Inria

Timesecbench

Évaluation des fuites d'information par dépendances temporelles dans la microarchitecture

Mathieu Escouteloup, Ronan Lashermes

Inria

*thèse dirigée par Jacques Fournier, CEA-Leti

Contents

Inria

01

Motivation


```
tab[ (*s_addr) * 4096];
```

- $s \leftarrow$ load [s_addr]: lecture du secret, normalement interdite.
- load [$tab + 4096 \cdot s$]: touche une ligne de cache dépendante de s.
- l'attaquant n'a plus qu'à trouver les addresses des données en cache (*cache timing attacks*).

Inria

```
if (x < tab1_size) {
   y = tab2[tab1[x]* 4096];
}
```

- $s \leftarrow$ load $[x + tab1]$: lecture du secret, normalement interdite.
- load $[tab2 + 4096 \cdot s]$: touche une ligne de cache dépendante de s.
- l'attaquant n'a plus qu'à trouver les addresses des données en cache (*cache timing attacks*).

Inria

Résumé

Attaques en 2 phases $\rightarrow$ 2 classes de vulnérabilités

- Accéder aux données sensibles: contrôle d'accès.
- Exfiltrer ces données: fuite d'information par dépendance temporelle.

Attaques en 2 phases $\rightarrow$ 2 classes de vulnérabilités

- Accéder aux données sensibles: contrôle d'accès.
- **Exfiltrer ces données:** fuite d'information par dépendance temporelle.

Domaines de sécurité

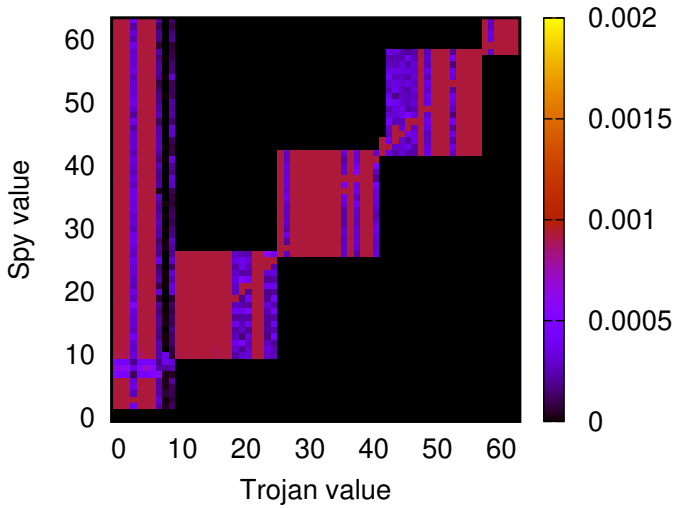
Plus précisément, dans les exemples précédents, les deux instructions **load** sont réalisées dans des domaines de sécurité différents.

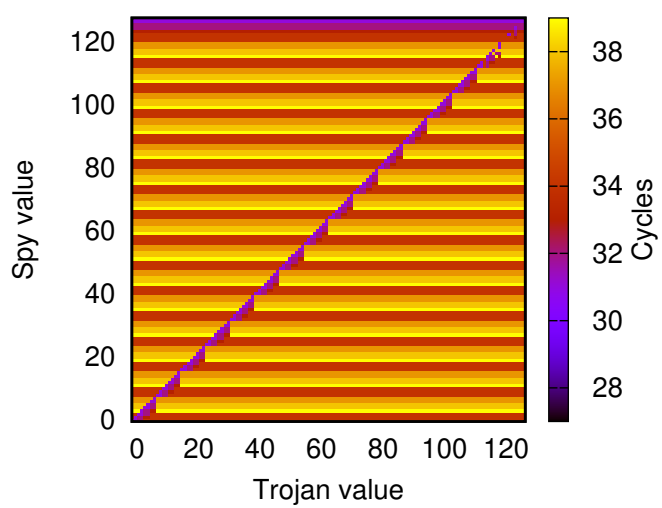
02

Objectifs

Présentation

Timesecbench est une suite de benchmarks **en cours de développement** pour évaluer les fuites d'information microarchitecturales par dépendance temporelle.



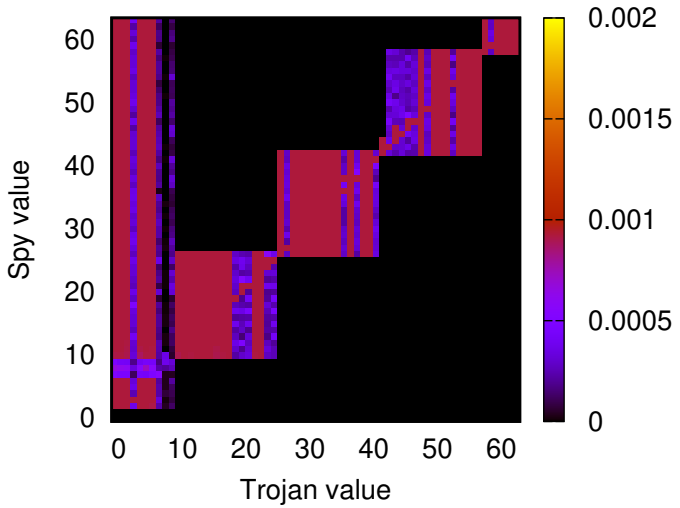


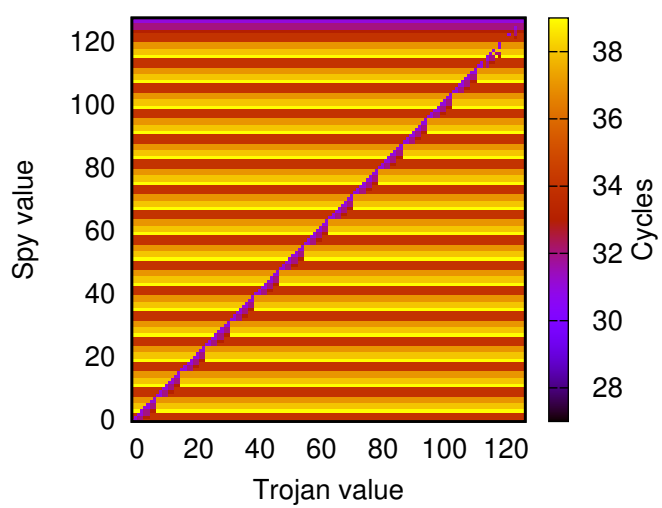
Présentation

Timesecbench est une suite de benchmarks **en cours de développement** pour évaluer les fuites d'information microarchitecturales par dépendance temporelle.

Pour guider la conception matérielle

Ne permets pas de garantir une exécution en temps constant : c'est le rôle d'autres outils (BESSPIN, . . .).





Scénario 1

Vous êtes un concepteur de processeurs et vous voulez vérifier qu'il n'y a pas de fuites d'information par dépendances temporelles dans votre implémentation.

Scénario 1

Vous êtes un concepteur de processeurs et vous voulez vérifier qu'il n'y a pas de fuites d'information par dépendances temporelles dans votre implémentation.

Scénario 2

Vous êtes un auditeur de sécurité et vous voulez vérifier la robustesse d'une implémentation matérielle.

Cas d'usage

Scénario 1

Vous êtes un concepteur de processeurs et vous voulez vérifier qu'il n'y a pas de fuites d'information par dépendances temporelles dans votre implémentation.

Scénario 2

Vous êtes un auditeur de sécurité et vous voulez vérifier la robustesse d'une implémentation matérielle.

Limitations

L'outil démontre la présence d'une vulnérabilité, pas l'absence de vulnérabilités.

Exploitabilité

Les concepteurs se concentrent actuellement sur les failles dont l'exploitabilité a été démontrée.

Vulnérabilité

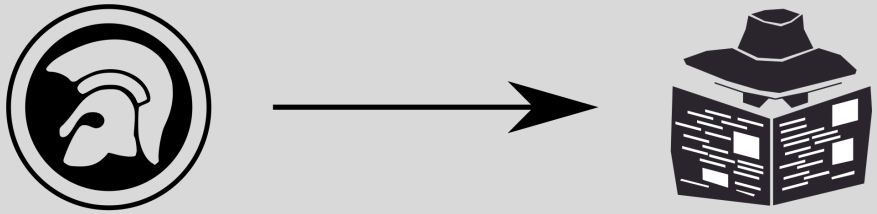
Notre suite de tests cherche à évaluer la présence de vulnérabilités, qu'elles soient exploitables ou non.

Nous nous positionnons comme défenseur : l'absence de vulnérabilité implique l'absence de faille exploitable.

03

Principes de fonctionnement

Un canal de communication



Modèle de menace pour les fuites microarchitecturales

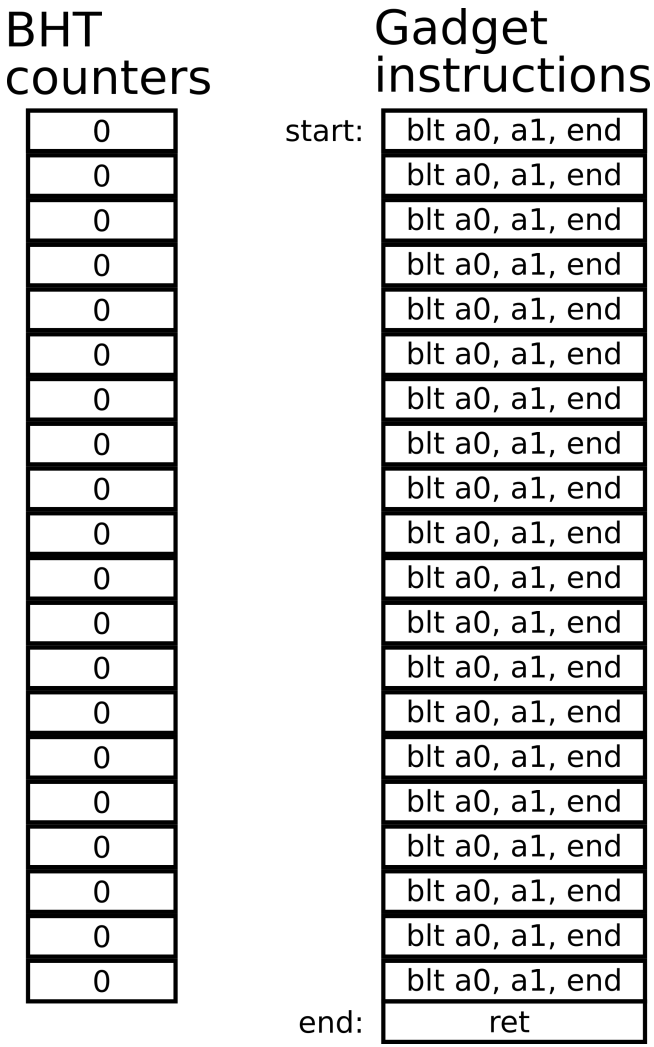
L'attaquant contrôle à la fois le Troyen et l'espion, mais ne peut encoder l'information que dans les états microarchitecturaux.

```
trojan(i);
architecture_clear(); //registers, memory, CSRs
domain_switch_to(spy_domain);
spy(o);
```

$\exists$ canal auxiliaire $\implies$ $\exists$ canal caché.

Inria

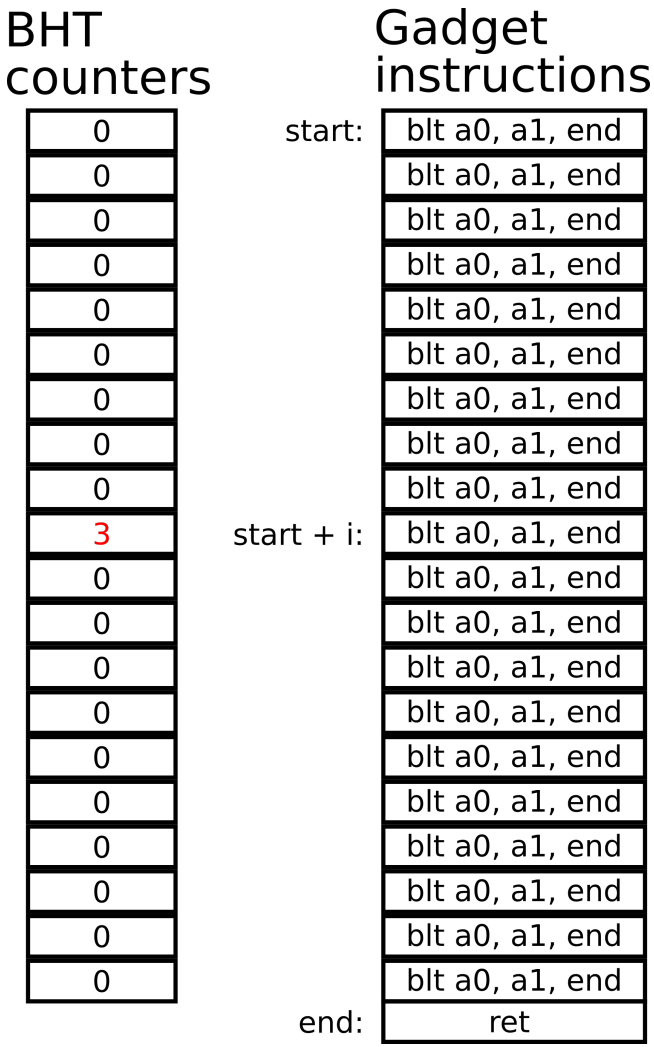
Exemple: Branch History Table (BHT)



Effacement des compteurs

Répéter `(*start)(2,1);`

Exemple: Branch History Table (BHT)



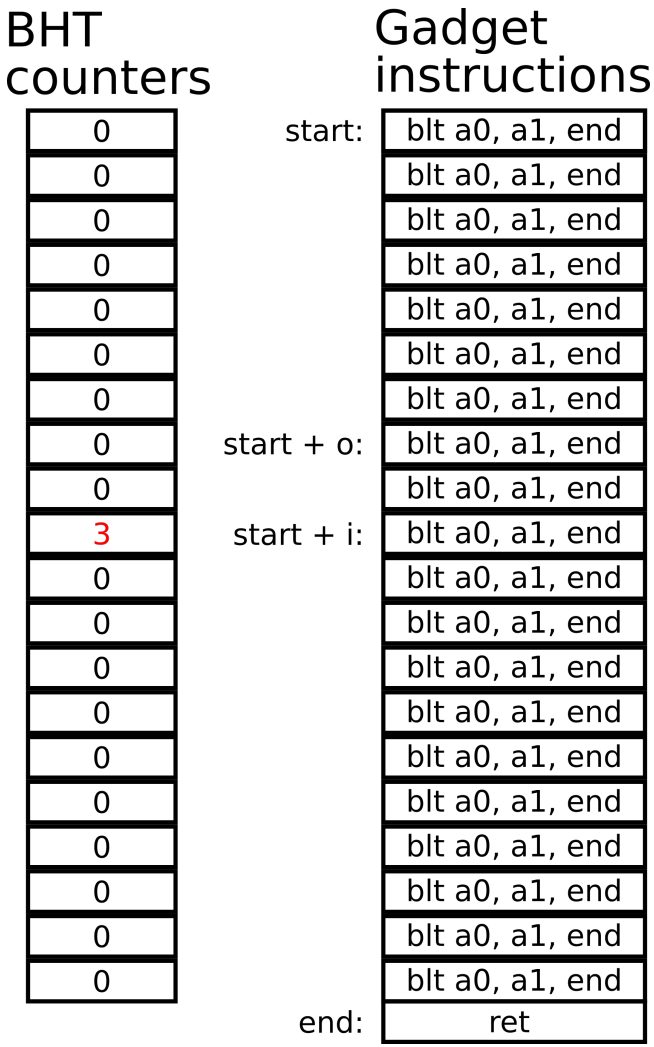
Effacement des compteurs

Répéter `(*start)(2,1);`

Encoder *i*, la valeur du troyen

Répéter `(*(start + i))(1,2);`

Exemple: Branch History Table (BHT)



Effacement des compteurs

Répéter `(*start)(2,1);`

Encoder *i*, la valeur du troyen

Répéter `(*(start + i))(1,2);`

Tester *o*, la valeur de l'espion

Exécuter:

```
 u32 a = read_time();
(*(start + o))(1,2);
return read_time() - a;
```


Inria

BHT matrice des temps

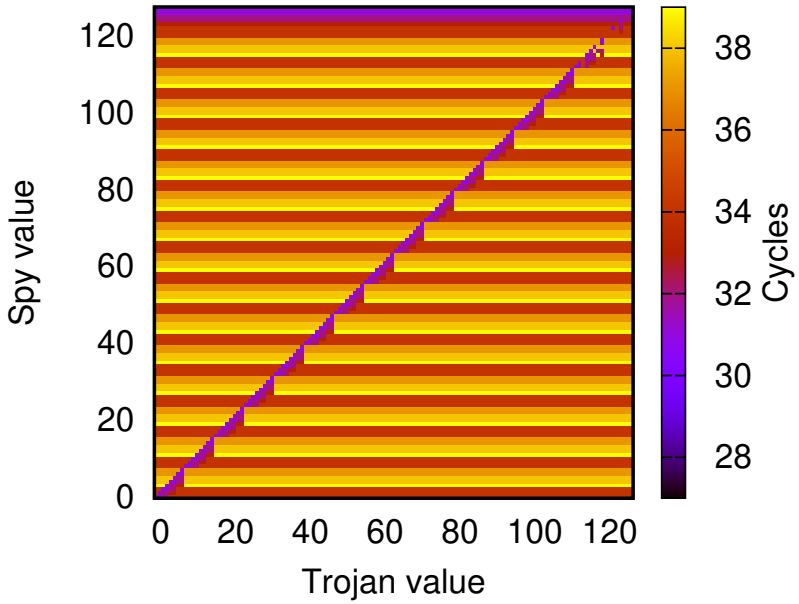


Figure: Implémentation vulnérable

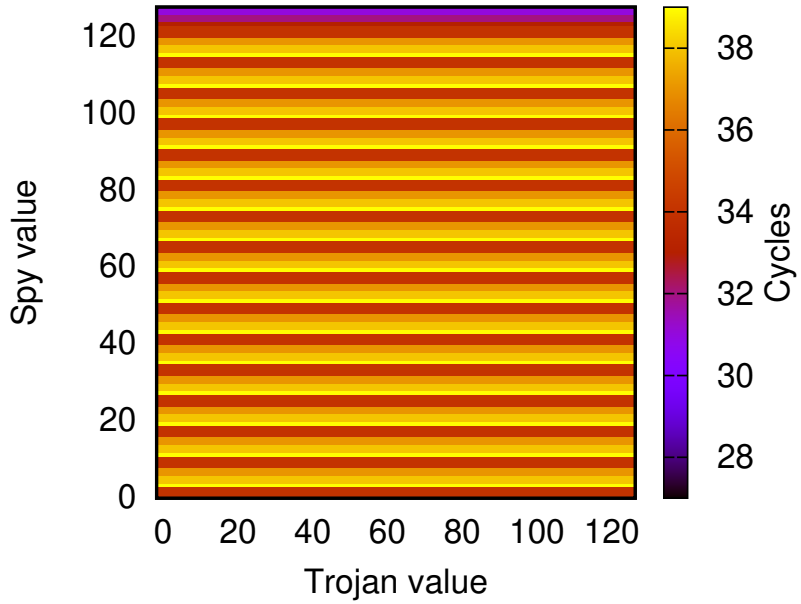


Figure: Implémentation sure

Visualiser la fuite

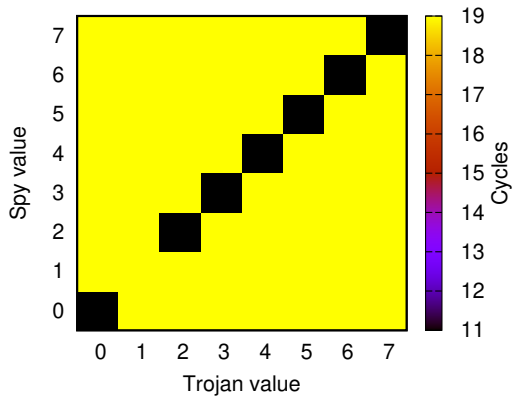
Il y a fuite d'information s'il y a de la variation horizontale: on peut détecter si `i == o`.

Entropie mutuelle

Quantité d'information (bits par symbole) transmise pour une distribution uniforme en entrée.

Capacité du canal de communication

Quantité d'information (bits par symbole) transmise pour une distribution en entrée qui maximise cette quantité.



Matrice des temps pour L1I sur Aubrac.

- Information mutuelle: **1,37 bits par symbole**.
- Capacité du canal: **2,62 bits par symbole**.
- Maximum théorique: **3 bits par symbole** ($2^3 = 8$).

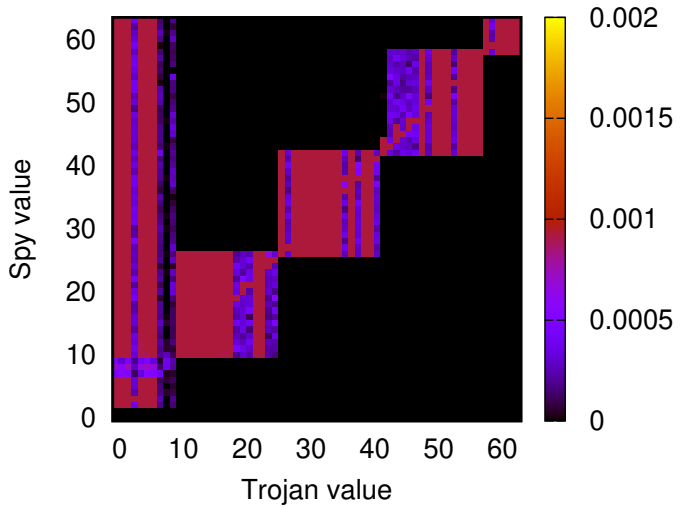
04

Démos

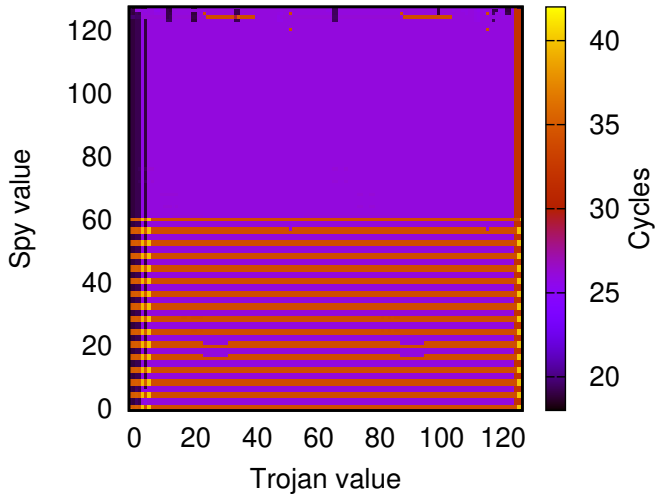
Raspberry Pi 3B+

ARMv8-A, quad-core, 8-stage, in-order, Cortex-A53, pas de contremesures.

Résultats RPi3



L1I (intensity map)

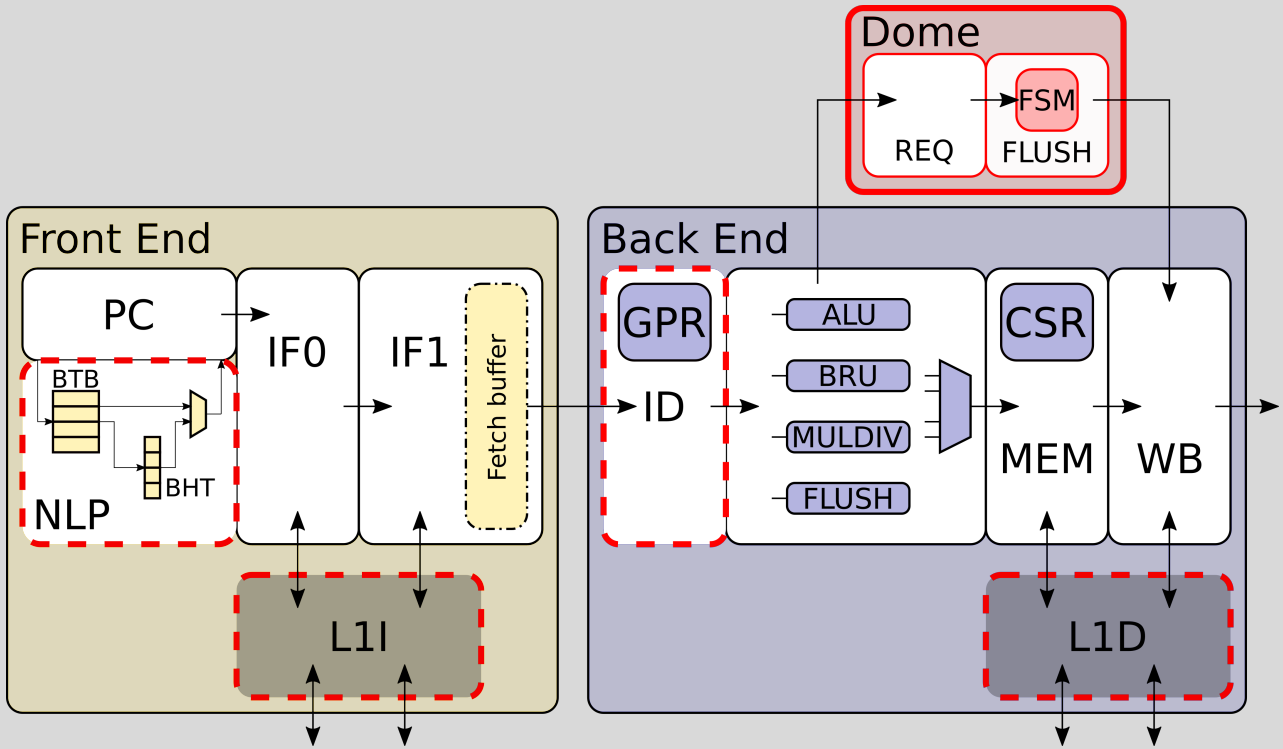


BHT

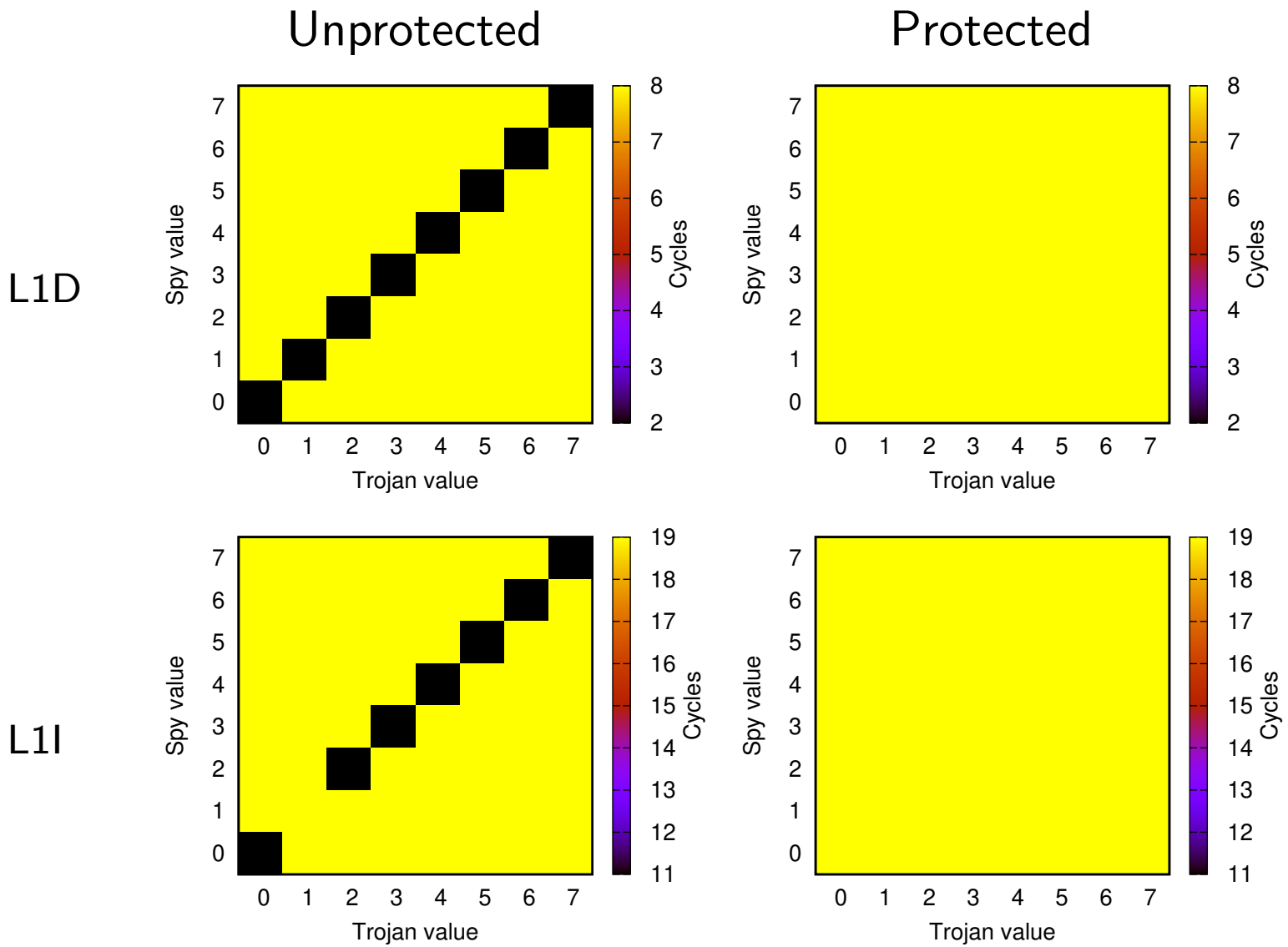
Aubrac

RISC-V, single-thread, 5-stage, in-order, contremesures optionnelles.

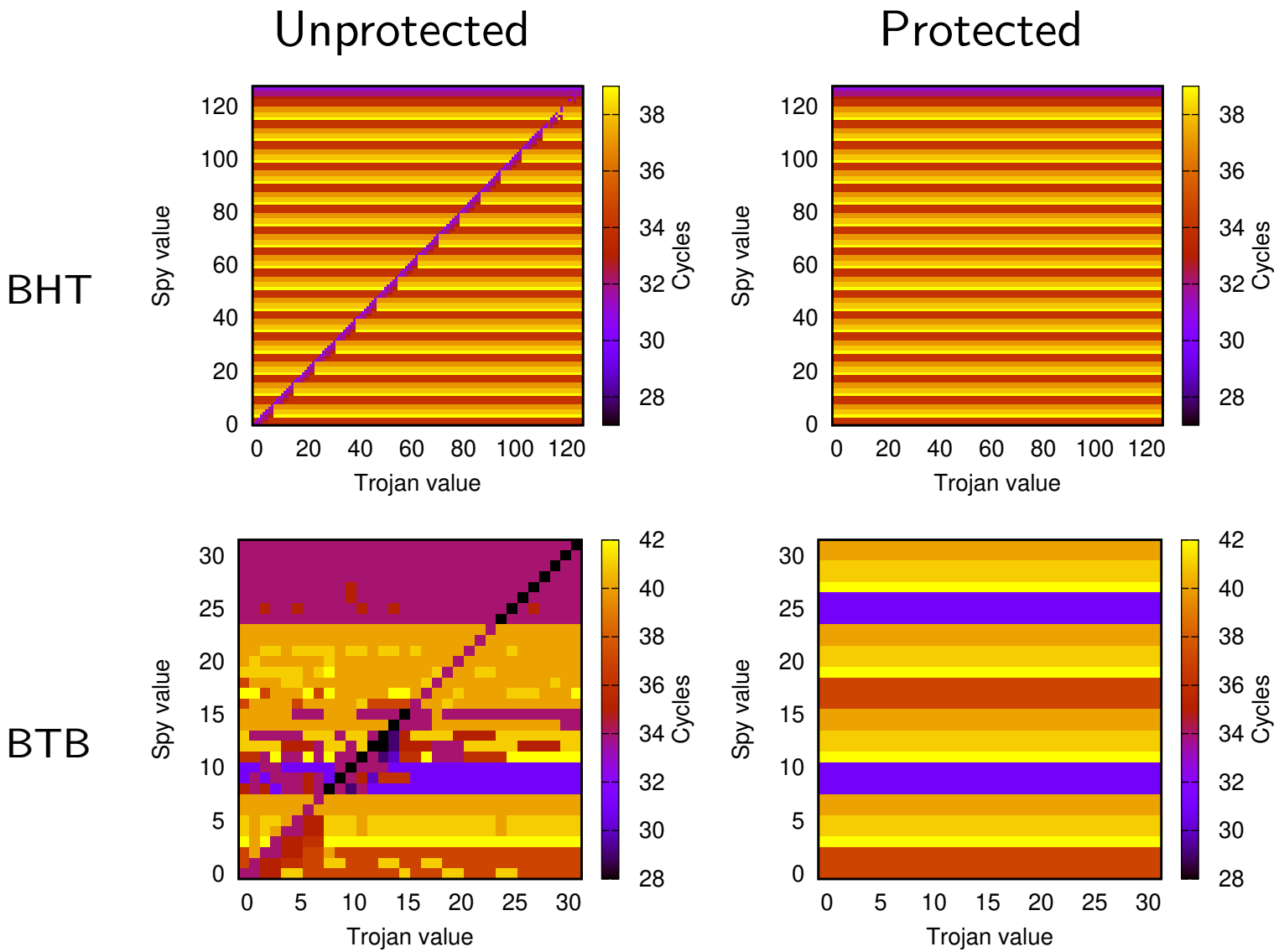
Conçu par **Mathieu Escouteloup, Inria**.



Aubrac results

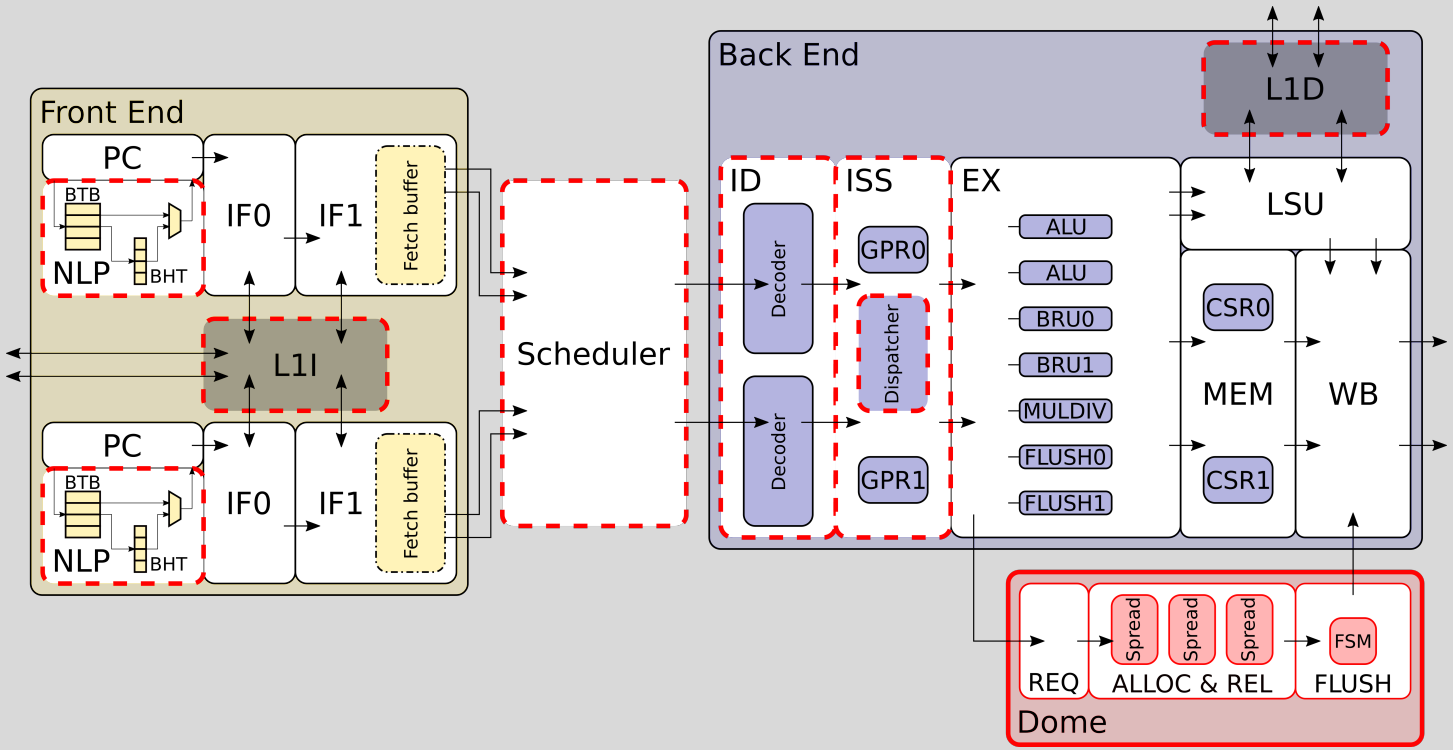


Résultats Aubrac



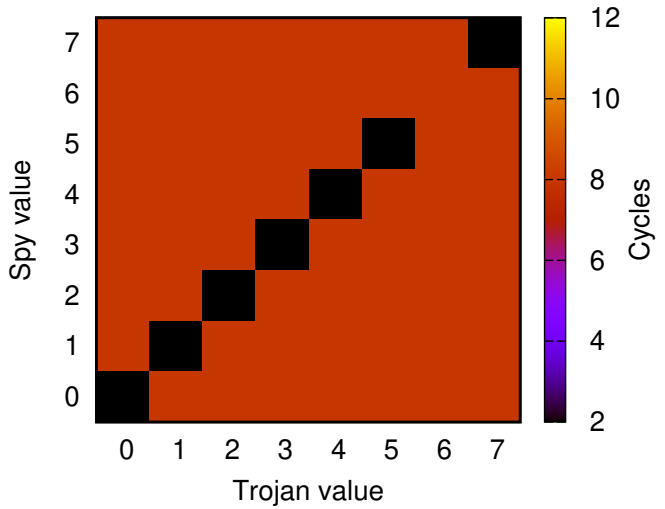
Salers

RISC-V, dual-thread, 6-stage, in-order, contremesures optionnelles.
Conçu par **Mathieu Escouteloup, Inria**.

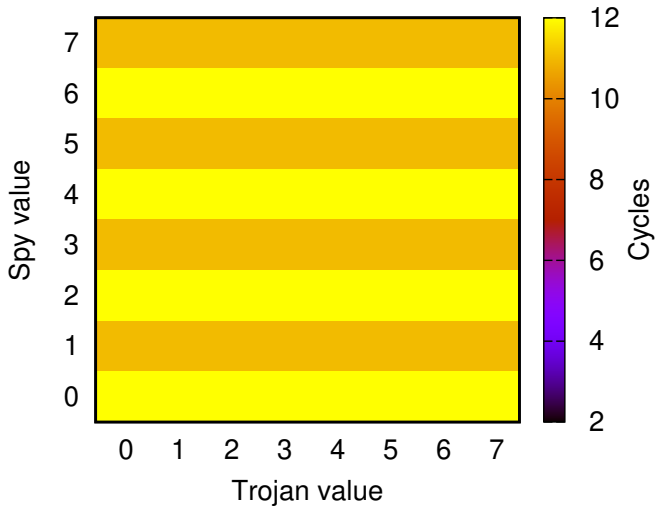


Cross L1D

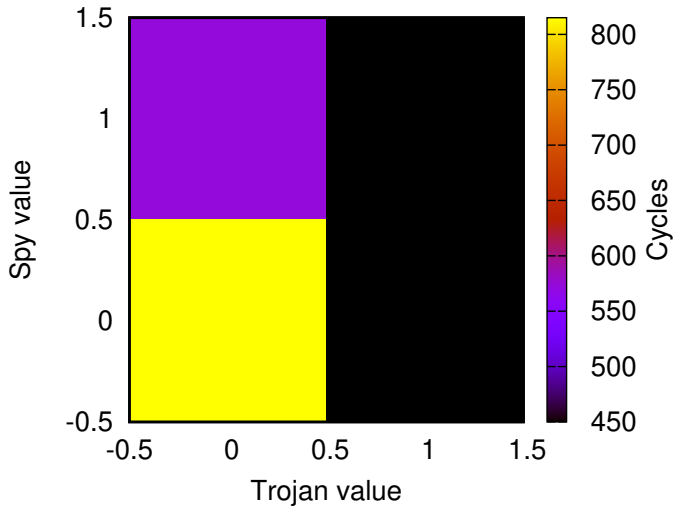
Unprotected



Protected



Port contention



05

Conclusion

Un outil pour la conception matérielle

- Permet de guider la conception pour prévenir les fuites d'information.
- Nécessite d'être étendu: quels sont vos cas d'usage ?
- License libre (MIT).

Un outil pour la conception matérielle

- Permet de guider la conception pour prévenir les fuites d'information.
- Nécessite d'être étendu: quels sont vos cas d'usage ?
- License libre (MIT).

Résistance aux attaques par dépendance temporelle

- Possible !
- Une modification du jeux d'instructions est nécessaire.
- Création d'une extension RISC-V en cours avec la fondation RISC-V.

Inria

- Publication: Under the dome: preventing hardware timing information leakage, Matthieu Escouteloup, Ronan Lashermes, Jacques Fournier à *CARDIS 2021*.
- Git pour Timesecbench: `https://gitlab.inria.fr/rlasherm/timesecbench`

Si le sujet vous intéresse, venez en discuter !

Des questions ?