

Analyse automatisée d'une bibliothèque cryptographique



Détection de failles par canal auxiliaire par analyse statique et symbolique

Duzés Florian



Introduction - 1



Introduction - 1

HACL*

"High Assurance Cryptography Library"^a est une bibliothèque cryptographique, écrite en F* ("F star"), implémentant tous les algorithmes de cryptographie modernes et est prouvée mathématiquement sûre.

HACL* est notamment utilisé dans plusieurs systèmes de production tels que Mozilla Firefox, le noyau Linux, le VPN WireGuard...

a. <https://hacl-star.github.io/>

Introduction - 2

1996 : Paul C. Kocher, *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*

Une mesure précise du temps requis par des opérations sur les clés secrètes permettrait à un attaquant de casser le cryptosystème.



Introduction - 2

1996 : Paul C. Kocher, *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*

Une mesure précise du temps requis par des opérations sur les clés secrètes permettrait à un attaquant de casser le cryptosystème.

2003 : BRUMLEY et BONEH *Remote Timing Attacks Are Practical*



Introduction - 2

1996 : Paul C. Kocher, *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*

Une mesure précise du temps requis par des opérations sur les clés secrètes permettrait à un attaquant de casser le cryptosystème.

2003 : BRUMLEY et BONEH *Remote Timing Attacks Are Practical*

2011 : BRUMLEY et TUVERI *Remote Timing Attacks are Still Practical*



Introduction - 3

- QR1** Est-il possible de propager les garanties de sécurité pendant la compilation ?
- QR2** Est-il possible d'automatiser la détection de ces failles sur des fichiers compilés ?
- QR3** Est-il possible d'appliquer ces mécanismes pour assurer la vérification d'une bibliothèque cryptographique ?



Sommaire

1. Méthodes de protection et limitations

2. Outils de vérifications

3. Automatismes

4. Érysichthon

1. Conception générale
2. Andhrímnir

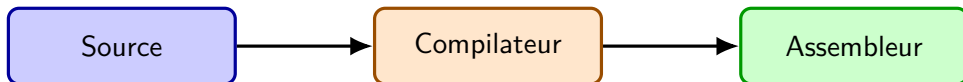
02

Méthodes de protection et limitations



État des lieux

Usage sécurisé

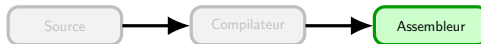


Analyse en remontée - 1

Écrire en assembleur

- + Efficace
- + Contrôle total

- Restreint l'architecture et les usages
- Beaucoup de connaissance spécifique au processeur ciblé



Analyse en remontée - 2

Utilisation des compilateurs

- Constantine - 2021
- Jasmine - 2017
- Raccoon - 2015
- CompCert - 2008 (2019)



Analyse en remontée - 2

Utilisation des compilateurs

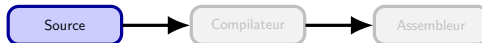
- Constantine - 2021
 - Jasmine - 2017
 - Raccoon - 2015
 - CompCert - 2008 (2019)
- Couverture des architectures supportée
 - Informations à transmettre
 - Spécifications ne sont plus respectées



Analyse en remontée - 3

Programmation en temps constant

- + Position haut niveau
- + Couverture d'architectures importantes
- Rigueur et conception particulière es actions
- Identification des points de fuites





Exemple



Bonnes pratiques de programmations

Écrire directement en Assembleur



Bonnes pratiques de programmations

Écrire directement en Assembleur

- Spécifique à un processeur
- Limite la portabilité du code

Bonnes pratiques de programmations

Programmation en temps constant

Écrire directement en Assembleur

- Spécifique à un processeur
- Limite la portabilité du code

Bonnes pratiques de programmations

Écrire directement en Assembleur


- Spécifique à un processeur
- Limite la portabilité du code

Programmation en temps constant

- Couverture des architectures supportée
- Informations à transmettre
- Spécifitées ne sont plus respectées

03

Outils de vérifications





04

Automatismes





05

Érysichthon

