Flavio

L'impact des phénomènes physiques dans la cybersécurité

J'envisage mon avenir professionnel dans les domaines de la cybersécurité ou de la robotique, que j'affectionne particulièrement. À travers ce TIPE, j'ai cherché un sujet en rapport avec mes perspectives d'avenir. Je trouve ce sujet intéressant parce qu'il aborde une thématique à laquelle nous ne pensons pas immédiatement en cybersécurité.

La ville du futur est une ville ultra-connectée, où chaque chose et chacun transmettent des informations aux autres. Ces différents échanges doivent être sécurisés mais pour cela il est impératif de prendre en compte les phénomènes physiques qui ne devraient pas changer avec le temps.

Positionnement thématique (ÉTAPE 1) :

- INFORMATIQUE (Informatique Théorique)
- INFORMATIQUE (Informatique pratique)
- MATHEMATIQUES (Autres)

Mots-clés (ÉTAPE 1) :

Mots-clés (en français)	**Mots-clés (en anglais)**
Algorithme en temps constant	*Constant time algorithm*
Temps d'éxécution	*Execution time*
Attaque temporelle	*Time attack*
Attaque par canal auxiliaire	*Side channel attack*
Code PIN	*Personal identification number*

Bibliographie commentée

Lorsque nous pensons à la cybersécurité, il nous vient en premier lieu à l'esprit les programmeurs qui écrivent différents algorithmes dans le but de protéger un système de potentielles intrusions externes et d'éviter les failles. Néanmoins, la cybersécurité ne se résume pas qu'à cela. En ce sens, afin de sécuriser un système, d'autres éléments que l'algorithmique entrent en jeu. C'est le cas par exemple de la physique. En effet, après que les programmes sont créés, ils sont implémentés dans des systèmes réels qui fonctionnent à l'aide de courants électriques qui émettent de la chaleur ; il est aussi possible de les perturber avec des champs électromagnétiques ; enfin, ils ont une certaine puissance de calcul qui ne permet pas de finir l' exécution de manière instantanée. À l'aide de tous ces facteurs, un attaquant bien préparé peut en déduire des informations.

Mon TIPE se focalise sur l'impact du temps dans la cybersécurité et notamment sur les attaques temporelles. Comment fonctionnent-elles ? De quelle manière pouvons-nous les

contrer ?

J'ai commencé par mener mes recherches sur le site *Interstices* [5] de l'institut national de recherche en sciences et technologies du numérique (INRIA) afin d'avoir une première introduction au sujet. Ensuite, ayant des difficultés à trouver des ressources, j'ai contacté un chercheur à l'INRIA, Adrien Koutsos, qui a bien voulu m'accueillir, me présenter les algorithmes dits « *constant time* », la solution qui est aujourd'hui utilisée pour se protéger des attaques temporelles, mais aussi me diriger vers divers articles de recherche afin d'approfondir le sujet. Grâce à ces précieuses informations et à l'aide des documents [3] et [4], j'ai pu me consacrer à l'analyse de plusieurs algorithmes codés d'abord naïvement puis en « *constant time* » et j'ai constaté divers problèmes pour les premiers : il est possible de déduire diverses informations en analysant le temps d'exécution des algorithmes. Néanmoins, les algorithmes « *constant time* » ne sont pas absolus, après qu'une personne a écrit un programme, c'est au tour du compilateur de travailler. Or, celui-ci fait des optimisations et comme le montre l'article de recherche [1], un algorithme qui est au départ « *constant time* » peut perdre cette propriété à la compilation, et la seule manière de vérifier est d'analyser le code assembleur généré par le compilateur. Ainsi ai-je repris un des programmes que j'avais déjà étudié, un algorithme de vérification de code PIN, et je me suis demandé s'il n'était pas possible de l'implémenter d'une façon différente pour que nous n'ayons pas à se soucier du compilateur. Pour cela, j'ai eu plusieurs approches : en premier lieu un algorithme de code PIN avec un ordre de vérification aléatoire et ensuite une implémentation directement physique à l'aide de portes logiques ou de transistors. Toutefois après diverses analyses, il s'avère que ce ne sont pas des moyens de protection suffisants. Dès lors, notre seule solution reste les algorithmes « *constant time* » puis de vérifier après compilation qu'ils gardent leurs propriétés.

Problématique retenue

Dans quelle mesure l'analyse du temps d'éxécution d'un algorithme permet-il d'en déduire des informations ? Et de quelle manière pouvons-nous nous en protéger ?

Objectifs du TIPE du candidat

Mon TIPE cherche à mettre en évidence que le temps d'exécution d'un algorithme peut nous permettre de déduire des informations sur celui-ci. Il a aussi pour objectif d'exposer une solution à ce problème à travers diverses démarches : les algorithmes dits "constant time", un ordre de vérification aléatoire pour un code PIN par exemple ou encore une implémentation physique via des portes logiques. Enfin il met en exergue les limites de ces différentes approches.

Références bibliographiques (ÉTAPE 1)

[1] GILLES BARTHE, BENJAMIN GRÉGOIRE , VINCENT LAPORTE : Secure Compilation of Side-Channel Countermeasures: The Case of Cryptographic "Constant-Time" *: CSF 2018 - 31st IEEE Computer Security Foundations Symposium, Jul 2018, Oxford, United Kingdom. hal-01959560*

[2] MANUEL BARBOSA, GILLES BARTHE, FRANÇOIS DUPRESSOIR, MICHAEL EMMI : Verifying Constant-Time Implementations *: Proceedings of the 25th USENIX Security Symposium August 10–12, 2016 , Austin, TX ISBN 978-1-931971-32-4*

[3] DANIEL J. BERNSTEIN : Cache-timing attacks on AES *: https://mimoza.marmara.edu.tr /~msakalli/cse466_09/cache%20timing-20050414.pdf, consulté en janvier 2023*

[4] TROMER, E., OSVIK, D.A, SHAMIR : A. Efficient Cache Attacks on AES, and Countermeasures *: J Cryptol 23, 37–71 (2010). https://doi.org/10.1007/s00145-009-9049-y*

[5] HÉLÈNE LE BOUDER : Des attaques informatiques utilisant la physique *: https://interstices.info/des-attaques-informatiques-utilisant-la-physique/, consulté en février 2022*

DOT

[1] *: Avril 2022 - Rencontre avec Adrien Koutsos, chercheur à l'INRIA au sujet des algorithmes constant time*

[2] *: Juin 2022 - Etude d'un code assembleur visant à montrer l'impact du compilateur : échec*

[3] *: Juillet 2022 - Début de tests d'algorithme et d'analyse de temps d'exécution*

[4] *: Septembre 2022 - Création de nouveaux algorithmes*

[5] *: Novembre 2022 - Etude d'un nouveau code assembleur pour voir l'impact du compilateur*

[6] *: Janvier 2023 - Création de la partie de code pin avec ordre de vérification aléatoire*

[7] *: Mars 2023 - Etude de circuits logique : implémentation du code pin*

[8] *: Mai 2023 - Préparation de la présentation*