Sécurité logicielle

Thomas Jensen

16 octobre 2005

Plan

Sécurité de systèmes d'information

Objectif : protéger ses biens (données, capacité de calcul) contre des utilisations malveillantes.

Trois propriétés à assurer :

- confidentialité,
- intégrité,
- disponibilité.

Politique de sécurité :

l'ensemble des dispositifs (lois, règles, pratiques) qui régit le traitement des informations.

Sécurité de systèmes d'information

Objectif : protéger ses biens (données, capacité de calcul) contre des utilisations malveillantes.

Trois propriétés à assurer :

- confidentialité,
- intégrité,
- disponibilité.

Politique de sécurité :

l'ensemble des dispositifs (lois, règles, pratiques) qui régit le traitement des informations.

Sécurité de systèmes d'information

Objectif : protéger ses biens (données, capacité de calcul) contre des utilisations malveillantes.

Trois propriétés à assurer :

- confidentialité,
- intégrité,
- disponibilité.

Politique de sécurité :

l'ensemble des dispositifs (lois, règles, pratiques) qui régit le traitement des informations.

Sécurité de systèmes d'information

Objectif : protéger ses biens (données, capacité de calcul) contre des utilisations malveillantes.

Trois propriétés à assurer :

- confidentialité,
- intégrité,
- disponibilité.

Politique de sécurité :

l'ensemble des dispositifs (lois, règles, pratiques) qui régit le traitement des informations.

Sécurité de systèmes d'information

Objectif : protéger ses biens (données, capacité de calcul) contre des utilisations malveillantes.

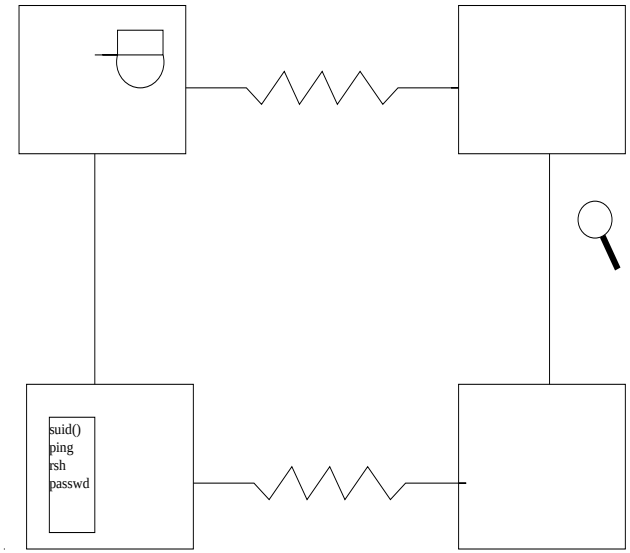
Trois propriétés à assurer :

- confidentialité,
- intégrité,
- disponibilité.

Politique de sécurité :

l'ensemble des dispositifs (lois, règles, pratiques) qui régit le traitement des informations.

La sécurité informatique - un problème “interdisciplinaire”



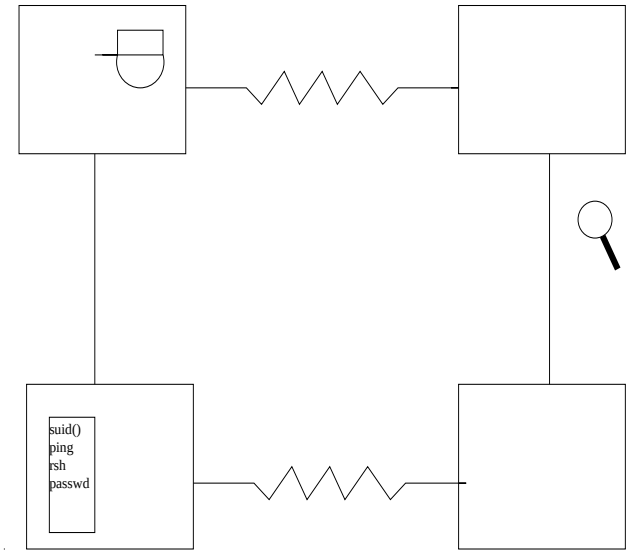
Une multitude de mécanismes de protection :

- communication par messages cryptés,
- gestionnaires de sécurité,
- moniteurs de l’activité réseau et système,
- logs d’audit

Problème fondamental

prouver que l’ensemble de dispositifs assure des propriétés globales de sécurité.

La sécurité informatique - un problème “interdisciplinaire”



Une multitude de mécanismes de protection :

- communication par messages cryptés,
- gestionnaires de sécurité,
- moniteurs de l’activité réseau et système,
- logs d’audit

Problème fondamental

prouver que l’ensemble de dispositifs assure des propriétés globales de sécurité.

Sécurité logicielle

Décrire

- l'architecture d'un logiciel,
- les protocoles cryptographiques utilisés,
- les composants de code Java ou autre

avec des formalismes ayant une sémantique formelle.

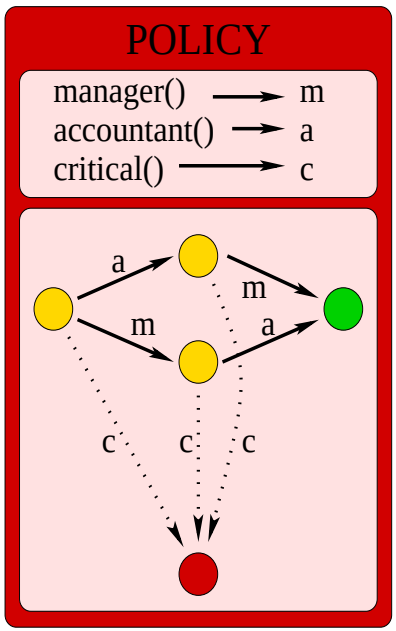
Vérification formelle d'une propriété de sécurité par

- analyse de programmes,
- vérification de modèles,
- preuve automatique.

Formalisations de politique de sécurité

Exprimer des propriétés de sûreté sur des traces d'exécutions, en logique temporelle ou par des automates de sécurité.

Exemple de politique : « une opération critique doit être valide par le comptable et par le directeur ».



Formalisation de la confidentialité

Non-interférence

Une donnée est **confidentielle** si elle n'interfère pas avec les calculs et les observations publiques.

> *Formellement : une variation dans des entrées secrètes n'est pas observable publiquement*

Observations possibles :

- résultat final
- communications intermédiaires
- variables publiques
- temps
- ...

Look at this if-statement ...

third rounds of a DES encryption operation. Many details of the DES operation are now visible. For example, the 28-bit DES key registers C and D are rotated once in round 2 (left arrow) and twice in round 3 (right arrows). In Figure 2, small variations between the rounds just can be perceived. Many of these discernable features are SPA weaknesses caused by conditional jumps based on key bits and computational intermediates.

Figure 3 shows even higher resolution views of the trace showing power consumption through two regions, each of seven clock cycles at 3.5714 MHz. The visible variations between clock cycles result primarily from differences in the power consumption of different microprocessor instructions. The upper trace in Figure 3 shows the execution path through an SPA feature where a jump instruction is performed, and the lower trace shows a case where the jump is not taken. The point of divergence is at clock cycle 6 and is clearly visible.

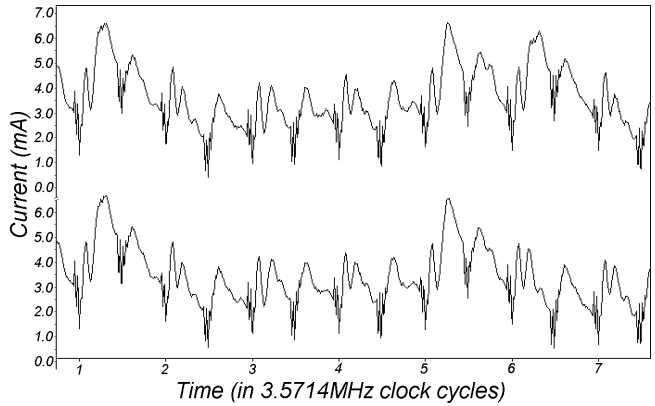


Figure 3: SPA trace showing individual clock cycles.

Because SPA can reveal the sequence of instructions executed, it can be used to break cryptographic implementations in which the execution path depends on the data being processed. For example:

DES key schedule: The DES key schedule computation involves rotating 28-bit key registers. A conditional branch is commonly used to check the bit shifted off the end so that "1" bits can be wrapped around. The resulting power consumption traces for a "1" bit and a "0" bit will contain different SPA features if the execution paths take different branches for each.

DES permutations: DES implementations perform a variety of bit permutations. Conditional branching in software or microcode can cause significant

Plan

Défis en sécurité logicielle

- non-interférence et *down-grading*
- formaliser intégrité et disponibilité,
- analyse modulaire de programmes, pour
  - traiter du code mobile,
  - le passage à l'échelle (Windows).
- formalisations de politiques de sécurité (automates, logiques temporelles,. . .) permettant la combinaison et la vérification efficace des politiques.
- communiquer une preuve de sécurité
  - certificats,
  - *proof-carrying code*

Défis en sécurité logicielle

- non-interférence et *down-grading*
- formaliser intégrité et disponibilité,
- analyse modulaire de programmes, pour
  - traiter du code mobile,
  - le passage à l'échelle (Windows).
- formalisations de politiques de sécurité (automates, logiques temporelles,. . .) permettant la combinaison et la vérification efficace des politiques.
- communiquer une preuve de sécurité
  - certificats,
  - *proof-carrying code*

Défis en sécurité logicielle

- non-interférence et *down-grading*
- formaliser intégrité et disponibilité,
- analyse modulaire de programmes, pour
  - traiter du code mobile,
  - le passage à l'échelle (Windows).
- formalisations de politiques de sécurité (automates, logiques temporelles,. . .) permettant la combinaison et la vérification efficace des politiques.
- communiquer une preuve de sécurité
  - certificats,
  - *proof-carrying code*

Défis en sécurité logicielle

- non-interférence et *down-grading*
- formaliser intégrité et disponibilité,
- analyse modulaire de programmes, pour
  - traiter du code mobile,
  - le passage à l'échelle (Windows).
- formalisations de politiques de sécurité (automates, logiques temporelles,. . .) permettant la combinaison et la vérification efficace des politiques.
- communiquer une preuve de sécurité
  - certificats,
  - *proof-carrying code*

Défis en sécurité logicielle

- non-interférence et *down-grading*
- formaliser intégrité et disponibilité,
- analyse modulaire de programmes, pour
  - traiter du code mobile,
  - le passage à l'échelle (Windows).
- formalisations de politiques de sécurité (automates, logiques temporelles,. . .) permettant la combinaison et la vérification efficace des politiques.
- communiquer une preuve de sécurité
  - certificats,
  - *proof-carrying code*

Communiquer une preuve