 2 out of 3

Good AC systems

Neat Security

- Non-bypassable : impossible to act outside of the bounds of the AC system
- Evaluatable : possible communications can be assessed before deployment
- Always invoked : no future rights, the AC system must always be queried
- Tamperproof : the AC system cannot be changed while the system is running

- These concerns are shared with safe systems
- Possible on systems dedicated to a task, not in general (*e.g.*, no new file and no new user)
- The CS Security dilemna :
  - we know how to secure a system
  - we know how to have a usable system
  - we know how to build a cheap system

but no one has ever (or ever hopes to have) more than 2 out of 3

Good AC systems

Neat Security

- Non-bypassable : impossible to act outside of the bounds of the AC system
- Evaluatable : possible communications can be assessed before deployment
- Always invoked : no future rights, the AC system must always be queried
- Tamperproof : the AC system cannot be changed while the system is running

- These concerns are shared with safe systems
- Possible on systems dedicated to a task, not in general (*e.g.*, no new file and no new user)
- The CS Security dilemna :
  - we know how to secure a system
  - we know how to have a usable system
  - we know how to build a cheap system

but no one has ever (or ever hopes to have) more than 2 out of 3

Good AC systems

NEAT Security

- Non-bypassable : impossible to act outside of the bounds of the AC system
- Evaluatable : possible communications can be assessed before deployment
- Always invoked : no future rights, the AC system must always be queried
- Tamperproof : the AC system cannot be changed while the system is running

- These concerns are shared with safe systems
- Possible on systems dedicated to a task, not in general (*e.g.*, no new file and no new user)
- The CS Security dilemna :
  - we know how to secure a system
  - we know how to have a usable system
  - we know how to build a cheap system

but no one has ever (or ever hopes to have) more than 2 out of 3

Good AC systems

Neat Security

- Non-bypassable : impossible to act outside of the bounds of the AC system
- Evaluatable : possible communications can be assessed before deployment
- Always invoked : no future rights, the AC system must always be queried
- Tamperproof : the AC system cannot be changed while the system is running

- These concerns are shared with safe systems
- Possible on systems dedicated to a task, not in general (*e.g.*, no new file and no new user)
- The CS Security dilemna :
  - we know how to secure a system
  - we know how to have a usable system
  - we know how to build a cheap system

but no one has ever (or ever hopes to have) more than 2 out of 3

GOOD AC SYSTEMS

NEAT SECURITY

- Non-bypassable : impossible to act outside of the bounds of the AC system
- Evaluatable : possible communications can be assessed before deployment
- Always invoked : no future rights, the AC system must always be queried
- Tamperproof : the AC system cannot be changed while the system is running

- These concerns are shared with safe systems
- Possible on systems dedicated to a task, not in general (*e.g.*, no new file and no new user)
- The CS Security dilemna :
  - we know how to secure a system
  - we know how to have a usable system
  - we know how to build a cheap system

but no one has ever (or ever hopes to have) more than 2 out of 3