SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Présentation
Validy Technology

Protection contre le piratage et le sabotage informatique

Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Trois aspects de la protection de logiciel

- Empêcher l'utilisation d'un logiciel quand on ne possède pas une licence valide

⇒ défense contre les copies non autorisées

- Empêcher la modification d'un logiciel

⇒ défense de l'intégrité

- Empêcher la compréhension du code d'un logiciel

⇒ défense contre la rétro-ingénierie

Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Protection d'un logiciel contre la copie non autorisée

- Protection additive
  - ajouter des fonctions complexes pour vérifier la validité de la licence (utilisation de la cryptographie, masquage,etc)
    - ⇒ Le pirate peut contourner les fonctions de protection
- Protection soustractive
  - Utiliser un “ jeton ” physique sécurisé pour “ cacher ” une partie du programme
    - ⇒ Le pirate doit “ réinventer ” la partie manquante

Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Conditions d'une protection soustractive effective

- La partie cachée doit être **essentielle** au programme
  – Le programme doit être inutilisable sans la partie cachée
- La partie cachée doit être suffisamment complexe pour être **non simulable**
  – L'observation des échanges entre hôte et jeton ne doit pas permettre d'en déduire un simulateur
- Les **performances** du programme protégés doivent rester **acceptables**
  – Le jeton ne doit exécuter qu'une partie du programme proportionnelle à sa puissance de calcul et/ou aux performances du bus de communication

Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Principe de la protection “Validy Technology”

- Conserver des variables d’un programme dans un jeton
  - les variables cachées sont modifiées dans le jeton
  - des valeurs dérivées des variables internes au jeton sont renvoyées au programme lorsqu'il en a besoin
- Pour cela utiliser le jeton comme un “coprocesseur”
  - les variables cachées sont transformées par des instructions cryptées envoyées au jeton
  - le jeton détecte les attaques visant à comprendre ces instructions
- En cas d’attaque le jeton peut cesser de fonctionner
  - dans ce cas le programme s’arrête

Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Le Jeton

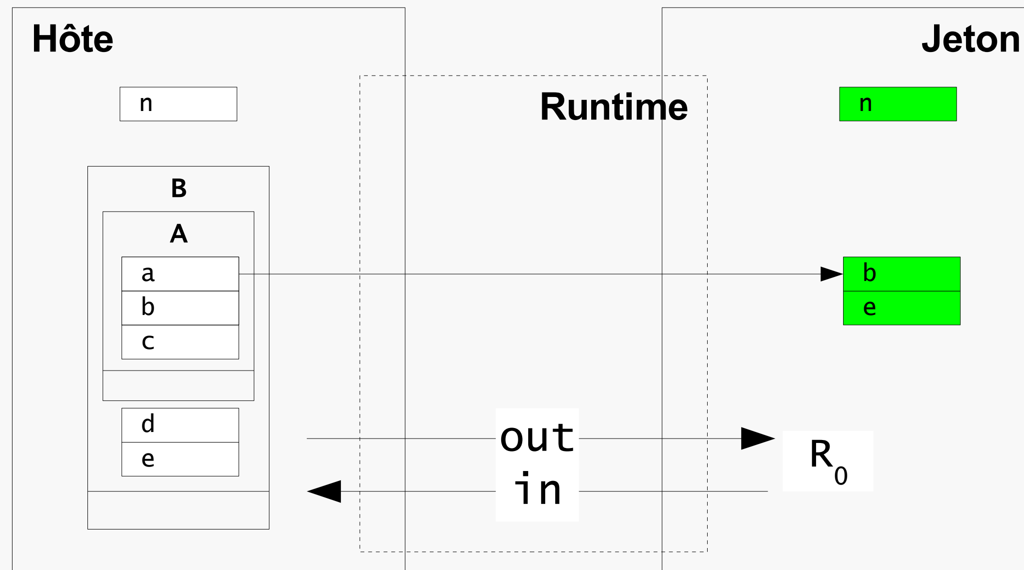
- Se présente sous la forme d'une carte à puce, d'une clé USB, d'une puce spécialisée…
- C'est un petit ordinateur dans un coffre fort
  - Dérivé des applications bancaires et téléphoniques
  - Très difficilement attaquable
- Possède un processeur et de la mémoire
  - Processeur 6 à 50 MHz
  - Mémoire rémanente : garde le programme et des données, même hors tension
  - Mémoire volatile : données de travail 3 à 16 kO
  - Accélérateur cryptographique pour crypter/décrypter
- Sera toujours beaucoup moins performant que l'ordinateur principal

Copyright © 2006 SA Validy Tous droits réservés

VALIDY
SECURITE INFORMATIQUE ET ECONOMIQUE
www.validy.com
www.validy-licensing.com

Variables

- certaines valeurs du programme sont allouées dans la mémoire vive du jeton plutôt que dans celle de l'hôte
- deux opérations du runtime (in/out) permettent d'échanger des valeurs avec le jeton



Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Fonctions élémentaires

- Les valeurs déplacées sont transformées dans le jeton par des fonctions élémentaires
- Le jeton implémente une machine virtuelle à registres 32 bits avec un jeu d'instruction RISC
  - Accès à la mémoire (load/store)
  - Opérations logiques et arithmétiques
  - Opérations spécifiques (tags)
  - Pas de contrôle de flot
- Les instructions exécutées par le jeton sont choisies par l'outil de protection en partant de celles qui chargent ou sauvent les variables cachées puis en étendant la sélection.
- Les instructions exécutées par l'hôte et par le jeton sont entrelacées dans le code du programme.

Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Renommage

- Les instructions à exécuter sont inclues dans le programme principal et envoyées au jeton pendant l'exécution.
- Pour être inintelligibles, les instructions sont renommées et le jeton sait inverser cette transformation.
- Le cryptage est une implémentation possible du renommage.

Représentation intermédiaire (outil de protection)

```
staticinvoke void com.validy.technology.runtime.Token::exe(long)
              (VM_SUB [186] $k2 $k4 [85] [0] $k3 [225] );
```

Bytecode Java

```
81: ldc2_w   #240; //long -2507418968542456500l
84: invokestatic #37; //Method com/validy/technology/runtime/Token.exe:(J)V
```

Copyright © 2006 SA Validy Tous droits réservés

Détection et coercition

- L'objectif est de détecter les manipulations du flux d'instructions visant à comprendre progressivement leur rôle (suppression, inversion, redoublement).
- L'outil encode dans les instructions des assertions sur le flot de données du programme.
- Le jeton vérifie pendant l'exécution que le flux reçu satisfait les assertions.
- En cas de violation, le jeton peut réagir :
  - Par un arrêt simple
  - Par un arrêt et un blocage temporaire
  - Par l'effacement de la clef de renommage

Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Tags - Définition

- Chaque instruction jeton est identifiée par son tag (1 octet)
- Chaque registre et chaque case mémoire du jeton comprend à la fois une valeur et le tag de l'instruction qui a produit cette valeur

```
Out 5
VM_MOV [27] R1 R0
VM_LDI [212]R2 23
VM_MUL [129]R3 R2 R1
```

	valeur	tag
R_0	5	?
R_1	5	27
R_2	23	212
R_3	115	129

Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Tags - vérification

- Le compilateur détermine pour chaque utilisation de la valeur d'un registre les définitions qui peuvent l'atteindre
- La liste des tags des instructions correspondant à ces définitions est associée à l'utilisation
- Le jeton vérifie que le tag des registres utilisés correspond à l'une des valeurs codées dans l'instruction

VM_ADD [186] R_2 R_4 [131] [57] R_2 [29]

VM_TAGP [244] R_2 [73] [168] [128] [0] [0]

VM_TAG [1] R_2 [244] [55] [18] [189] [30]

Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Tags - Exemple

```
R3 = 12
if () {
    R4 = R2 - R1;
} else {
    R4 = -1;
}
R1 = R4 + R3
```

```
VM_LDI [32] R3 12
if () {
    VM_SUB [123] R2 R2 [12] [203] R1 [45]
} else {
    VM_LDI [174] R2 -1
}
VM_ADD [83] R1 R2 [174] [123] R3 [32]
```

```
LDI [32]  R3 12
ADD [123] R2 R2 [12] [203] R1 [45]
ADD [83]  R1 R2 [174] [123] R3 [32]
```

	valeur	tag
R1	4	45
R2	7	203
R3	12	32

Copyright © 2006 SA Validy Tous droits réservés

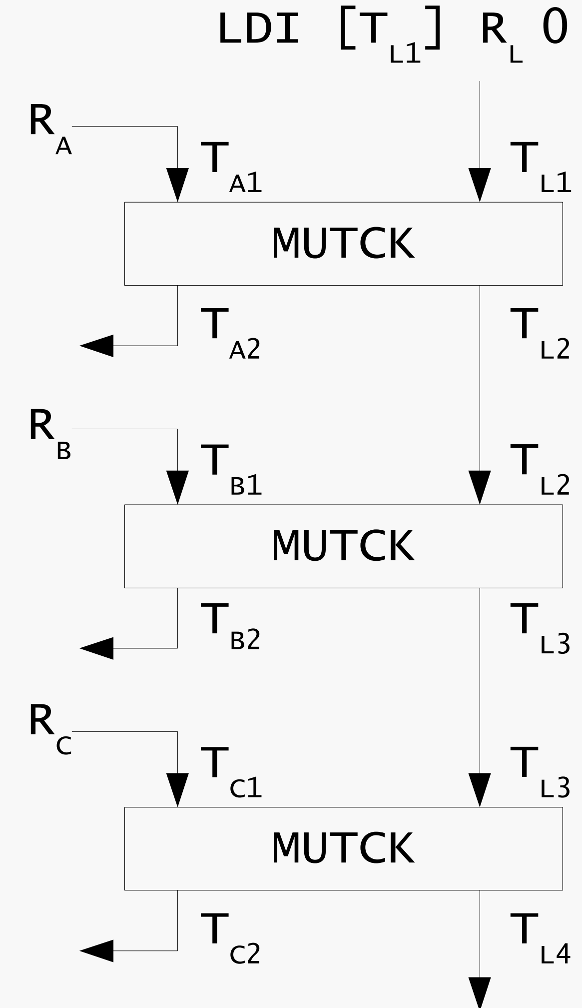
VALIDY
SECURITE INFORMATIQUE ET ECONOMIQUE
www.validy.com
www.validy-licensing.com

Protection mutuelle

- Le mécanisme des tags permet de lier artificiellement deux calculs indépendants

  VM_MUTCK R_1 [T_{o1}] [T_{i1}] R_2 [T_{o2}] [T_{i2}]

  - Si R_1 possède le tag T_{i1} **et** si R_2 possède le tag T_{i2}, l'opération réussit, R_1 prend le tag T_{o1} et R_2 le tag T_{o2}
  - Sinon la machine virtuelle a détecté une erreur
  - La valeur des registres n'est pas affectée
- Par l'intermédiaire d'un registre réservé R_L, on peut relier une suite de calculs



Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

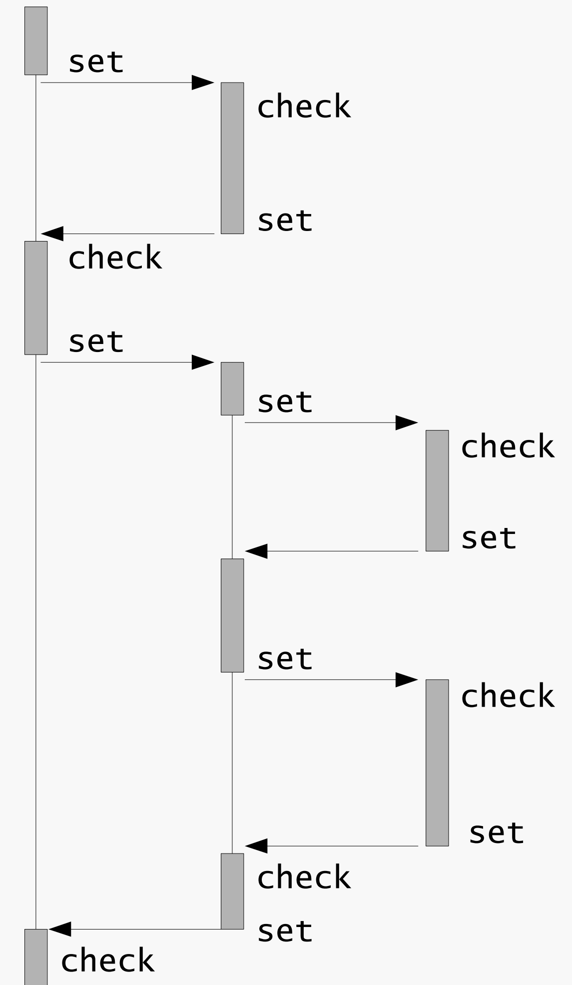
Protections additives

- Relier entre eux les calculs du programme original a peu de sens
- Mais des protections additives liées au code protégé original deviennent inamovibles
- Cela ouvre de nombreuses possibilités :
  - La vérification d'invariants
  - Des compteurs infalsifiables
  - Le contrôle d'accès aux ressources cryptographiques du jeton
  - L'ajout de contraintes sur le graphe d'appel des fonctions

Copyright © 2006 SA Validy Tous droits réservés

Contraintes sur le graphe d'appel

- Les sites d'appel sont liés aux fonctions appelées par la valeur du tag d'un registre de liaison
- Les instructions de liaison (set tag/ check tag) sont reliées aux autres calculs du jeton
- Le calcul des groupes de fonctions et de sites d'appel associés dépend du langage (interface, méthode virtuelle, pointeur de fonction)
- Impossible de supprimer ou d'ajouter un appel aux fonctions ainsi protégées



Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Virtualisation

- Espace d'adressage « infini »
  - Le jeton gère une mémoire virtuelle paginée
  - L'hôte sert de zone d'échange
  - Le jeton crypte les pages avant de les transmettre à l'hôte pour stockage
- Multithread/process
  - Le runtime de l'hôte étiquette les instructions avec l'identifiant du thread/process courant pour que le jeton les exécutent dans le bon contexte
  - Le jeton exécutent les instructions dans l'ordre où elles sont émises pour préserver la sémantique du programme

Copyright © 2006 SA Validy Tous droits réservés

Mise en oeuvre

- Marquage des valeurs à déplacer et des fonctions à protéger par le développeur (dans le futur, par l'outil)
- Sélection des instructions déplacées, substitution et renommage par l'outil de protection ou le compilateur
- Tests du programme protégé avec un jeton simulé ou réel
- Distribution du programme transformé (sans restrictions) et des jetons représentant les licences d'utilisation
- Distribution de nouvelles versions protégées par la même clef de renommage (maintenance) ou par une clef différente (version majeure)

Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Implémentation actuelle

- Recompilateur de bytecode Java et .NET avec marquage manuel par des attributs
- Runtime mixte (bytecode + natif) sous Windows
- Bus USB 1.1
- Jeton Atmel AT90SC6464C (8bits, 6MHz, 3kO RAM)
- Machine virtuelle implémentée en C et assembleur, pas de virtualisation, renommage par 3DES, (30 kilo instructions/s max, 10 kI/s en utilisation réelle)
- Simulateur du jeton sous Windows

Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Conclusion

- La matérialisation de la licence par un jeton physique permet le contrôle de la diffusion du logiciel
- Le pirate ne peut plus essayer d'apprendre en utilisant une copie du programme et doit utiliser le logiciel avec licence
- Les tentatives de compréhension de la protection sont détectées par le jeton
- Validy Technology est indépendant du système d'exploitation, la protection fonctionne aussi sur des systèmes dédiés avec microcontrôleur dépourvus de système d'exploitation
- Il n'est pas nécessaire de sécuriser le système d'exploitation et/ou le matériel pour exécuter le logiciel de manière sûre

Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Conclusion

- La sécurité de Validy Technology n'est pas basée sur le secret de la solution. Tous les principes peuvent être audités et connus de tous y compris des pirates. Seule la clef de renommage est secrète.
- Validy Technology n'est pas un système de sécurité centralisé : le système n'a pas de point faible unique. Si un jeton est compromis, seul le logiciel protégé par ce jeton n'est plus protégé.

Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Présentation Validy Technology

Protection contre le piratage et le sabotage informatique

Copyright © 2006 SA Validy Tous droits réservés

Trois aspects de la protection de logiciel

- Empêcher l'utilisation d'un logiciel quand on ne possède pas une licence valide

⇒ défense contre les copies non autorisées

- Empêcher la modification d'un logiciel

⇒ défense de l'intégrité

- Empêcher la compréhension du code d'un logiciel

⇒ défense contre la rétro-ingénierie

Copyright © 2006 SA Validy Tous droits réservés

Copy protection, tamper resistance, reverse engineering

Il existe d'autres aspects (white box cryptography)

VT s'applique à ces trois aspects avec des limites possibles liées aux performances pour le reverse engineering (algorithmes de calcul intensif)

Protection d'un logiciel contre la copie non autorisée

- Protection additive
  - ajouter des fonctions complexes pour vérifier la validité de la licence (utilisation de la cryptographie, masquage,etc)
    - ⇒ Le pirate peut contourner les fonctions de protection
- Protection soustractive
  - Utiliser un “ jeton ” physique sécurisé pour “ cacher ” une partie du programme
    - ⇒ Le pirate doit “ réinventer ” la partie manquante

Copyright © 2006 SA Validy Tous droits réservés

Cette distinction s'applique essentiellement à la protection contre la copie.

Conditions d'une protection soustractive effective

- La partie cachée doit être **essentielle** au programme
  - Le programme doit être inutilisable sans la partie cachée
- La partie cachée doit être suffisamment complexe pour être **non simulable**
  - L'observation des échanges entre hôte et jeton ne doit pas permettre d'en déduire un simulateur
- Les **performances** du programme protégés doivent rester **acceptables**
  - Le jeton ne doit exécuter qu'une partie du programme proportionnelle à sa puissance de calcul et/ou aux performances du bus de communication

Copyright © 2006 SA Validy Tous droits réservés

Le programme doit être inutilisable si on remplace les valeurs renvoyées par le jeton par une constante.

La taille de la partie cachée est limitée par le rapport de puissance jeton/hôte et les caractéristiques du bus de communication (débit/latence).

Une solution qui essaye de déplacer une fonction de haut niveau appelée par RPC dans le jeton se heurte à ces problèmes : existence de fonctions suffisamment complexes pour que l'observation d'une série d'appels ne permettent pas de reconstituer leur contenu tout en restant suffisamment simples pour que leur exécution dans le jeton soit possible et ne ralentisse pas trop le programme.

Le problème consiste donc à trouver une partition satisfaisante du programme.

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Principe de la protection “Validy Technology”

- Conserver des variables d’un programme dans un jeton
  - les variables cachées sont modifiées dans le jeton
  - des valeurs dérivées des variables internes au jeton sont renvoyées au programme lorsqu'il en a besoin
- Pour cela utiliser le jeton comme un “coprocesseur”
  - les variables cachées sont transformées par des instructions cryptées envoyées au jeton
  - le jeton détecte les attaques visant à comprendre ces instructions
- En cas d’attaque le jeton peut cesser de fonctionner
  - dans ce cas le programme s’arrête

Copyright © 2006 SA Validy Tous droits réservés

Après la détection d'une attaque, les instructions cryptées ne sont plus exécutables sans réinitialiser le jeton ce qui entraîne la perte de l'état des variables cachées.

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Le Jeton

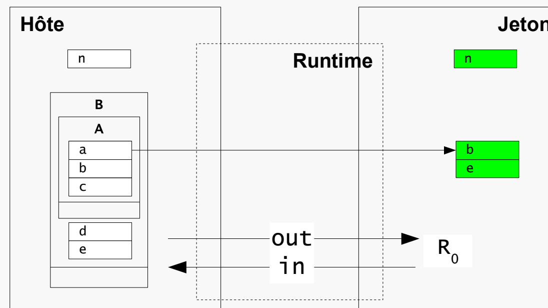
- Se présente sous la forme d'une carte à puce, d'une clé USB, d'une puce spécialisée…
- C'est un petit ordinateur dans un coffre fort
  - Dérivé des applications bancaires et téléphoniques
  - Très difficilement attaquable
- Possède un processeur et de la mémoire
  - Processeur   6 à 50 MHz
  - Mémoire rémanente : garde le programme et des données, même hors tension
  - Mémoire volatile : données de travail   3 à 16 kO
  - Accélérateur cryptographique pour crypter/décrypter
- Sera toujours beaucoup moins performant que l'ordinateur principal

Copyright © 2006 SA Validy Tous droits réservés

VALIDY
SECURITE INFORMATIQUE ET ECONOMIQUE
www.validy.com
www.validy-licensing.com

Variables

- certaines valeurs du programme sont allouées dans la mémoire vive du jeton plutôt que dans celle de l'hôte
- deux opérations du runtime (in/out) permettent d'échanger des valeurs avec le jeton



Copyright © 2006 SA Validy Tous droits réservés

Le choix des variables déplacées peut être réalisé par le développeur ou par l'outil de protection. Les variables d'état qui influent sur le contrôle de flot du programme sont de bons candidats.

La manière dont la mémoire du jeton est alloué et référencé par le programme protégé dépend des caractéristiques du langage utilisé et en particulier des classes de stockage (statique/dynamique/pile/registre).

L'échange de valeurs entre hôte et jeton (fonction out et in du runtime) se fait par l'intermédiaire d'un registre réservé du jeton (R_0).

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Fonctions élémentaires

- Les valeurs déplacées sont transformées dans le jeton par des fonctions élémentaires
- Le jeton implémente une machine virtuelle à registres 32 bits avec un jeu d'instruction RISC
  - Accès à la mémoire (load/store)
  - Opérations logiques et arithmétiques
  - Opérations spécifiques (tags)
  - Pas de contrôle de flot
- Les instructions exécutées par le jeton sont choisies par l'outil de protection en partant de celles qui chargent ou sauvent les variables cachées puis en étendant la sélection.
- Les instructions exécutées par l'hôte et par le jeton sont entrelacées dans le code du programme.

Copyright © 2006 SA Validy Tous droits réservés

L'unité d 'exécution dans le jeton est une instruction élémentaire avec deux registres source et un registre destination.

Les fonctions élémentaires exécutées dans le jeton sont des fonctions qui étaient avant transformation exécutées par le programme sur l'hôte.

Aux frontières entre instructions effectuées dans l'hôte et dans le jeton, on peut trouver des échanges de valeur par in ou out.

A la différence d'une approche RPC dans laquelle toute la fonction déportée doit pouvoir être exécutée par le jeton, l'outil de protection à la flexibilité de ne pas déplacer les instructions qui ne sont pas exécutables dans le jeton.

Le contrôle de flot est effectué par le programme sur l'hôte. Le jeton reçoit un flux d'instructions. Il exécute toutes les instruction reçues dans leur ordre d'arrivée.

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Renommage

- Les instructions à exécuter sont inclues dans le programme principal et envoyées au jeton pendant l'exécution.
- Pour être inintelligibles, les instructions sont renommées et le jeton sait inverser cette transformation.
- Le cryptage est une implémentation possible du renommage.

Représentation intermédiaire (outil de protection)

```
staticinvoke void com.validy.technology.runtime.Token::exe(long)
                (VM_SUB [186] $k2 $k4 [85] [0] $k3 [225] );
```

Bytecode Java

```
81: ldc2_w   #240; //long -25074189685424565001
84: invokestatic #37; //Method com/validy/technology/runtime/Token.exe:(J)V
```

Copyright © 2006 SA Validy Tous droits réservés

La fonction exe du runtime transmet au jeton une instruction pour exécution.

L'exécution de cette instruction (ainsi qu'un appel à la fonction out) n'a pas besoin d'être synchronisé avec le programme hôte. exe retourne avant l'exécution réelle de l'instruction dans le jeton. Seules les appels à la fonction in qui renvoie une valeur du jeton vers l'hôte sont bloquantes.

Détection et coercition

- L'objectif est de détecter les manipulations du flux d'instructions visant à comprendre progressivement leur rôle (suppression, inversion, redoublement).
- L'outil encode dans les instructions des assertions sur le flot de données du programme.
- Le jeton vérifie pendant l'exécution que le flux reçu satisfait les assertions.
- En cas de violation, le jeton peut réagir :
  - Par un arrêt simple
  - Par un arrêt et un blocage temporaire
  - Par l'effacement de la clef de renommage

Copyright © 2006 SA Validy Tous droits réservés

Le renommage par cryptage rend déjà les modifications d'une instruction seule inutile car le changement d'un bit de l'instruction cryptée change une grande partie de l'instruction décryptée.

Les assertions sont comprises dans l'instruction et donc aussi protégées par le renommage.

Le niveau de réaction est ajustable en fonction des besoins de l'application protégée.

Tags - Définition

- Chaque instruction jeton est identifiée par son tag (1 octet)
- Chaque registre et chaque case mémoire du jeton comprend à la fois une valeur et le tag de l'instruction qui a produit cette valeur

```
Out 5
VM_MOV [27] R1 R0
VM_LDI [212]R2 23
VM_MUL [129]R3 R2 R1
```

	valeur	tag
R_0	5	?
R_1	5	27
R_2	23	212
R_3	115	129

Copyright © 2006 SA Validy Tous droits réservés

L'identifiant est choisi par l'outil de protection. Il n'est pas unique (un programme peut contenir plus de 256 instructions différentes).

Deux instructions, même si elles sont identiques en terme d'opcode et de registres de source et de destination n'apparaîtront jamais identiques dans le code protégé.

Le stockage du tag est réalisé automatiquement par la machine virtuelle. L'instruction STORE change le tag de l'espace mémoire visé.

Tags - vérification

- Le compilateur détermine pour chaque utilisation de la valeur d'un registre les définitions qui peuvent l'atteindre
- La liste des tags des instructions correspondant à ces définitions est associée à l'utilisation
- Le jeton vérifie que le tag des registres utilisés correspond à l'une des valeurs codées dans l'instruction

VM_ADD [186] R_2 R_4 [131] [57] R_2 [29]

VM_TAGP [244] R_2 [73] [168] [128] [0] [0]

VM_TAG [1] R_2 [244] [55] [18] [189] [30]

Copyright © 2006 SA Validy Tous droits réservés

Le calcul effectué par le compilateur correspond à l'analyse des « reaching definitions ». La même information est disponible implicitement si la représentation intermédiaire utilisée est en forme SSA (Static Single Assignment).

Les instructions à un registre destination et deux registres sources sont asymétriques dans le nombre de tags vérifiés.

On peut vérifier un nombre quelconque de tags en combinant des instructions TAGP (tag and propagate) et TAG.

Si le compilateur ne peut pas déterminer l'ensemble des tags possibles, il utilise la valeur 0 qui accepte toutes les valeurs possibles (uniquement en première position).

Si il y a plus d'emplacements pour les tags acceptés que de tags possibles, on complète avec des 0 (ou des valeurs aléatoires).

Tags - Exemple

```
R3 = 12                VM_LDI [32] R3 12
if () {                if () {
    R4 = R2 - R1;          VM_SUB [123] R2 R2 [12] [203] R1 [45]
} else {               } else {
    R4 = -1;               VM_LDI [174] R2 -1
}                      }
R1 = R4 + R3           VM_ADD [83] R1 R2 [174] [123] R3 [32]
```

```
LDI [32]   R3 12
ADD [123]  R2 R2 [12] [203] R1 [45]
ADD [83]   R1 R2 [174] [123] R3 [32]
```

	valeur	tag
R1	4	45
R2	7	203
R3	12	32

Copyright © 2006 SA Validy Tous droits réservés

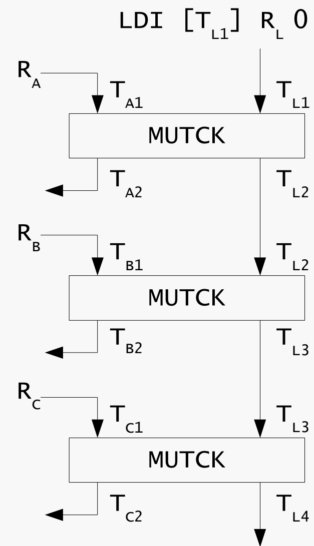
Deux séquences sont valides (VM_LDI, VM_SUB, VM_ADD) et (VM_LDI, VM_LDI, VM_ADD).

La qualité de la détection dépend en partie de celle de l'allocateur de registres. Plus les registres sont réutilisés, plus on peut détecter d'anomalies.

VALIDY
SECURITE INFORMATIQUE ET ECONOMIQUE
www.validy.com
www.validy-licensing.com

Protection mutuelle

- Le mécanisme des tags permet de lier artificiellement deux calculs indépendants
  VM_MUTCK R_1 [T_{o1}] [T_{i1}] R_2 [T_{o2}] [T_{i2}]
  – Si R_1 possède le tag T_{i1} **et** si R_2 possède le tag T_{i2}, l'opération réussit, R_1 prend le tag T_{o1} et R_2 le tag T_{o2}
  – Sinon la machine virtuelle a détecté une erreur
  – La valeur des registres n'est pas affectée
- Par l'intermédiaire d'un registre réservé R_L, on peut relier une suite de calculs



Copyright © 2006 SA Validy Tous droits réservés

L'instruction MUTCK peut-être implémentée en terme d'instructions de plus bas niveau: CKMV (check and move) voir XOR.

Si l'un des calculs liés par le registre R_L est obligatoire donc inamovible, les autres le deviennent également.

Protections additives

- Relier entre eux les calculs du programme original a peu de sens
- Mais des protections additives liées au code protégé original deviennent inamovibles
- Cela ouvre de nombreuses possibilités :
  - La vérification d'invariants
  - Des compteurs infalsifiables
  - Le contrôle d'accès aux ressources cryptographiques du jeton
  - L'ajout de contraintes sur le graphe d'appel des fonctions

Copyright © 2006 SA Validy Tous droits réservés

Les protections additives ne renvoient jamais de résultat à l'hôte. Leur seul effet est de provoquer une réaction en cas d'attaque détectée. Elles n'introduisent donc pas de synchronisations supplémentaires.

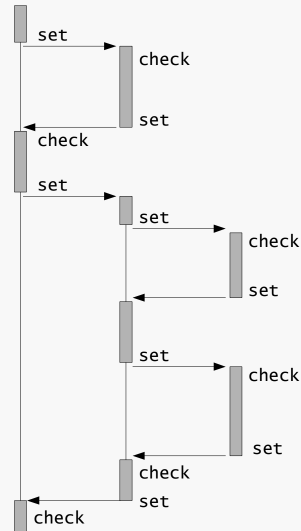
Si le jeton stocke une clef secrète ou privée pour une application cliente, la réponse aux challenges cryptographiques du serveur peut être conditionnée à la bonne exécution de l'application.

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Contraintes sur le graphe d'appel

- Les sites d'appel sont liés aux fonctions appelées par la valeur du tag d'un registre de liaison
- Les instructions de liaison (set tag/ check tag) sont reliées aux autres calculs du jeton
- Le calcul des groupes de fonctions et de sites d'appel associés dépend du langage (interface, méthode virtuelle, pointeur de fonction)
- Impossible de supprimer ou d'ajouter un appel aux fonctions ainsi protégées



Copyright © 2006 SA Validy Tous droits réservés

Les tags utilisés pour un groupe de fonctions donné sont dérivés de la signature de la fonction. On peut donc protéger un programme « par parties », par exemple un programme principal et des plugins.

Toutes les fonctions ne peuvent pas être protégées. Il faut que toutes les implémentations et les sites d'appel soient tous accessibles à l'outil de protection. Ce n'est pas forcément le cas : interfaces définies et implémentées dans une librairie standard/externe, utilisation de pointeurs de fonction que l'outil ne sait pas analyser, etc.

Virtualisation

- Espace d'adressage « infini »
  - Le jeton gère une mémoire virtuelle paginée
  - L'hôte sert de zone d'échange
  - Le jeton crypte les pages avant de les transmettre à l'hôte pour stockage
- Multithread/process
  - Le runtime de l'hôte étiquette les instructions avec l'identifiant du thread/process courant pour que le jeton les exécutent dans le bon contexte
  - Le jeton exécutent les instructions dans l'ordre où elles sont émises pour préserver la sémantique du programme

Copyright © 2006 SA Validy Tous droits réservés

Le jeton crypte les pages avec une clef choisie aléatoirement pour chaque exécution du programme.

Le rôle du runtime est :

- d'isoler complètement le programme protégé des problèmes de virtualisation en collaborant avec le jeton pour gérer la mémoire virtuelle,

- d'implémenter l'allocation de mémoire (malloc/free) sur l'hôte.

- d'optimiser les communications entre l'hôte et le jeton en ne transmettant que les changements de contexte.

Les langages à « garbage collector » rendent la virtualisation importante car la libération réelle de la mémoire du jeton peut-être retardée tant qu'il n'y a pas de pression sur la mémoire côté hôte.

Des implémentations simplifiées sont possibles pour certaines applications (besoins limités et connus en mémoire, pas d'allocation dynamique).

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Mise en oeuvre

- Marquage des valeurs à déplacer et des fonctions à protéger par le développeur (dans le futur, par l'outil)
- Sélection des instructions déplacées, substitution et renommage par l'outil de protection ou le compilateur
- Tests du programme protégé avec un jeton simulé ou réel
- Distribution du programme transformé (sans restrictions) et des jetons représentant les licences d'utilisation
- Distribution de nouvelles versions protégées par la même clef de renommage (maintenance) ou par une clef différente (version majeure)

Copyright © 2006 SA Validy Tous droits réservés

Le marquage manuel représente l'état actuel des développements et n'est pas une limite intrinsèque

Plusieurs technique d'analyse de programmes peuvent s'appliquer pour guider et/ou automatiser le choix: profiling, program slicing...

Les opérations de renommage d'instruction peuvent être effectuées dans un jeton de compilation pour ne pas exposer la clef de renommage.

Du point de vue du développement, la distinction entre programme non protégé et programme protégé est similaire à la distinction entre version de debug et version release.

Le choix de la clef de renommage offre une grande flexibilité dans la distribution des applications: plusieurs applications pour une clef, une clef par application, version majeure et mineure, version personnalisée pour un client.

Implémentation actuelle

- Recompilateur de bytecode Java et .NET avec marquage manuel par des attributs
- Runtime mixte (bytecode + natif) sous Windows
- Bus USB 1.1
- Jeton Atmel AT90SC6464C (8bits, 6MHz, 3kO RAM)
- Machine virtuelle implémentée en C et assembleur, pas de virtualisation, renommage par 3DES, (30 kilo instructions/s max, 10 kI/s en utilisation réelle)
- Simulateur du jeton sous Windows

Copyright © 2006 SA Validy Tous droits réservés

Les langages à bytecode simplifient le développement de l'outil de protection mais la technique s'applique aux langages compilés en général (C, C++).

Les manipulations à effectuer correspondent bien au niveau de représentation intermédiaire utilisé par les passes d'optimisation d'un compilateur.

Plusieurs jetons qui existent aujourd'hui permettraient une implémentation plus performante (16/32 bits, 16 kO RAM, USB 2)

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Conclusion

- La matérialisation de la licence par un jeton physique permet le contrôle de la diffusion du logiciel
- Le pirate ne peut plus essayer d'apprendre en utilisant une copie du programme et doit utiliser le logiciel avec licence
- Les tentatives de compréhension de la protection sont détectées par le jeton
- Validy Technology est indépendant du système d'exploitation, la protection fonctionne aussi sur des systèmes dédiés avec microcontrôleur dépourvus de système d'exploitation
- Il n'est pas nécessaire de sécuriser le système d'exploitation et/ou le matériel pour exécuter le logiciel de manière sûre

Copyright © 2006 SA Validy Tous droits réservés

SECURITE INFORMATIQUE ET ECONOMIQUE

www.validy.com
www.validy-licensing.com

Conclusion

- La sécurité de Validy Technology n’est pas basée sur le secret de la solution. Tous les principes peuvent être audités et connus de tous y compris des pirates. Seule la clef de renommage est secrète.
- Validy Technology n’est pas un système de sécurité centralisé : le système n'a pas de point faible unique. Si un jeton est compromis, seul le logiciel protégé par ce jeton n’est plus protégé.

Copyright © 2006 SA Validy Tous droits réservés