Sécurité et ingénierie

(dans Chromium) - Julien Tinnès

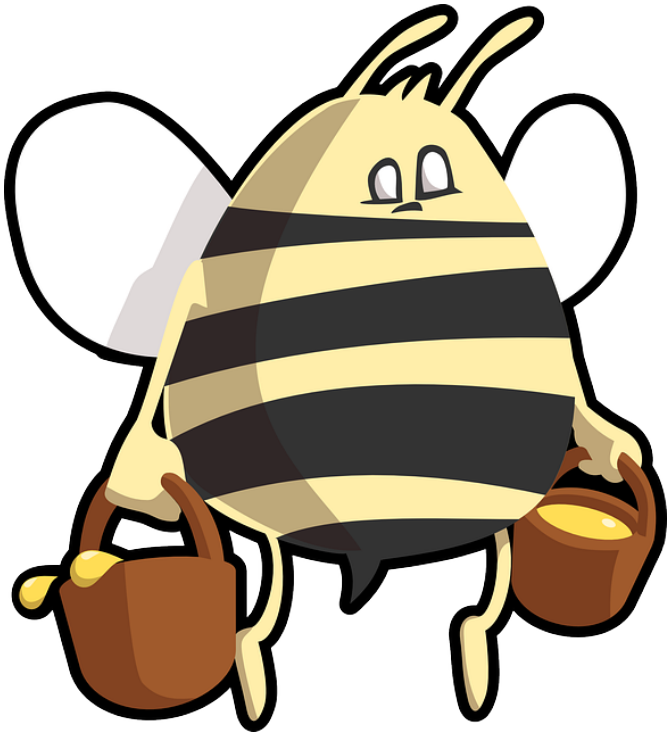
about:me

- Travaille à Google
  - Les opinions exprimées ici sont les miennes
- Vit à San Francisco, CA
- Membre de l’équipe de sécurité de Chromium / Chrome
  - Security Architecture (GUTS) TL

2005-2015

- Mon premier SSTIC en 2005

2005



Credit

2015

http://www.arcyber.army.mil/

2005

- On se remettait de MS Blaster (2003) et Sasser (2004)
- Le début de prise de conscience pour le grand public
- Pour un poste de travail, les failles “coté client” prennent le dessus

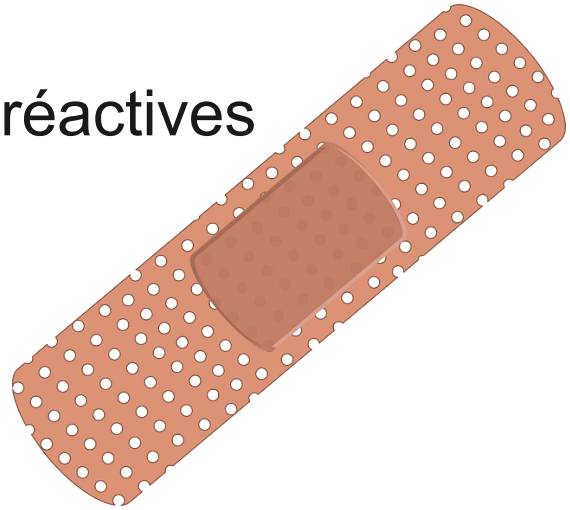
2015

- Les vulnérabilités sont en première page
- … avec des noms amusants et des logos
  - Heartbleed, Shellshock, BEAST, POODLE, LogJam

LogJam (Credit: @0xabad1dea)

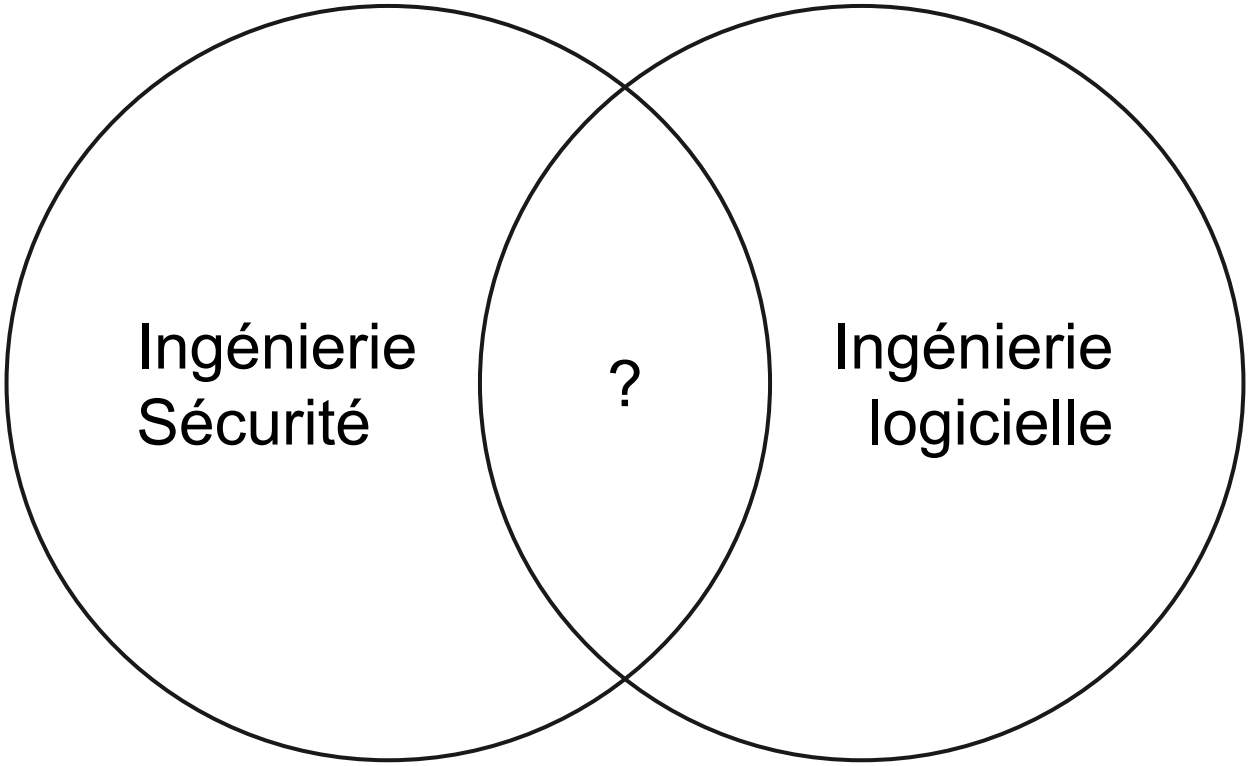
Ingénierie de sécurité

- 2005: encore très rare
  - Equipes de sécurité principalement réactives



- 2015: existe, en tout cas chez les gros éditeurs logiciels

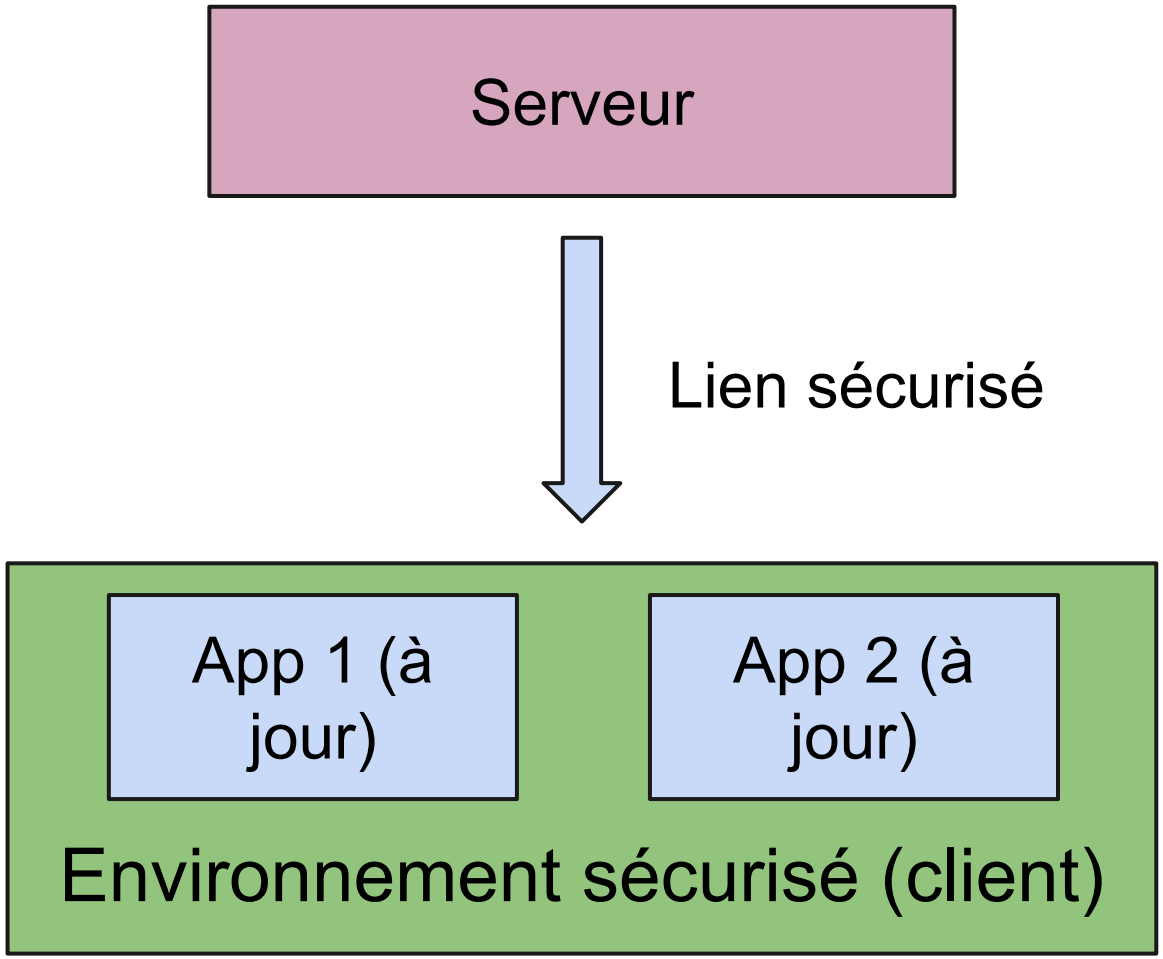
Sécurité et ingénierie



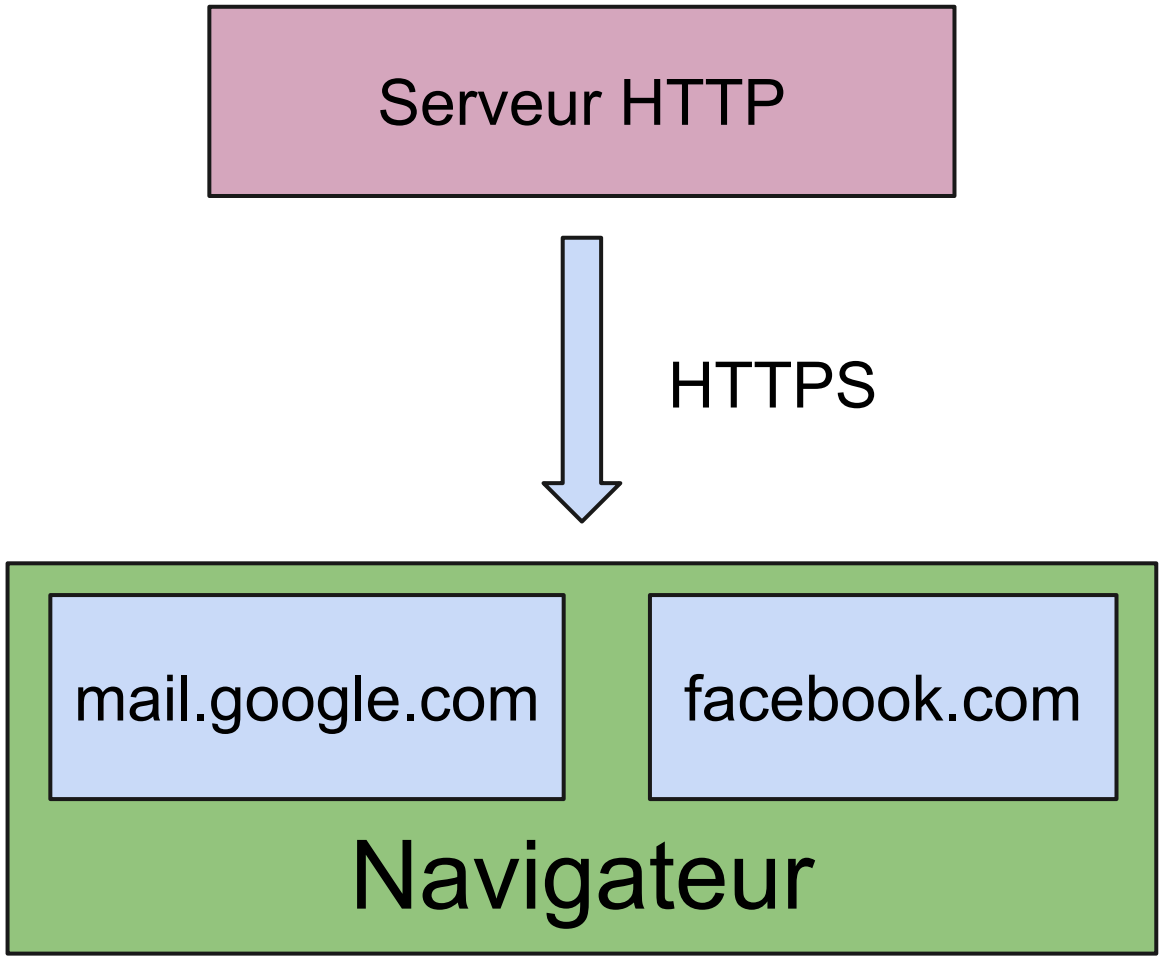
The dream of the 90s

Credit: “Portlandia” (https://www.youtube.com/watch?v=FE_9CzLCbkY)

The dream of the 90s

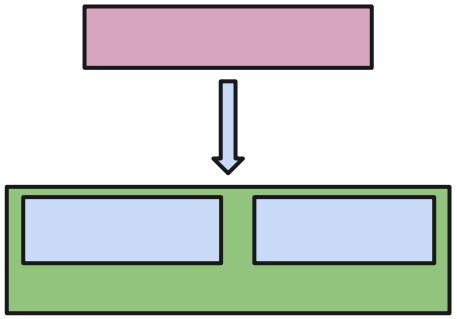


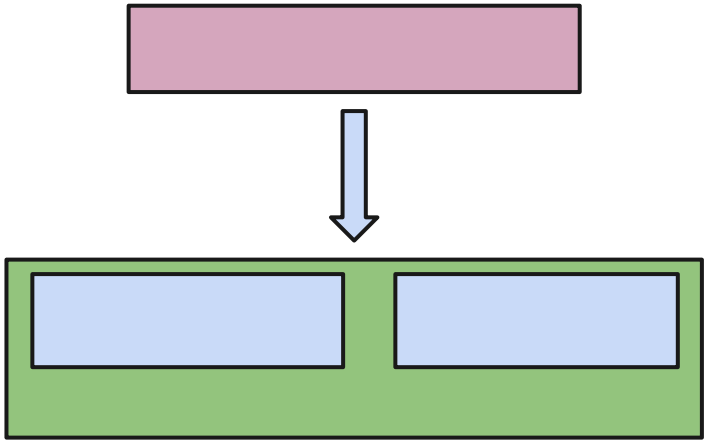
Applications web

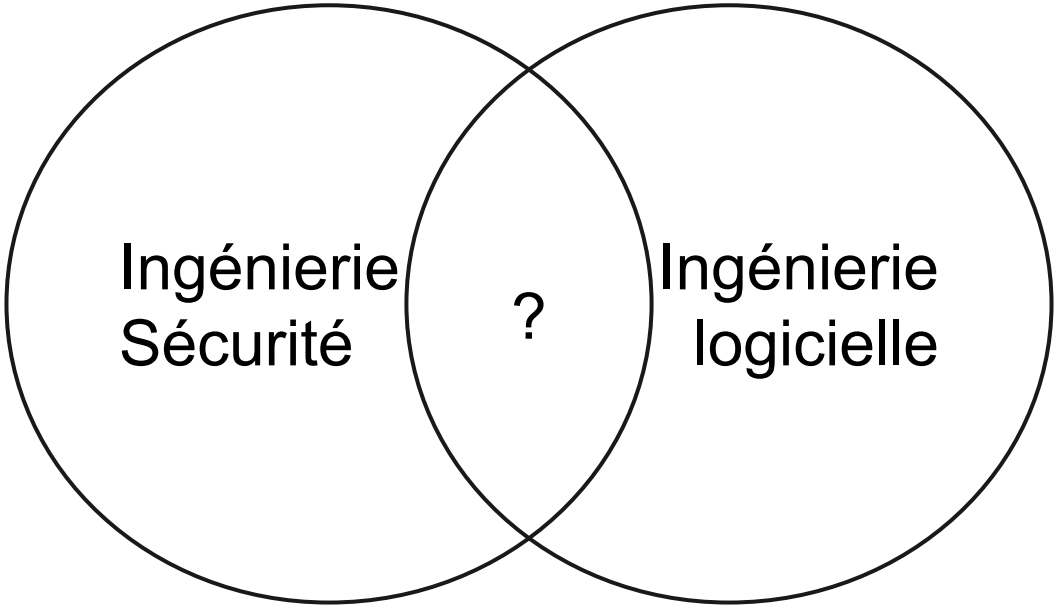


Web apps et sécurité

- Environnement d'exécution isolé du système
  - HTML5 + Javascript
- Chaque origine est isolée des autres
  - (Avec un modèle relativement chaotique)
- La sécurité est un point clé pour que le web fonctionne






Ingénierie
Sécurité
?
Ingénierie
logicielle

Google +  ≈ 

(+ Flash)

Chromium(OS)

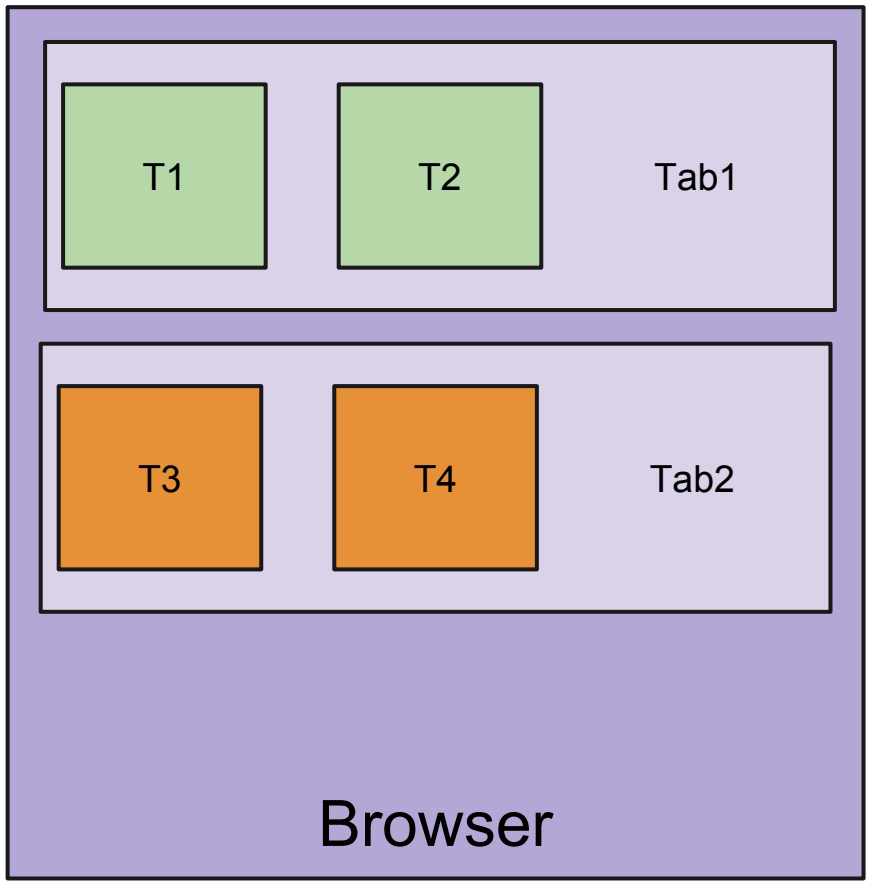
Chrome(OS)

- Desktop: Windows, Linux, OS X
- Chromium(OS): projet open-source
- Chrome(OS): Google pousse les mises à jour

Chromium

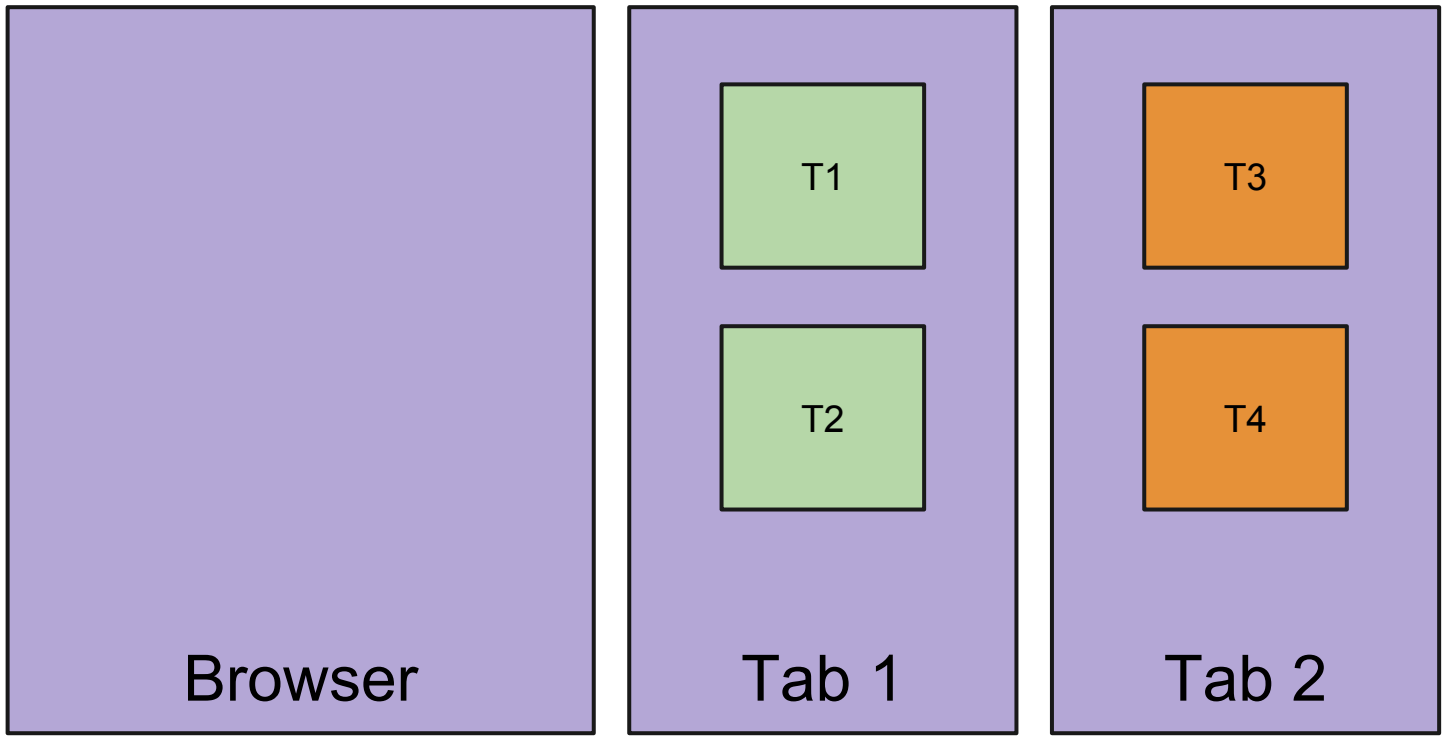
- Mission: Speed, Simplicity, Security
- Ça a de l'importance!

Fault isolation

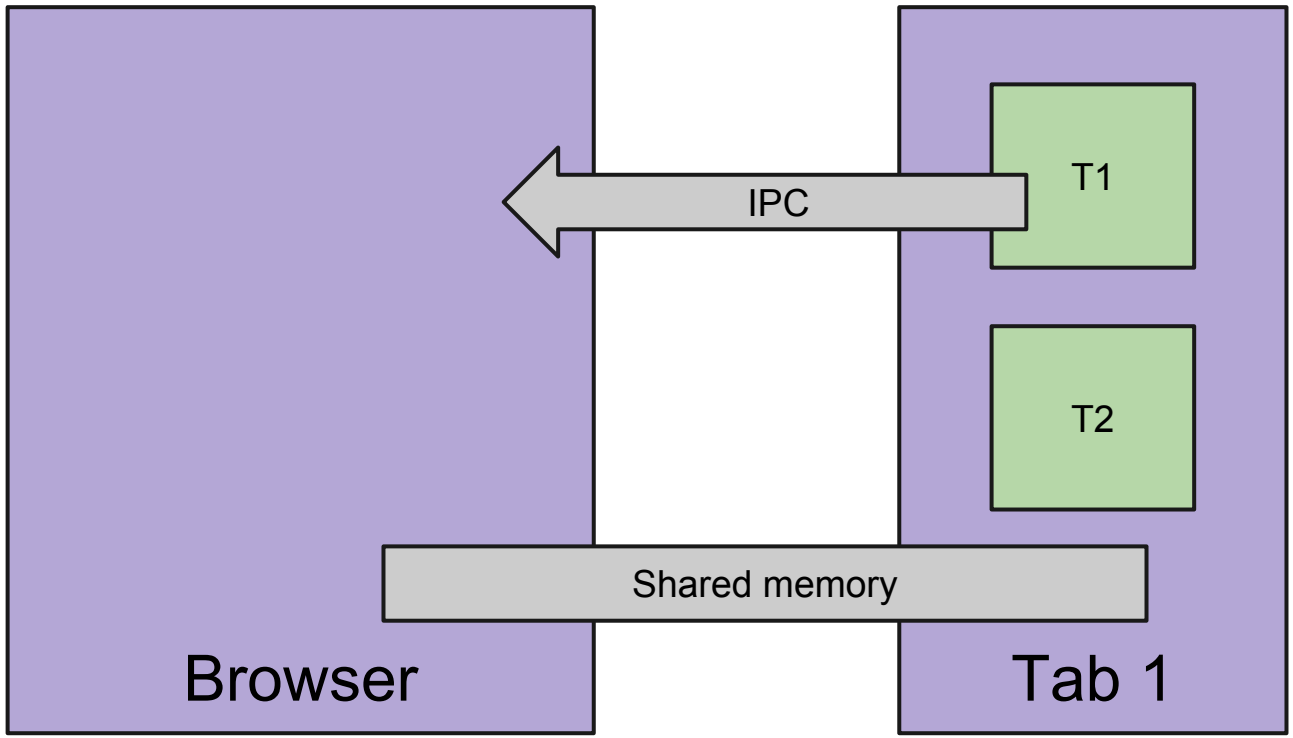


- T1 crashe
- Crash du navigateur, perte de tous les tabs

Fault isolation



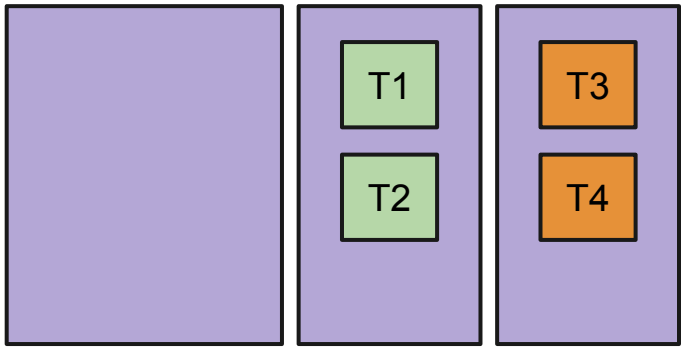
Fault isolation (par MMU)



Inter **P**rocess
Communication

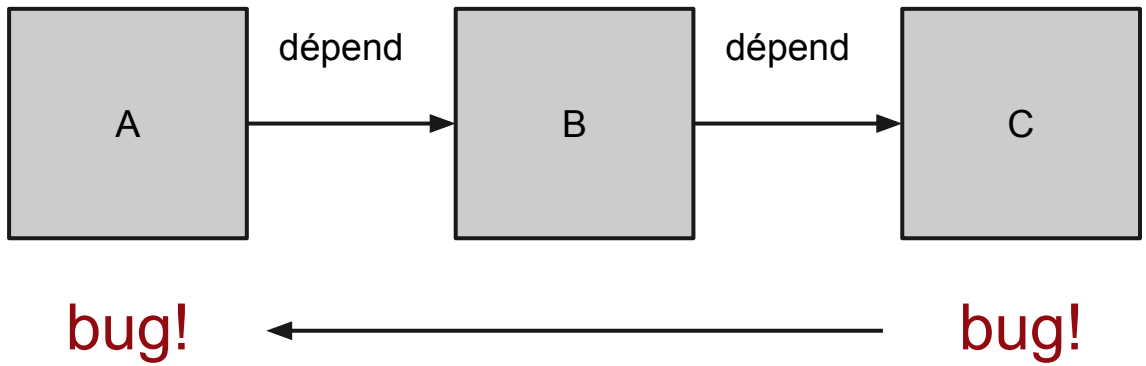
Dépendances

- Processus différents et (relativement) indépendants
  - Dépendances explicites via IPC
  - Toutes les dépendences du “renderer” vis à vis du “browser” sont explicites.

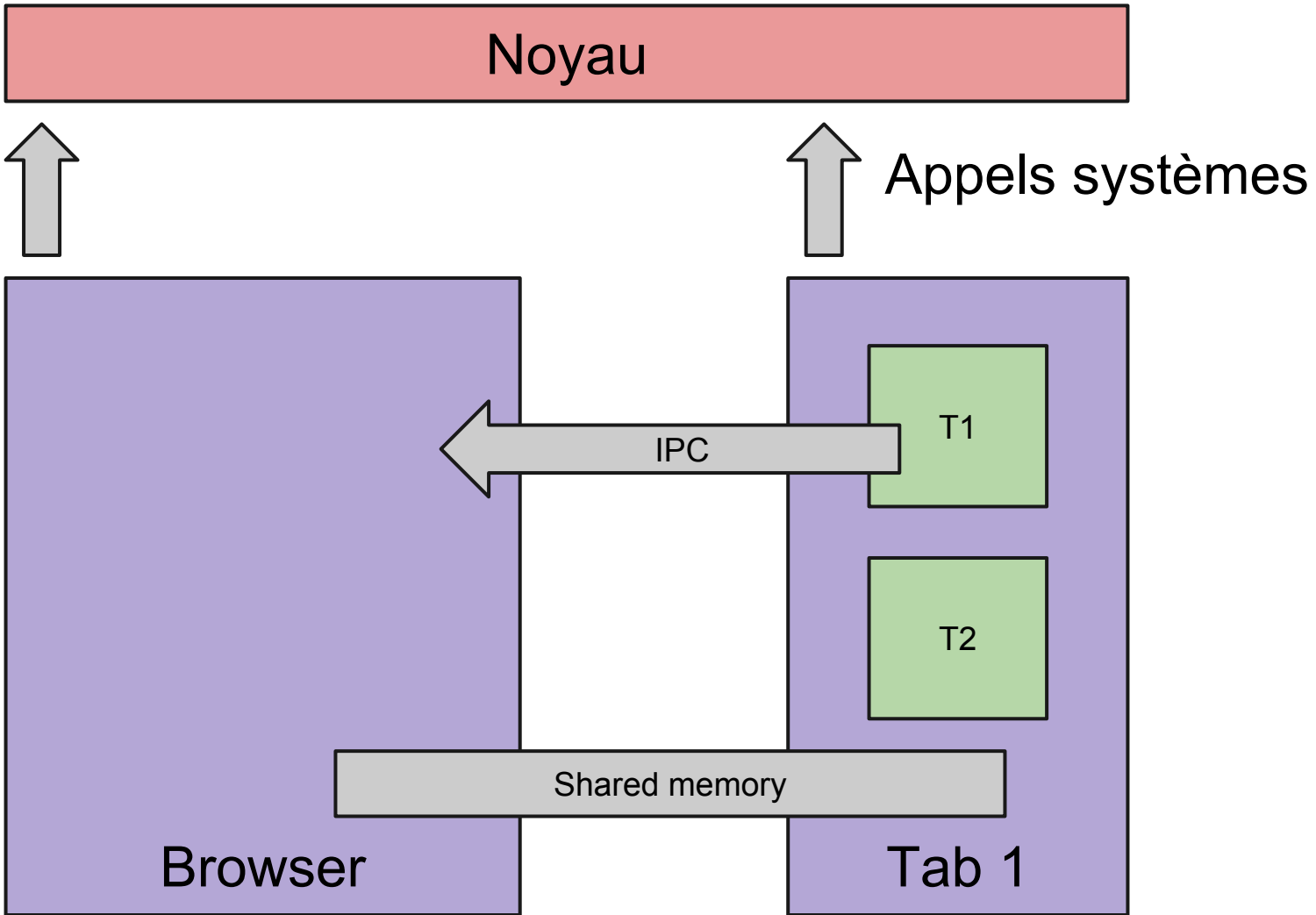


Dépendances (ingénierie)

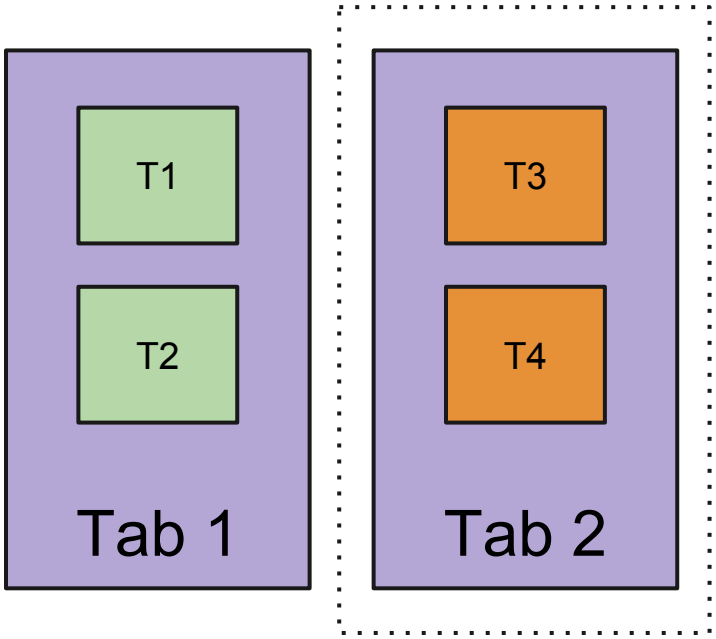
- Les dépendances introduisent des bugs par transitivité
- Dépendances systèmes
  - Code non portable
  - A la merci des bugs du système



Dépendances systèmes



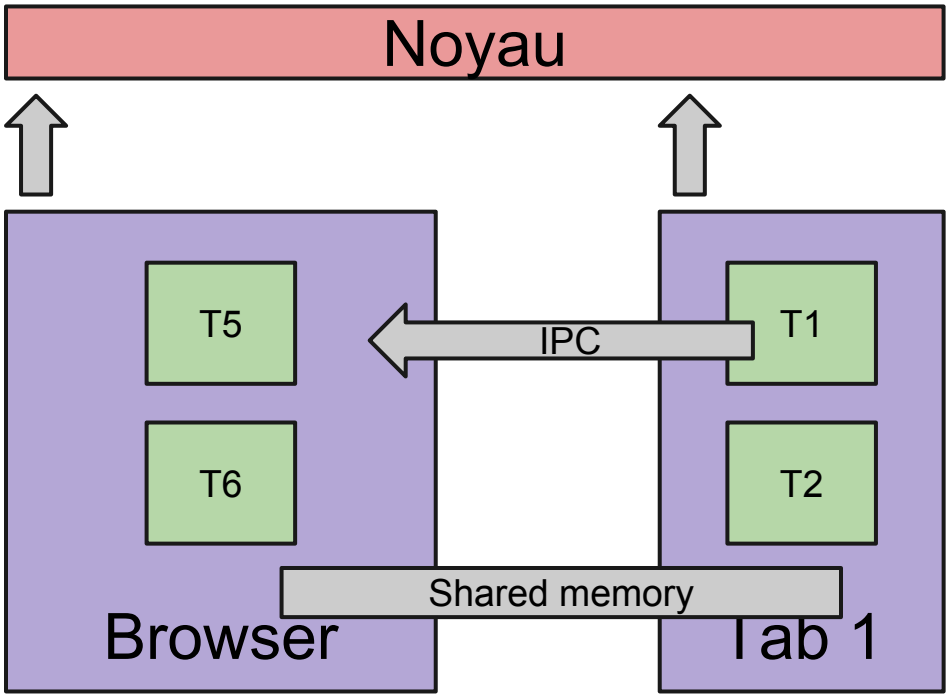
Ingénierie ➜ sécurité



~~Fault isolation~~

Séparation des privilèges!

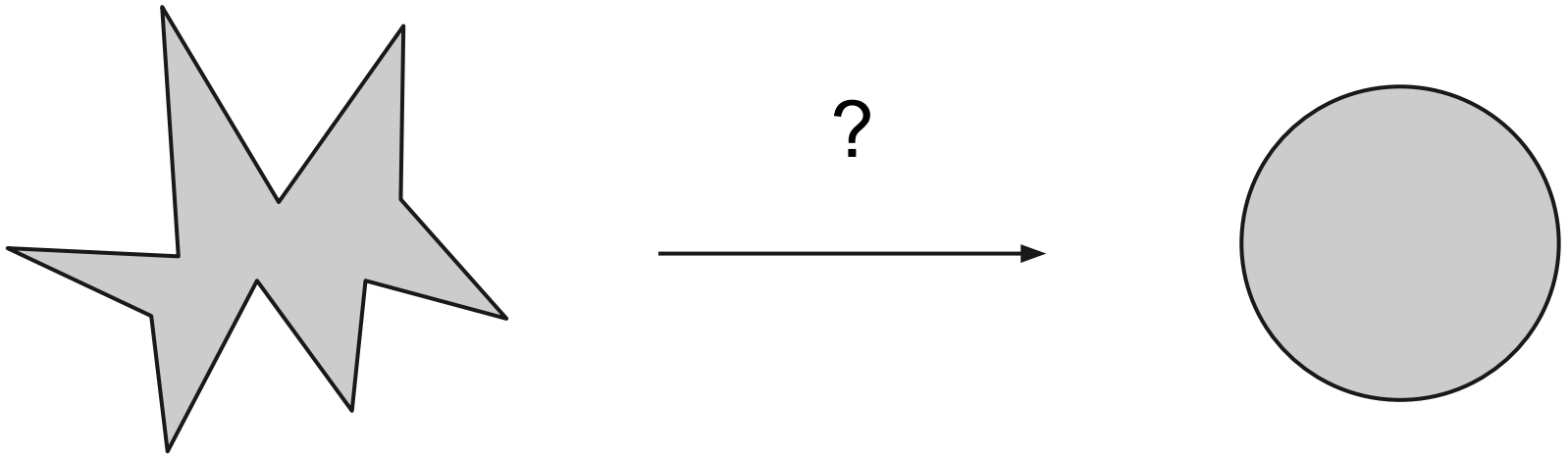
Ingénierie ➜ sécurité



Réduction

- ~~des dépendances~~
- du périmètre d'attaque!

Périmètre d’attaque

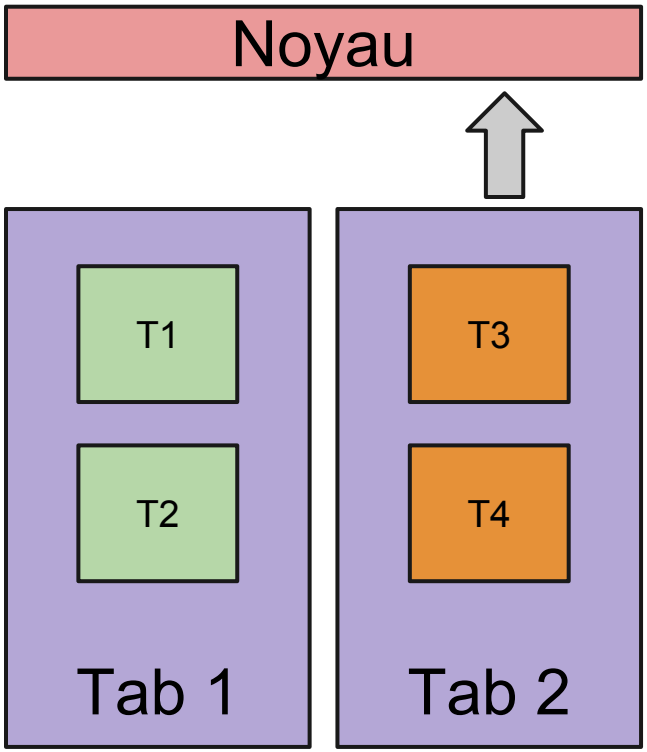


Même “contenu” (surface), différents périmètres

Ingénierie <-> Sécurité

- Simplification utopique, mais importante
- Un design où stabilité, sécurité et autres considérations profitent
  - Un élément considérable de la faisabilité

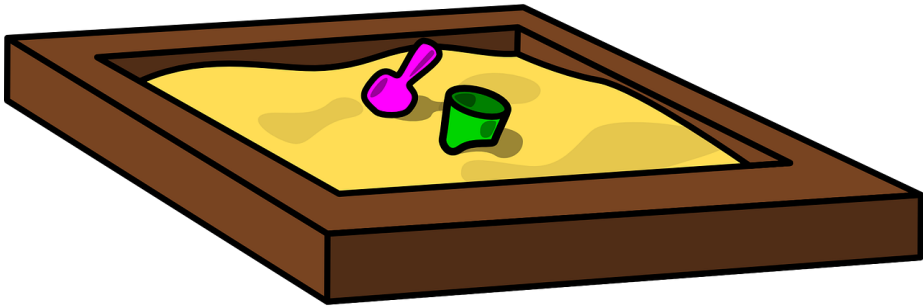
Séparation des privilèges



- La séparation des privilèges exige des restrictions au niveau noyau

Sandboxing

- Mécanisme discrétionnaire permettant de réduire l’ accès d’un processus aux aux resources du système.

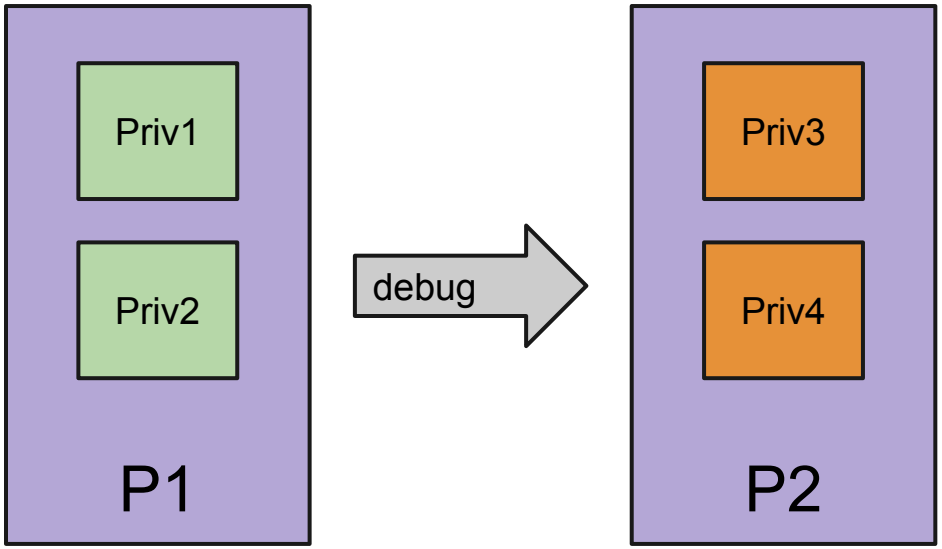


Sandboxing de Chromium

- Example 2009: Comment “sandboxer” Chrome Linux?

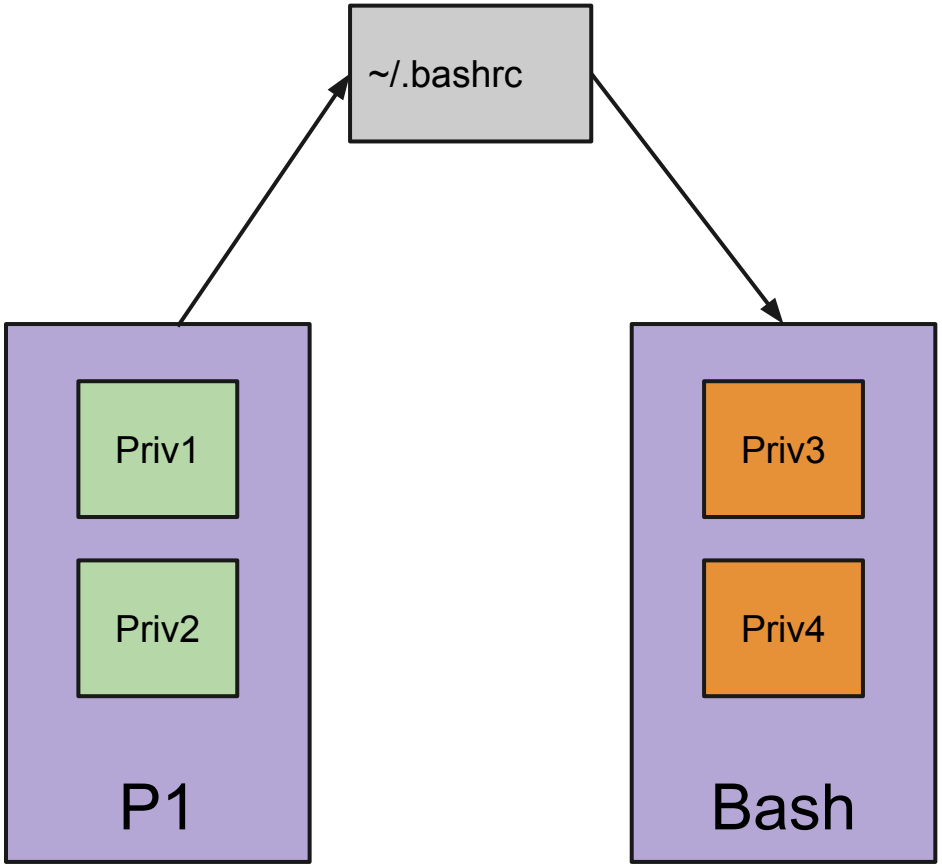
- Règle n°1: le niveau de privilège d’un processus sandboxé est inclus dans l’union des privilèges des processus de sa cloture transitive par la relation “A peut contrôler B”.

Sandboxing de Chromium



- P1 peut débogguer P2.
- Il a donc accès aux privilèges Priv3 et Priv4.

Sandboxing de Chromium



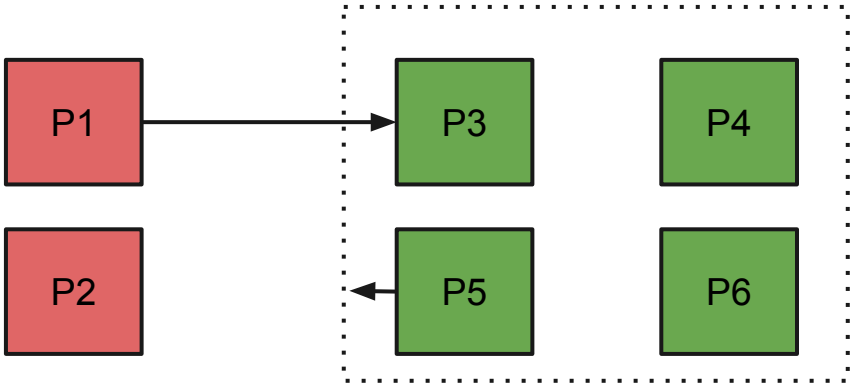
Plus subtil: P1 peut contrôler P2

Sandboxing de Chromium

- Il faut au minimum garantir
  - l’intégrité des processus non sandboxés
  - l’intégrité du système de fichier

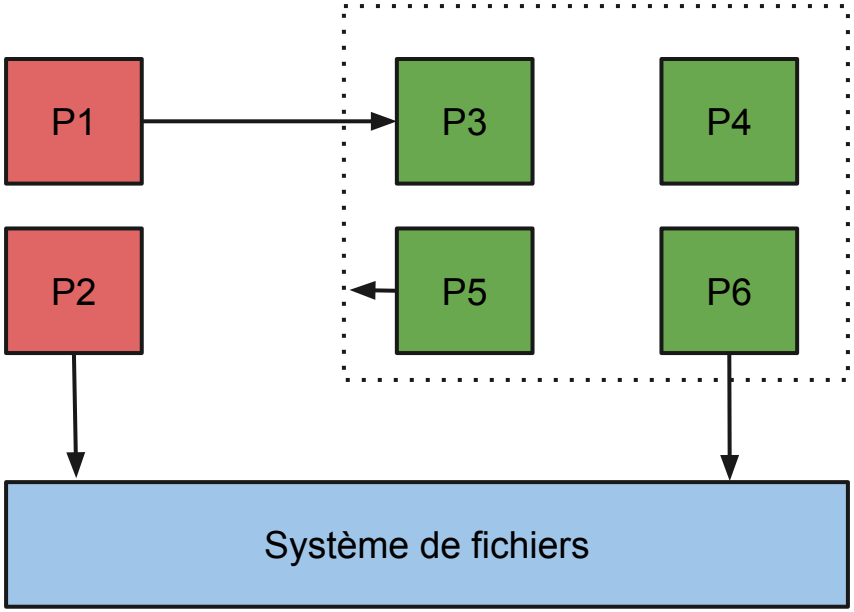
setuid sandbox

- Binaire setuid
- Qui crée un nouveau “PID namespace”



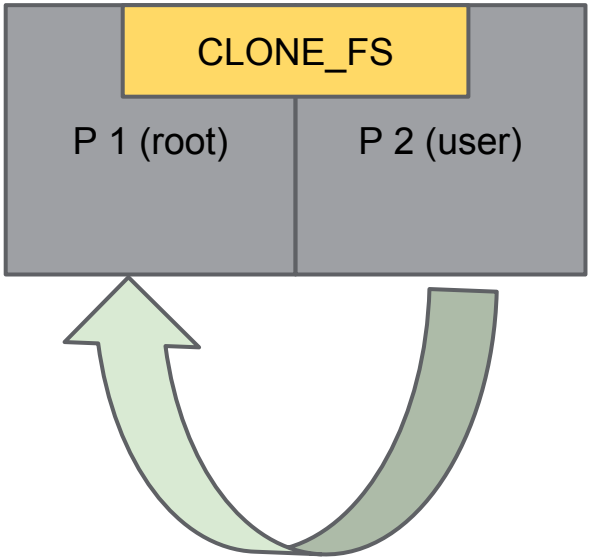
(CLONE_NEWPID, ajouté à Linux 2.6.24 (expérimental))

setuid sandbox

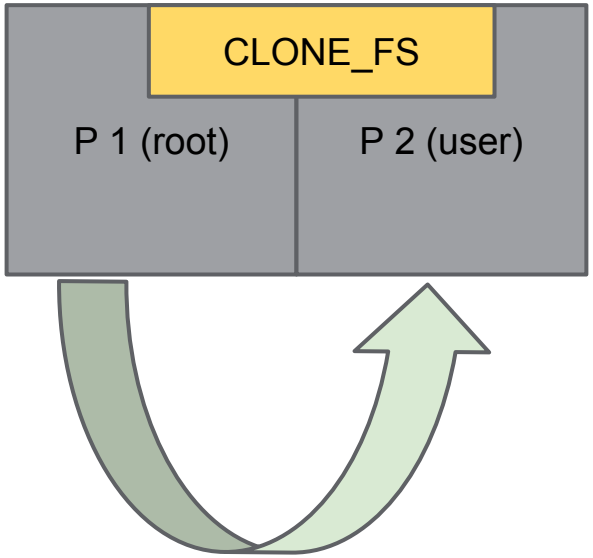


- Mais même UID. Comment protéger l'accès au système de fichiers?
  - Intégrité et confidentialité

chroot() par IPC



sandbox moi!



c’est fait!

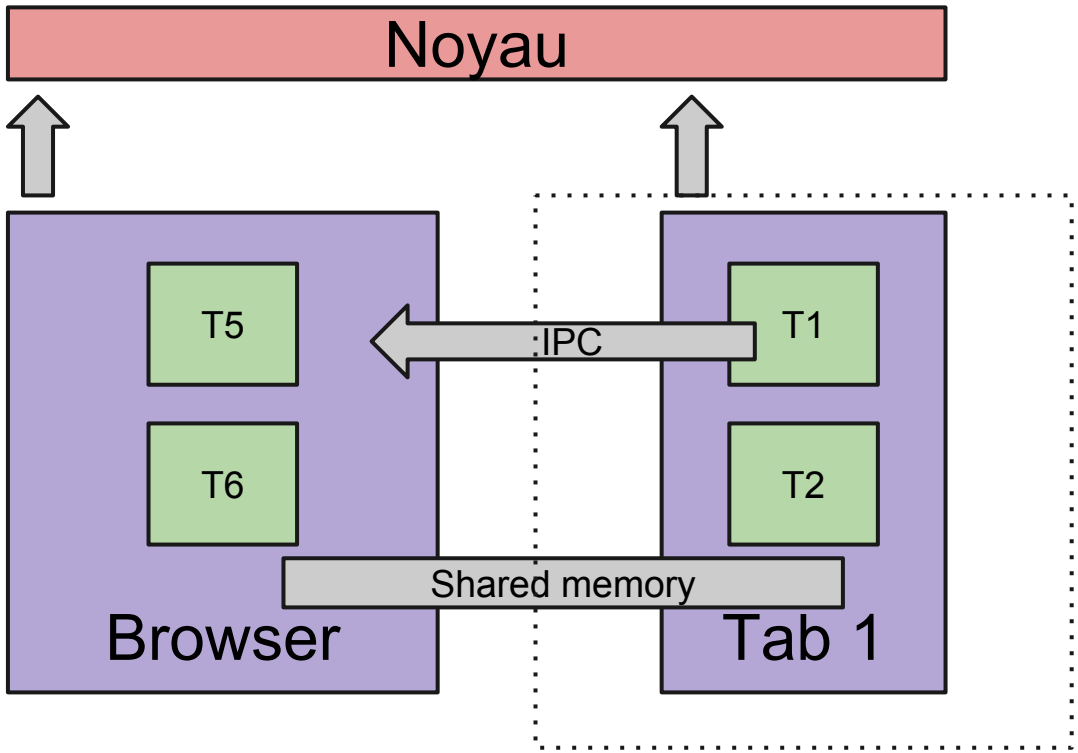
```
chroot("/proc/self/fdinfo/");
```

setuid sandbox

- Un binaire setuid
  - Compromis ingénierie vs. sécurité
- Binaire exécuté
  - dans son PID namespace
  - Et qui peut se chrooter() dans un répertoire vide à volonté

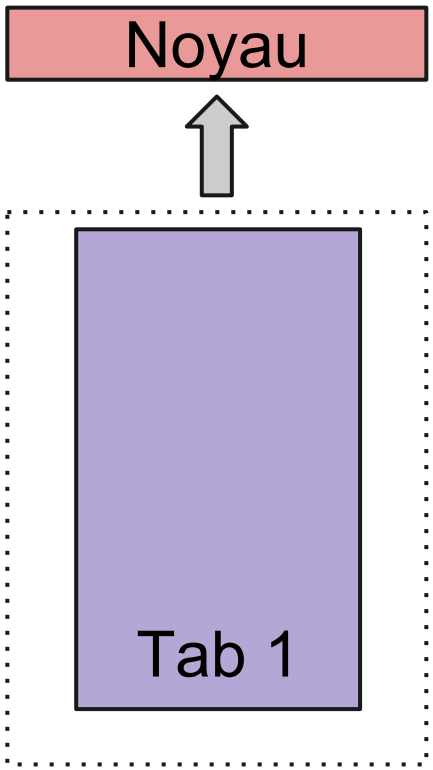
(Probablement obsolète dans Chrome 45 avec noyau récent)

Surface d’attaque



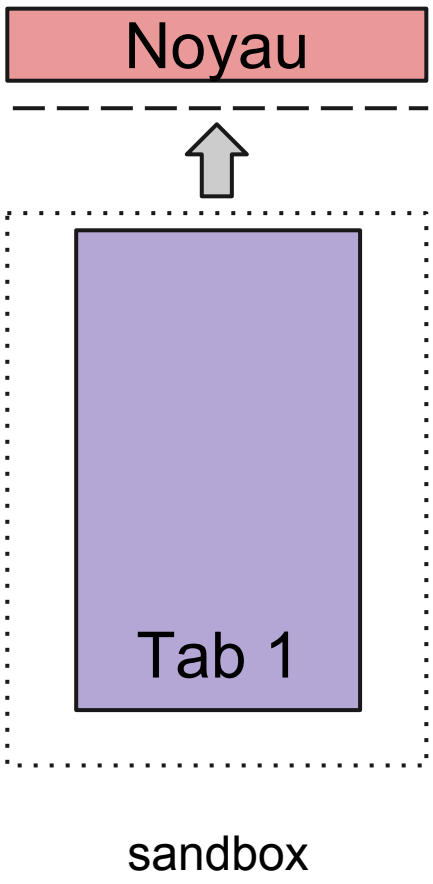
- IPC
- Noyau
  - Pas la plupart des drivers
  - Pas procfs etc...

Failles noyau

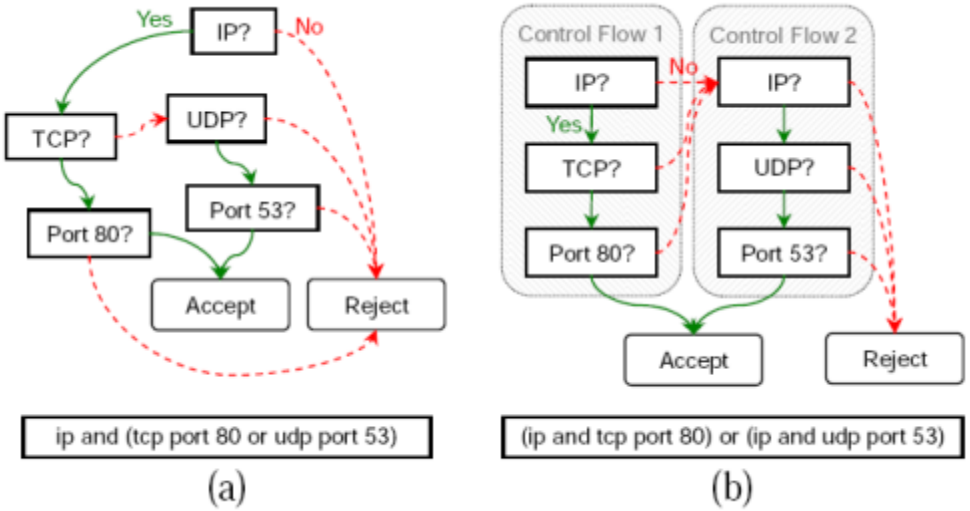


- Grosse surface d’attaque dans le noyau
  - Un très bon moyen de sortir de la sandbox

seccomp-bpf: un pare-feu



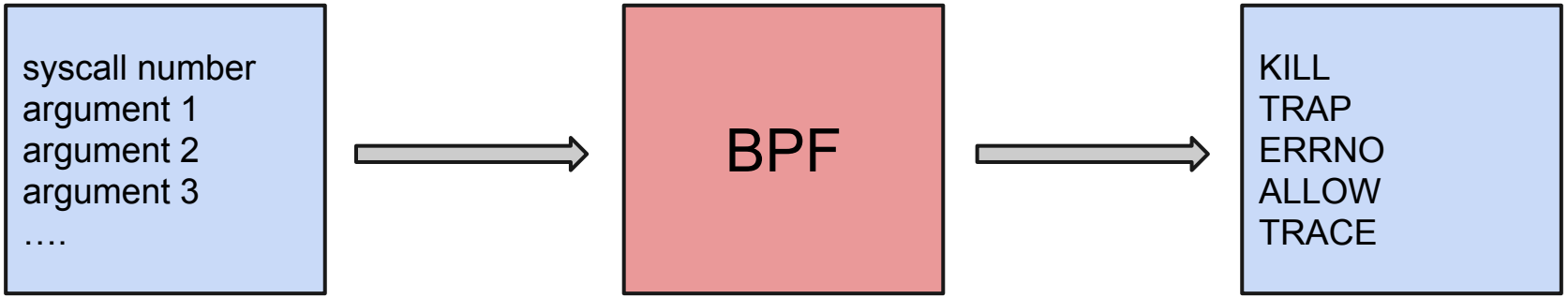
- Trop de surface exposée?
  - Pare-feu noyau!
  - Ajouté dans Chrome OS puis Linux 3.5



- Réutilisation du langage BPF

- On envoie un petit programme de filtrage d’ appels systèmes
  - Par processus (no_new_privs)
  - Hérité par fork() et execve()

Pare-feu BPF



- A chaque appel système, on décide: accepter ou rejeter
- TRAP: méchanisme puissant qui permet de gérer l'appel système en mode utilisateur

Ecrire nos programmes BPF

- renderer, NaCl, processus GPU, Flash
  - Ont des programmes BPF différents
  - Qui sont générés dynamiquement, à l'exécution
    - Nécessité de capturer des éléments dynamique comme le PID
- La complexité de ces programmes est considérable

BPF_DSL: un compilateur

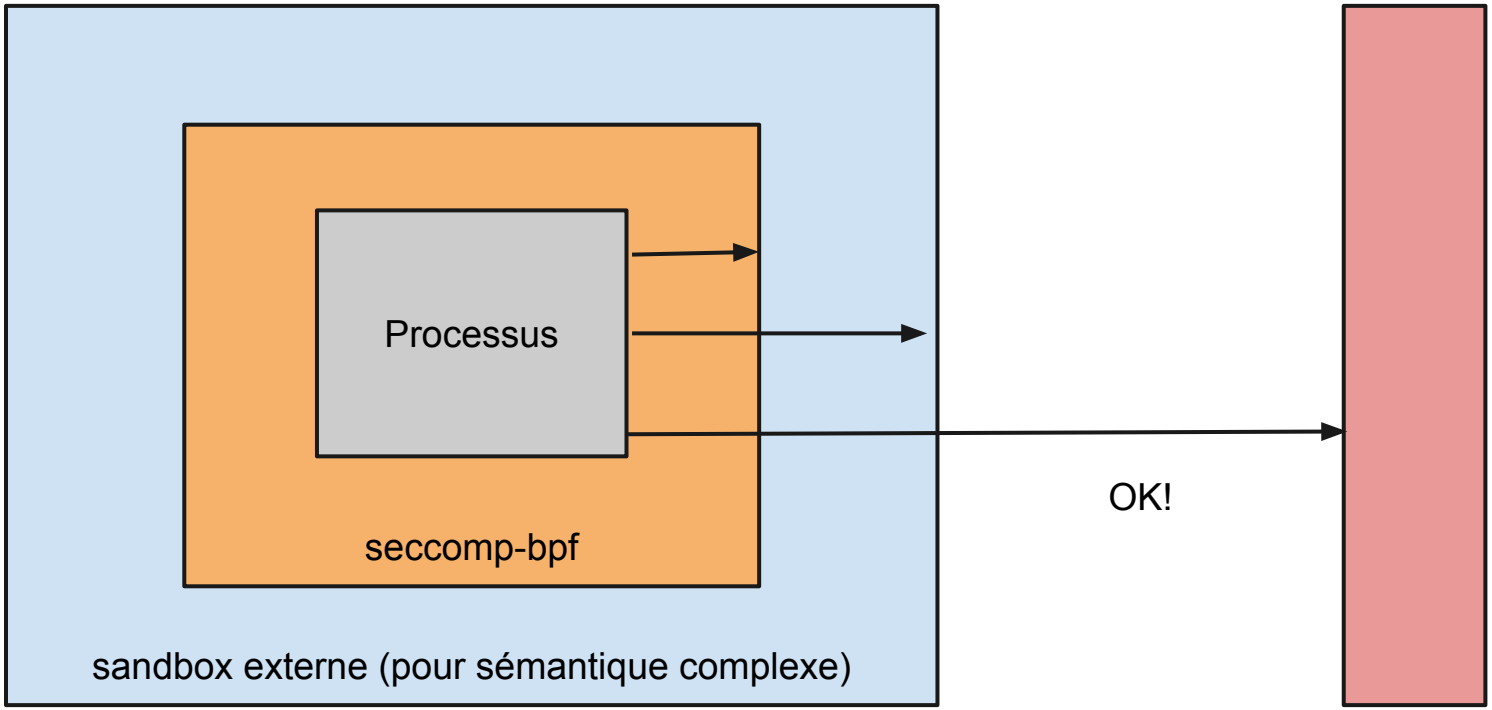
- “Domain specific language”, exprimé en C++
- Compilé dynamiquement à l’initialisation du processus (référence)

```
// int open(const char* name, int flags);
Arg<const char*> name(0);
Arg<int> flags(1);
return If(name != nullptr && (flags & O_CREAT) == 0,
          Allow()).Else(Error(EPERM));
```

sandbox avec seccomp

- Exprimer des sémantiques fortes avec seccomp?
  - “Ne peut pas violer l’intégrité et la confidentialité d’un autre processus”
  - “Ne peut pas accéder au système de fichiers”
  - “Ne peut pas accéder au réseau”
- Trop complexe, à éviter
  - Exemple: clock_getres/clock_gettime (https://crbug.com/374479)

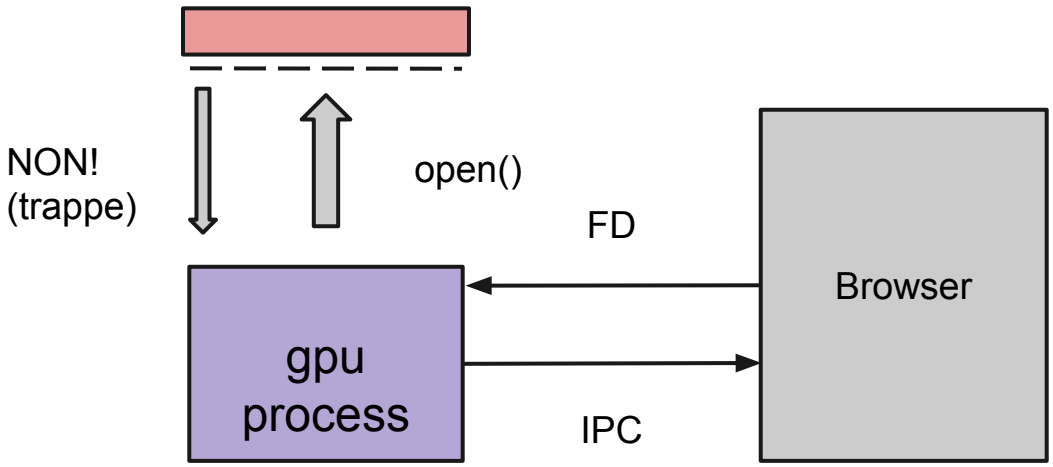
Règle n°2



Deux rôles différents!

Gérer le code existant

- Peut-on sandboxer du code existant avec seccomp?
- Oui, grâce au méchanisme de trappes



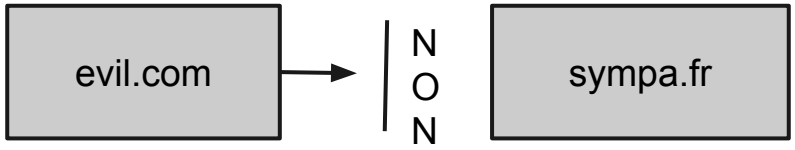
Chromium utilise des douzaines de bibliotèques développées par des tierces parties

Windows: win32k lockdown

- Nouveau Chrome 42 (14 Avril)
- SetProcessMitigationPolicy
- Windows 8+
  - DisallowWin32kSystemCalls
- Enlève la surface d’attaque de win32k.sys
- Ingénierie: a amélioré la stabilité!

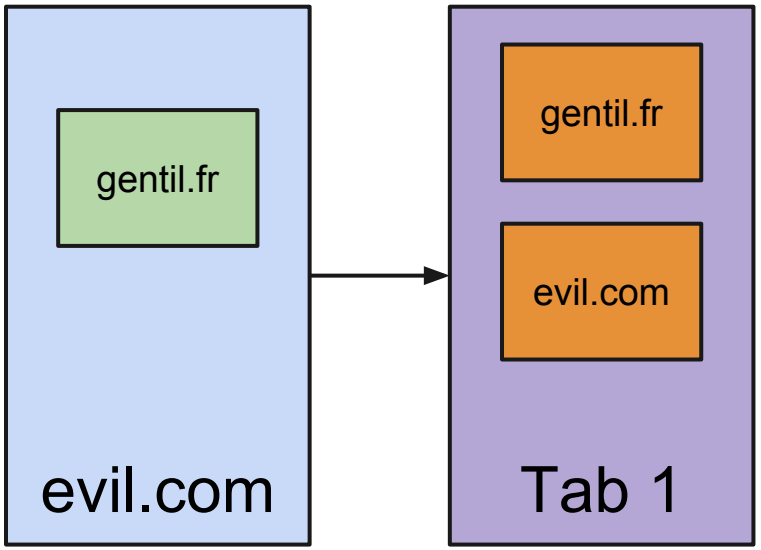
Site isolation

- La sandbox peut-elle garantir le modèle de sécurité du web?



- Problème:
  - Un processus par **tab**
  - Privilèges web par **origine**

Site isolation: attaque



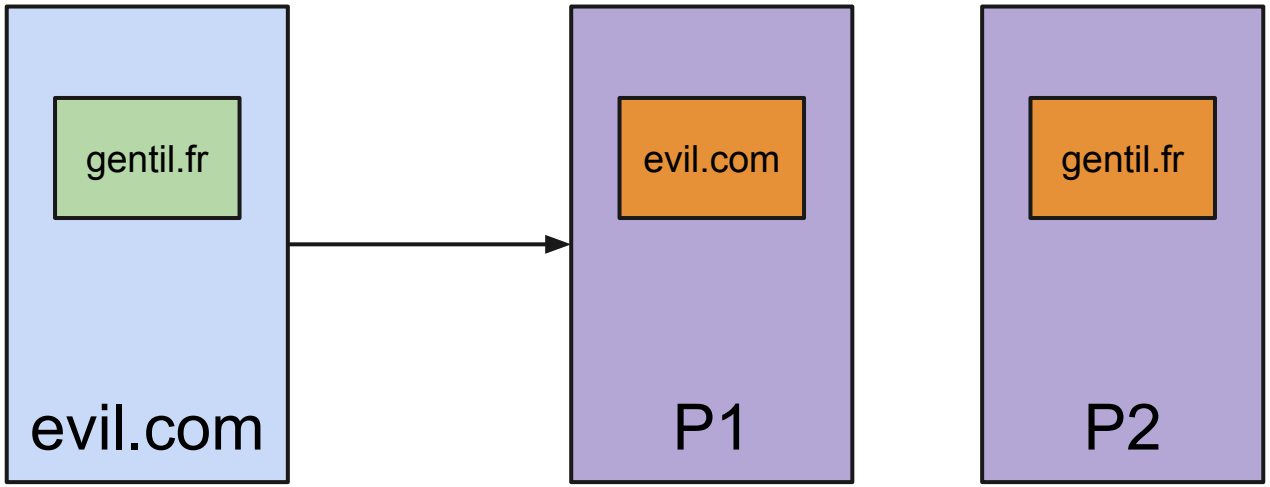
Page web

Processus “Tab 1”

- “Evil.com” inclue “gentil.fr” dans une iframe
- Le processus Tab1 a les privilèges de “evil.com” et “gentil.fr”
- evil.com exploite une faille “corruption mémoire” pour exécuter du code arbitraire dans Tab1
- Il peut utiliser les privilèges de “gentil.fr”

Site isolation

- Comment la sandbox peut garantir le modèle de sécurité du web?



Page web

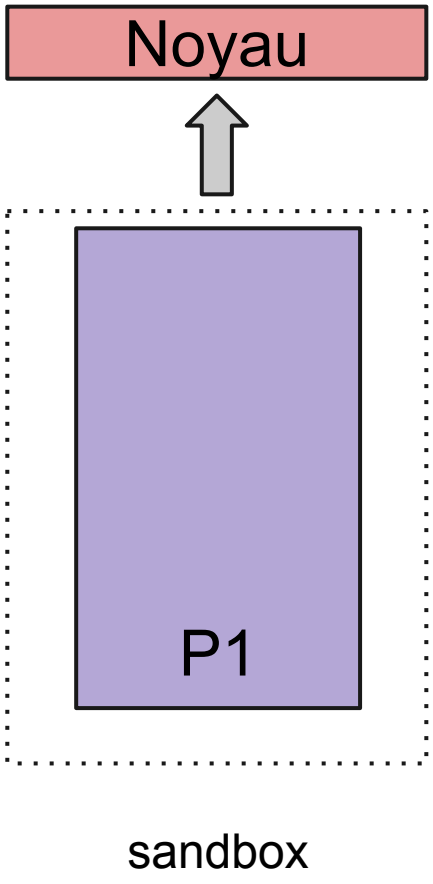
“Tab 1”

Site isolation

- L’ingénierie logicielle, au niveau système doit se coller au modèle de sécurité du web
  - Un **site** par processus
- Projet en cours

Utopie: si confiance totale dans la sandbox, on pourrait laisser un site web exécuter du code natif!

Surface d’attaque cachée



Rowhammer: les accès à la mémoire du processus peuvent générer des fautes dans la mémoire du reste du système

The dream?

Credit

Bugs--

- Ingénierie: tests, avec bonne couverture
- Problème: un bogue de corruption mémoire ne génère pas forcément de crash

- Solution: Sanitizers! (ASAN, TSANv2, UBSAN)
  - Utilisés au delà du projet Chromium

Bugs--

- Ingénierie: génération de tests automatisée?
- Solution: Fuzzing
- ClusterFuzz
  - ~6500 machines, 24h/24
  - Triage, tests de régression, création de rapports de bugs automatisé
- 2014: ~650 bugs trouvés automatiquement

PS: contrairement à ce que que certains pensent, le nombre de bogues publiés n'est pas une bonne métrique de sécurité

Bugs--

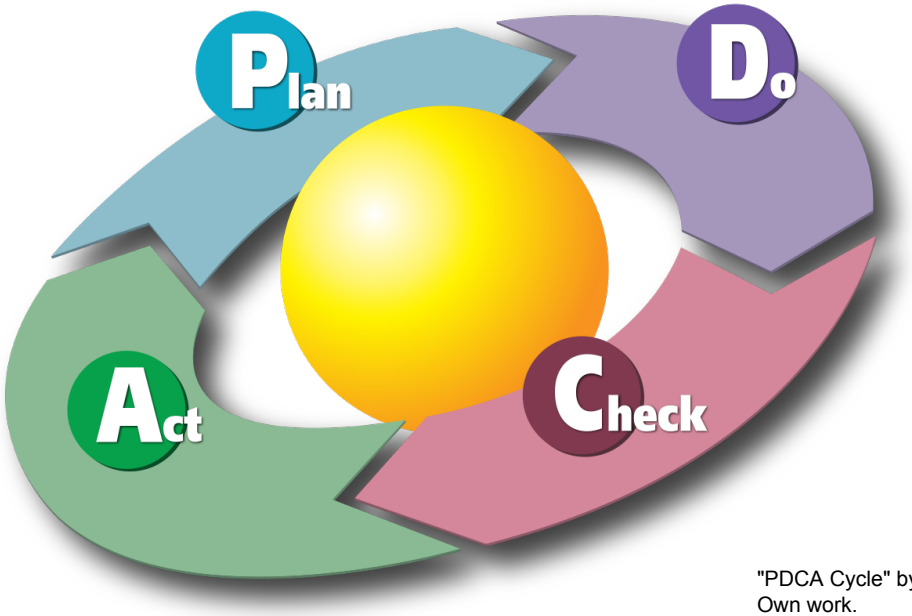
- Corriger les failles de sécurité est la responsabilité de tous
- ClusterFuzz utile, au delà de la sécurité, pour l’ingénierie logicielle
  - Plate-forme de combat contre l’entropie
  - Supplémente l’intégration continue classique

Mises à jour rapides

- Fondamental
- Pwnium: 24h entre chaine de bugs reçue et mise à jour des utilisateurs
- Courgette permet de faire des mises à jour différentielles sur des binaires

Bug bounty

- Un jour, j’ai suivi une formation de chef de projet



"PDCA Cycle" by Karn G. Bulsuk. Originally published here - Own work.

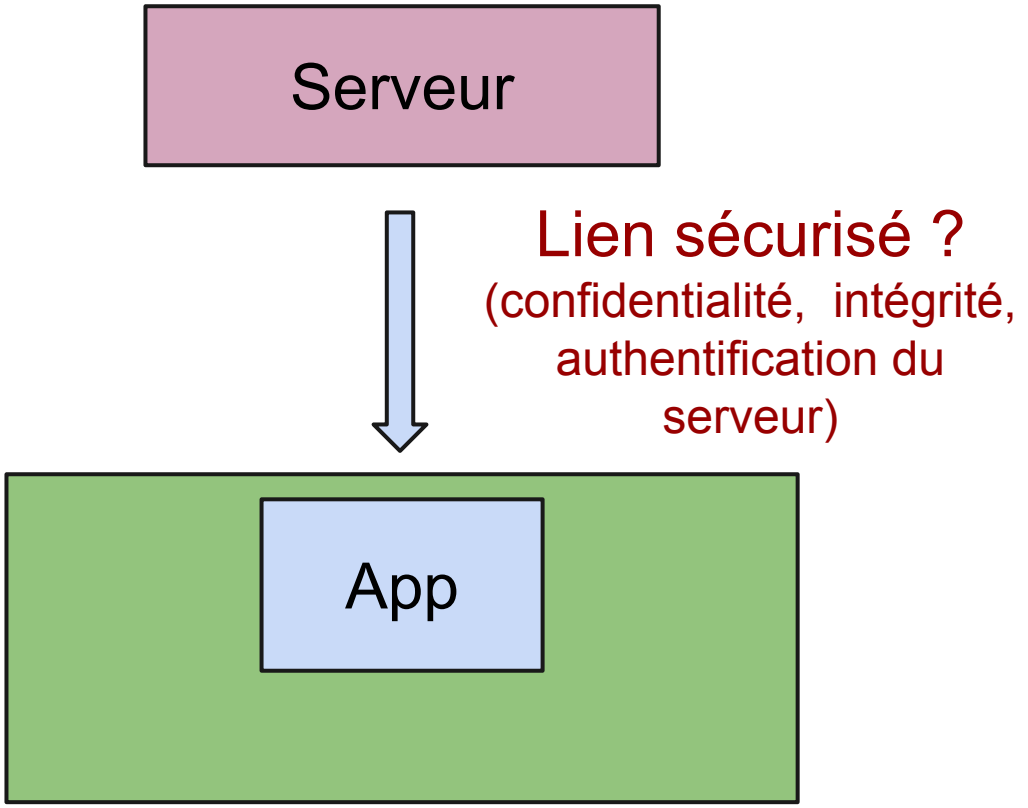
Bug bounty

- Permet d’obtenir des bogues
- Permet de garder les pieds sur terre
  - Comprendre où sont les points faibles
  - Quelles sont les techniques d’exploitation
- + de 1.75 million de $ de récompenses délivrées via
  - notre programme
  - ainsi que la compétition Pwnium (pour Chrome OS)

Mitigations diverses

- ASLR, NX, stack cookies, etc.
- PartitionAlloc
- **C**ontrol **f**low **i**ntegrity (CFI): sanitize=vptr (peut-être)
  - Rapidité vs sécurité peut être un conflit entre ingénierie et sécurité

The dream?


Serveur
Lien sécurisé ?
(confidentialité, intégrité, authentification du serveur)
App

SSL/TLS/x509 PKI

Credit

Lien sécurisé: essayons (1)

- Certificate pinning (CAs can't CA)
- Certificate transparency (Ditto + se font hacker + erreurs humaines)
- CRLSets (CRLs ont trop de latence)
- CBC est cassé (Encrypt(MAC(Plaintext)))
  - Vaudenay, Poodle, Lucky-13, BEAST
  - Obligés d'utiliser RC4 sur TLS 1.0
  - AES-GCM ou CHACHA20_POLY1305 pour ne pas être considéré obsolète

Lien sécurisé: essayons (2)

- RSA < 1024 obsolète (bientôt <= 2048 pour CAs)
- DHE < 512 bits obsolète
  - Bientôt < 1024, DHE lui-même obsolète (utilisez ECDHE)
- SHA-1 obsolète

Lien sécurisé: essayons (3)

- On peut rétrograder les versions TLS
  - TLS Fallback Signaling Cipher Suite Value
  - Mise à la retraite de SSLv3

Références

Autres problèmes

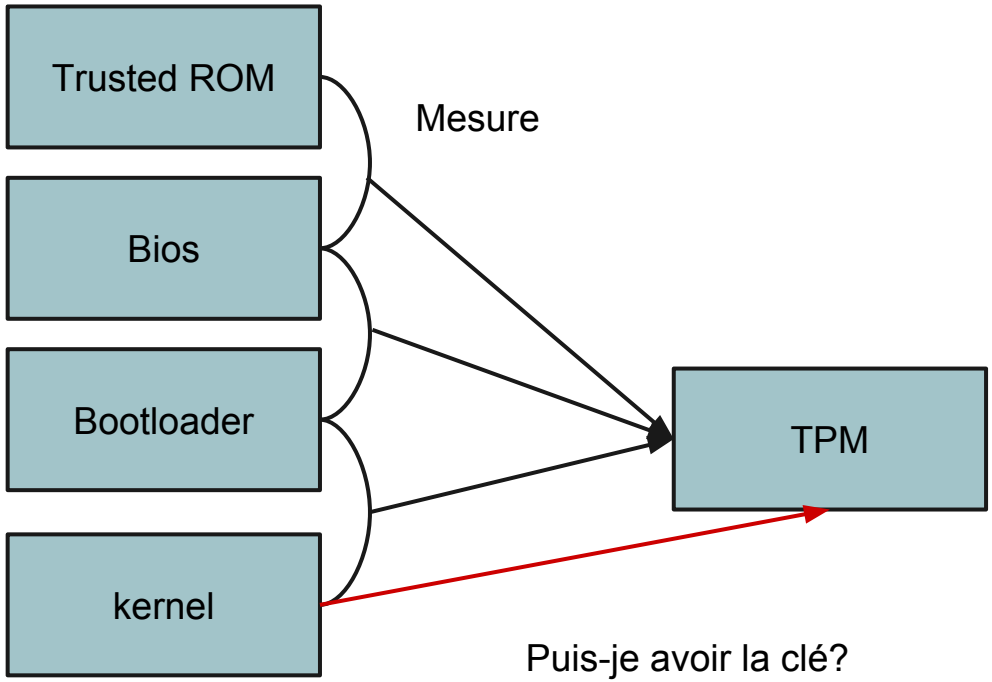
- Bloquer le “mixed scripting”
  - “Mixed pinning”?
- Quelle est la cloture transitive de votre site par la relation “X inclut <script src=Y…>?

The dream?

Chrome OS

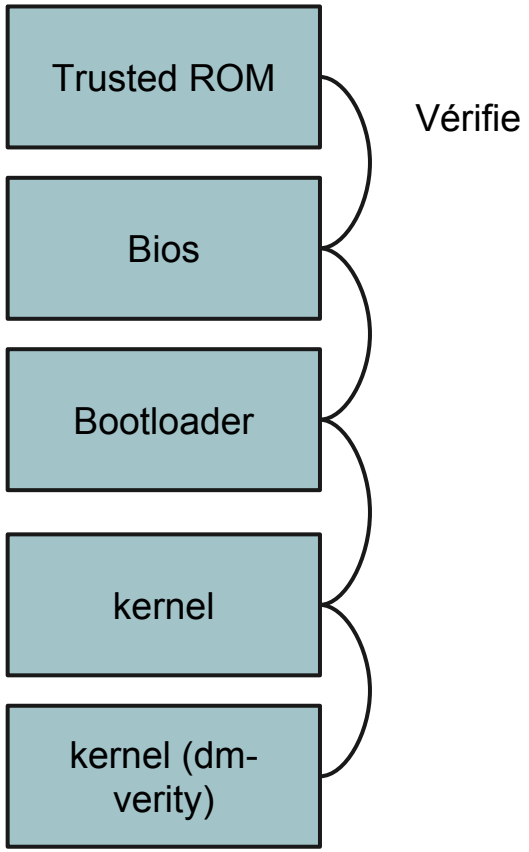
- Simple, fast, secure
- Contrôle du noyau, du système
  - Sandboxing
  - Kernel ASLR
  - Toolchain hardening
- Démarre sur une base saine
  - Verified Boot

Trusted boot (classique)



- Même backdooré, le système démarre!
- Il peut demander un mot de passe à l'utilisateur puis redémarrer (Evil Maid attack)

Verified boot (Chrome OS)



- Si quoi que ce soit est corrompu, le système ne redémarre pas
- (Le système est toujours installé en double, en cas de corruption il redémarrera sur l’ autre copie).

Chiffrement de disque (Chrome OS)

- Par utilisateur
- Un utilisateur malveillant, même en passant root ou kernel, ne peut pas lire les données des autres utilisateurs
- Si vous utilisez un Chromebook inconnu:
  - Rebootez
  - Attaquant logiciel, avec exécution en mode kernel, doit avoir déjoué le “verified boot” pour déchiffrer vos données

Chiffrement de disque (Chrome OS)

- Perte de Chromebook dans un Taxi
- Après avoir démonté le disque dur
  a. Attaque par force brute du mot de passe doit passer par le TPM (très lent)
  b. Extraire les clés du TPM n'est pas suffisant pour déchiffrer les données de l'utilisateur ou retrouver le mot de passe
- Comment?
  a. https://goo.gl/wlhbNG
  b. https://crbug.com/454638

Etat du disque au démarrage

- Partie “stateless” vérifiée
- Partie chiffrée par utilisateur pas accessible
- Partie “stateful” très petite

- Ingénierie logicielle: le processus de boot a peu de dépendances non constantes
  - On peut reproduire ce qui se passe
  - Si quelque chose de subtil est corrompu, on ne démarre pas (et on démarre sur la partition de rechange)

The dream?

Credit

PBCK?

PBCK?

- ~~**P**roblem **B**etween **C**hair and **K**eyboard~~
  - Idée fausse
  - A nous de ne pas présenter de décision de sécurité à l'utilisateur
- Qui peut décrire comment trouver l'origine dans une URL?
  - httpS://ou.est.l:origine@dans.cette.url:443/ici?oula=peut-etre

Efforts

- Enamel
  - Erreurs de certificats
  - Origines plus claires
  - Marque http comme dangereux
- U2F
  - Cryptographie à clé publique -> pas de phishing
- Password manager

Credit

The dream?

- Navigateurs web
  - Plateforme d’exécution d’ applications sécurisée
  - Défi difficile

- Ingénierie de sécurité importante
- Dans de nombreux cas: sécurité => qualité
- Ingénierie de sécurité et ingénierie logicielle classique peuvent faire bon ménage

The dream?

- Navigateurs web
  - Plateforme d'exécution d' application**s** sécurisé**e**

- Isoler les applications web du système (et entre elles) ne les sécurise pas
  - De nombreux efforts pour améliorer le modèle de sécurité du web

Merci! <security-dev@chromium.org> (public)

Slides bonus

Channel ID

Référence

NaCl

Référence

Publication technique