

Module 0: Introduction à la Sécurité

Méthodes de la Sécurité des Systèmes

Université Claude Bernard Lyon 1

Hugo Blanc
CC BY-SA 4.0

2026 - 2027

1. Définir la sécurité
2. Modéliser les risques
3. The Attacker Mindset
4. En résumé

0.1 Prérequis

- Avoir une machine GNU/Linux en état de fonctionnement
- Avoir un utilisateur différent de root appartenant au groupe sudo
- Avoir une connexion à Internet
- Savoir naviguer dans un terminal UNIX
- Connaître les bases des réseaux informatiques TCP/IP

0.2 Où trouver le cours ?

Tout est disponible sur mon site web: <https://syscall.cafe/t/>

0.3 Évaluation

Ce cours sera évalué par :

- Participation et implication en classe;
- Un TP noté à faire à la maison;
- Un DS de 2h en fin de cours.

À certains endroits du cours, généralement en fin de chapitre, vous trouverez des blocs :



Aide aux révisions

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua quaerat.

Ils contiennent des exercices-types des évaluations.

- L'utilisation des LLMs (et autres outils) est **déconseillée** car ils ne participent pas à la réflexion et à l'apprentissage.
- Pas de sanctions si usage des “modéré” des LLMs pour les **TP notés seulement**. Des questions orales de validation des acquis peuvent être posées lors de la restitution.
- Toute aide durant les contrôles sur table sera **considérée comme de la triche**, et mènera à une note de zéro.

1. Définir la sécurité
2. Modéliser les risques
3. The Attacker Mindset
4. En résumé

1.1 Qu'est-ce qu'un système ?

Selon Wikipédia:

“Un système est un ensemble d'**éléments interagissant entre eux** selon certains principes et règles. Par exemple une molécule, le système solaire, une ruche, une société humaine, un parti, une armée etc.”

1.1 Qu'est-ce qu'un système ?

Voici un système:



1.1 Qu'est-ce qu'un système ?

Voici un système:

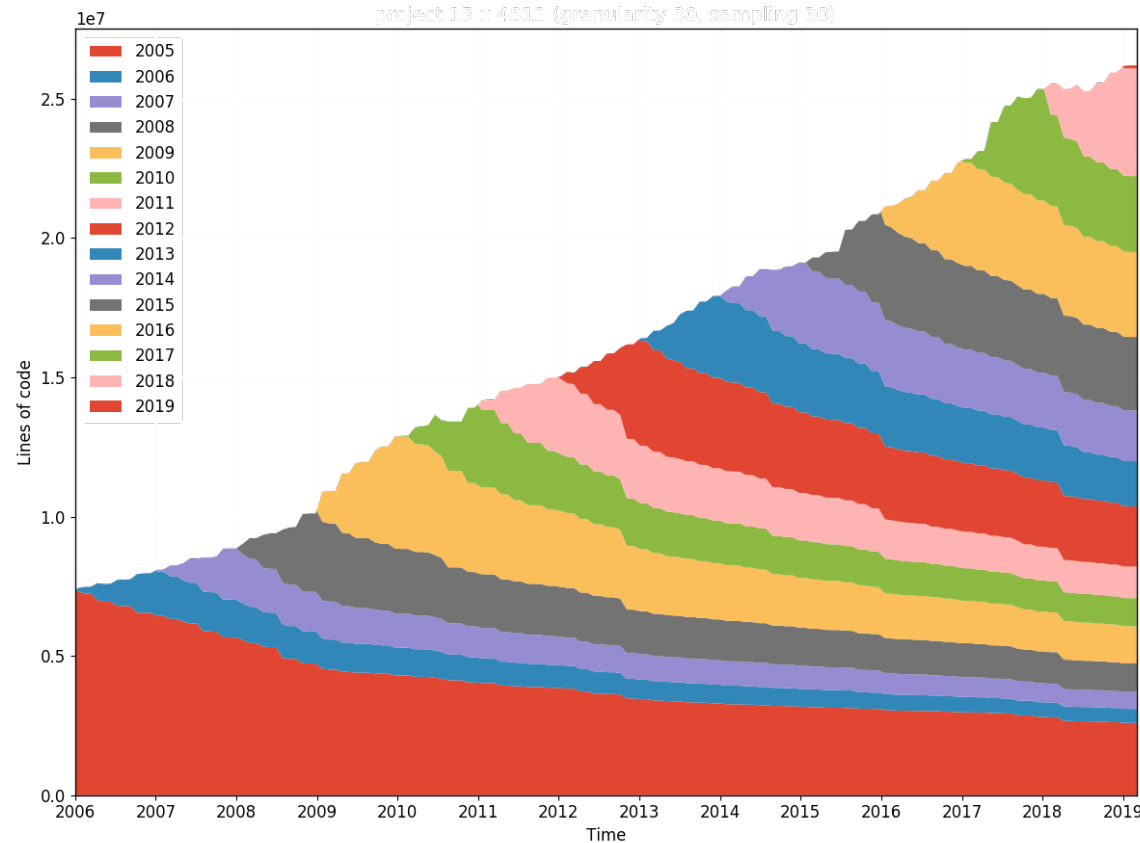


Question Quelles attaques pouvons-nous mettre en place pour abuser de ce système ? Comment les fixer ?

1.1 Qu'est-ce qu'un système ?

Un système n'est jamais plus sécurisé que son maillon le plus faible !

1.2 Des systèmes informatiques de plus en plus complexes



Le code écrit il y a plus de 20 ans ne correspond plus aux standards de sécurité actuels !

Figure 2: Nombre de lignes de code dans le kernel Linux en fonction des années.

1.3 Triade CIA

La triade CIA est un concept qui permet de définir sur quoi la sécurité d'un SI doit se concentrer:

- la **Confidentialité** (*confidentiality*);

1.3 Triade CIA

La triade CIA est un concept qui permet de définir sur quoi la sécurité d'un SI doit se concentrer:

- la **Confidentialité** (*confidentiality*);
- l'**Intégrité** (*integrity*);

1.3 Triade CIA

La triade CIA est un concept qui permet de définir sur quoi la sécurité d'un SI doit se concentrer:

- la **Confidentialité** (*confidentiality*);
- l'**Intégrité** (*integrity*);
- la **Disponibilité** (*availability*).

Question Quel(s) aspect(s) de la triade CIA est (sont) impacté(s) lors des incidents de sécurité suivants:

- Dénî de service distribué (*DDoS*) ?
- Injection SQL ?
- Inondation dans le datacenter ?
- Rançongiciel (*ransomware*) ?

1. Définir la sécurité
- 2. Modéliser les risques**
3. The Attacker Mindset
4. En résumé

2.1 Vulnérabilités

Pour faire en sorte que les trois critères de la triade soient respectés, il faut limiter les **vulnérabilités**.

Une vulnérabilité est une faille d'origine diverse (bug, laisser-aller...) qui crée une **faiblesse** qu'une **menace** peut exploiter.

2.1 Vulnérabilités

2.1.1 CVE

Les vulnérabilités connues sont standardisées à l'aide d'un identifiant CVE (*Common Vulnerabilities and Exposures*)¹.

Chaque vulnérabilité reçoit un identifiant unique au format: CVE-YYYY-NNNN:

- YYYY : année de publication
- NNNN : numéro séquentiel (au moins 4 chiffres)

¹https://en.wikipedia.org/wiki/Common_Vulnerabilities_and_Exposures

2.1 Vulnérabilités

2.1.2 CVSS

Le score CVSS (*Common Vulnerability Scoring System*) évalue la gravité d'une vulnérabilité sur une échelle de 0 à 10¹.

Score	Sévérité
0.0	None
0.1 - 3.9	Low
4.0 - 6.9	Medium
7.0 - 8.9	High
9.0 - 10.0	Critical

¹<https://nvd.nist.gov/vuln-metrics/cvss>

2.1 Vulnérabilités

Le score CVSS se base sur 8 métriques principales¹:

- Vecteur d'attaque
- Complexité d'attaque
- Privilèges requis
- Interaction utilisateur
- Portée
- Impact confidentialité
- Impact intégrité
- Impact disponibilité

¹https://en.wikipedia.org/wiki/Common_Vulnerability_Scoring_System#Version_4.0

2.2 Menaces et risques

Une **menace** est un danger possible qui peut exploiter une vulnérabilité pour outrepasser des mesures de sécurité.

“A potential cause of an incident that may result in harm of systems and organization. - ISO 27005¹”

¹https://en.wikipedia.org/wiki/ISO/IEC_27005

2.2 Menaces et risques

Une **menace** est un danger possible qui peut exploiter une vulnérabilité pour outrepasser des mesures de sécurité.

“A potential cause of an incident that may result in harm of systems and organization. - ISO 27005¹”

Les menaces peuvent être **intentionnelles** (*insiders*, attaquant.e...) ou **accidentelles** (environnement...).

¹https://en.wikipedia.org/wiki/ISO/IEC_27005

2.2 Menaces et risques

Le **risque** peut être défini comme suit:

$$\text{Risque} = \text{Menaces} \times \text{Vulnerabilites}$$

Si les menaces ou le nombre de vulnérabilités dans notre SI augmentent, alors le risque augmente également.

Le corollaire est que si nous arrivons à réduire un de ces facteurs (ou les deux !), alors le risque général diminue.

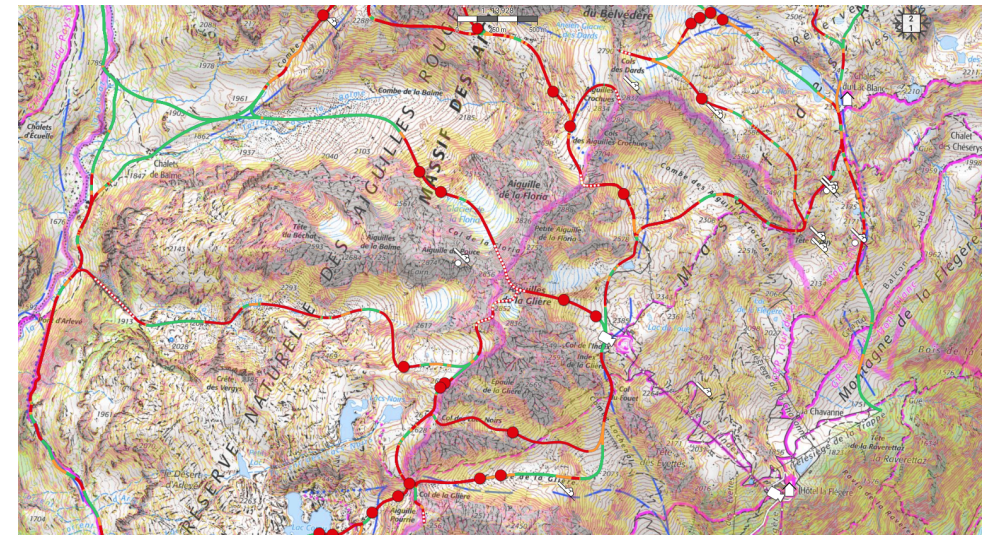
2.3 Threat modeling

Processus d'identification de menaces potentielles du point de vue *hypothétique* de l'adversaire.

Faire du threat modeling, c'est répondre à la question: “Qu'est-ce qui pourrait mal se passer ?”

2.3 Threat modeling

Des variantes au threat modeling sont déjà pratiquées au quotidien sans s'en rendre compte.



2.3 Threat modeling

Un threat model est *ad hoc*: ses paramètres évoluent constamment dans le temps.



2.3 Threat modeling

Dans le monde de l'infosec, il existe des frameworks (des méthodes) facilitant le processus de threat modeling, les principaux étant:

- STRIDE
- PASTA

2.3 Threat modeling

2.3.1 STRIDE

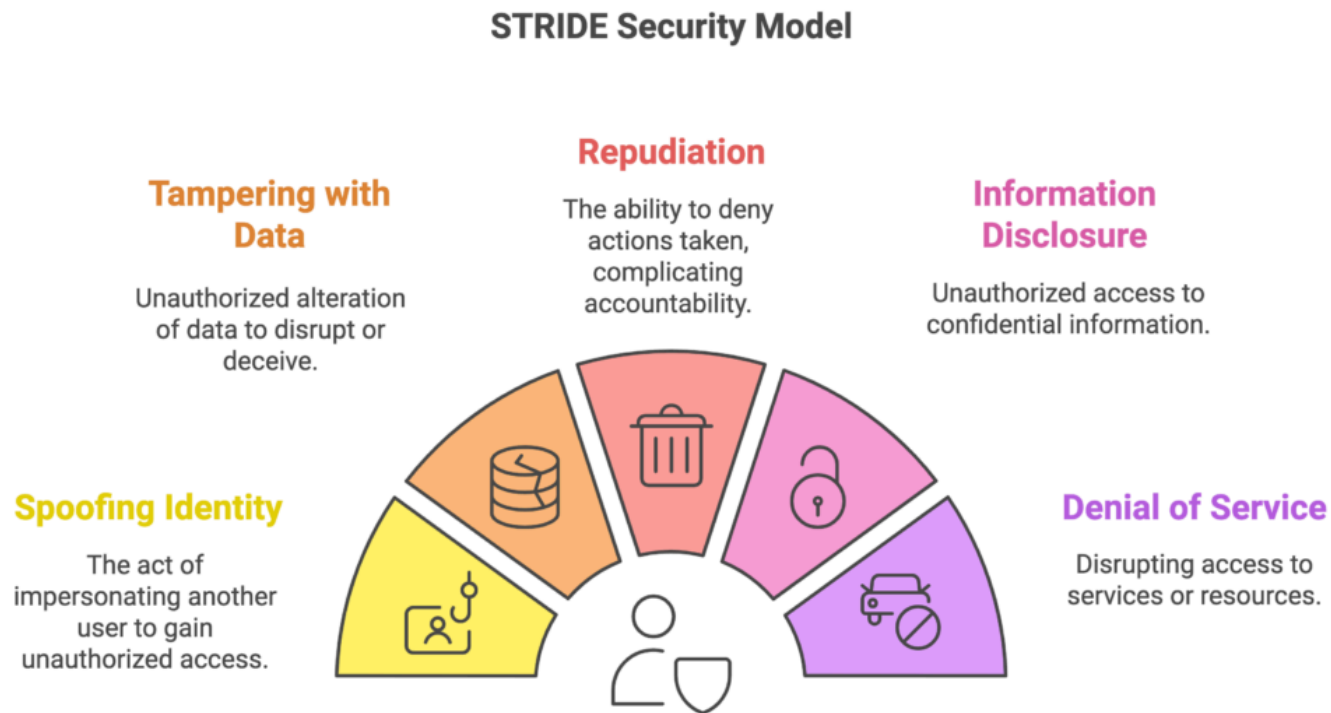


Figure 6: <https://www.practical-devsecops.com/what-is-stride-threat-model/>

1. Définir la sécurité
2. Modéliser les risques
- 3. The Attacker Mindset**
4. En résumé

3.1 Deux visions opposées du logiciel

Les dévs et les attaquant.e.s regardent le même code, mais en miroir¹:

- Les dévs développent selon un cahier des charges: focus sur ce que le code **doit faire**;

¹Chapitre adapté de *Thinking Like an Attacker, Owning the Bits*, University of Wisconsin-Madison: <https://research.cs.wisc.edu/mist/SoftwareSecurityCourse/>

3.1 Deux visions opposées du logiciel

Les dévs et les attaquant.e.s regardent le même code, mais en miroir¹:

- Les dévs développent selon un cahier des charges: focus sur ce que le code **doit faire**;
- Les efforts de test et de debug portent sur les features qui doivent fonctionner;

¹Chapitre adapté de *Thinking Like an Attacker, Owning the Bits*, University of Wisconsin-Madison: <https://research.cs.wisc.edu/mist/SoftwareSecurityCourse/>

3.1 Deux visions opposées du logiciel

Les dévs et les attaquant.e.s regardent le même code, mais en miroir¹:

- Les dévs développent selon un cahier des charges: focus sur ce que le code **doit faire**;
- Les efforts de test et de debug portent sur les features qui doivent fonctionner;
- Très peu de focus sur ce que le code **peut faire d'imprévu** !

¹Chapitre adapté de *Thinking Like an Attacker, Owning the Bits*, University of Wisconsin-Madison: <https://research.cs.wisc.edu/mist/SoftwareSecurityCourse/>

3.1 Deux visions opposées du logiciel

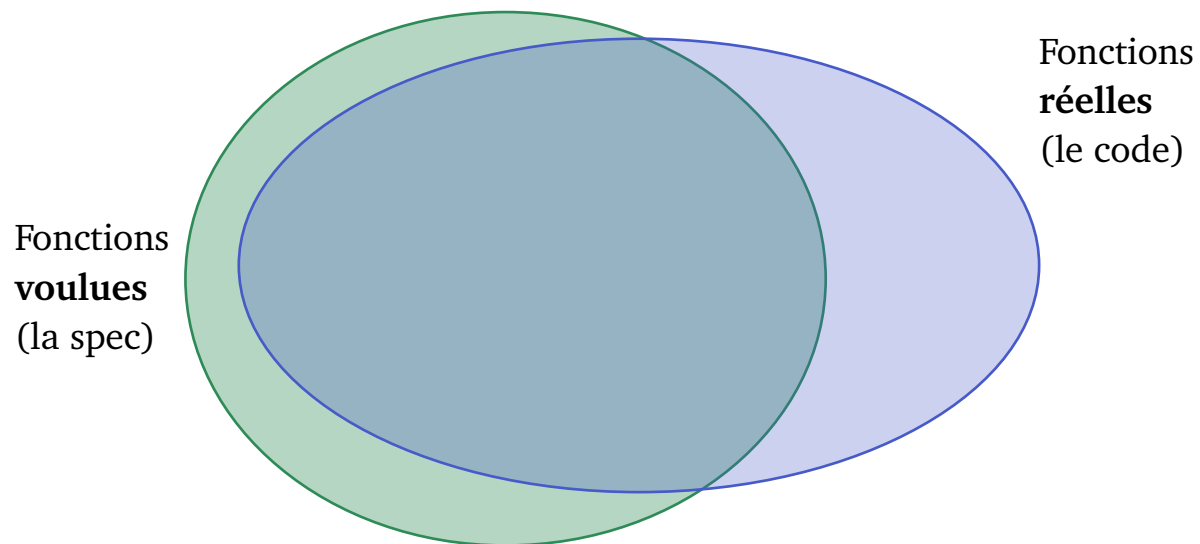


Figure 8: Fonctionnement idéal vs. réel d'un logiciel.

3.1 Deux visions opposées du logiciel

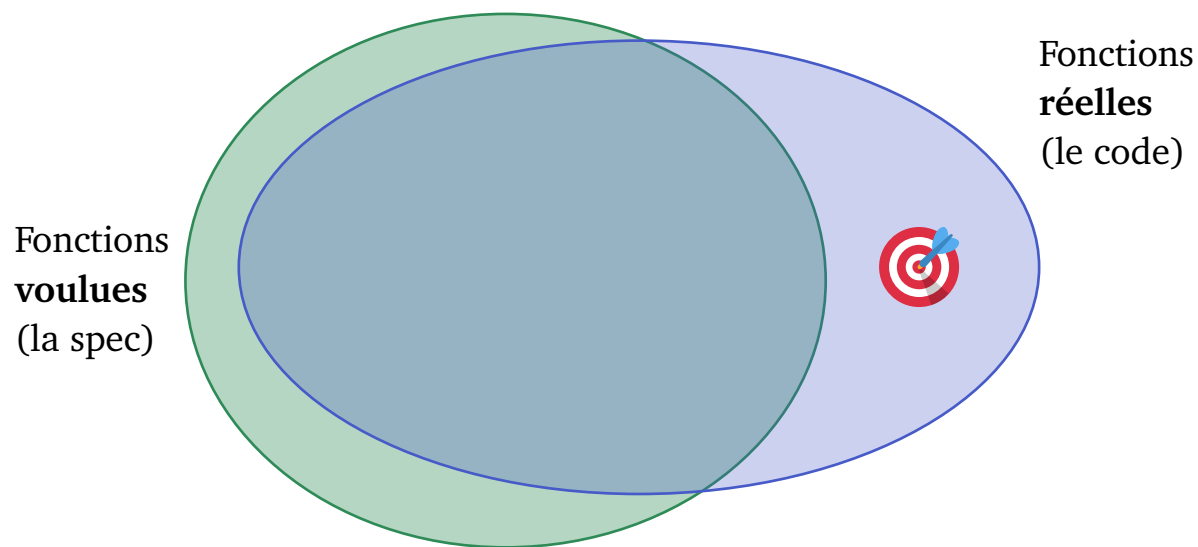


Figure 9: "Fonctionnalités" cibles des attaquants.

La zone blueue: des fonctionnalités **jamais voulues**, invisibles pour lea dév, et que les tests fonctionnels ne révèlent pas.

3.2 Owning the bits

Un serveur web public copie le contenu d'un formulaire en mémoire pour le traiter:

- N'importe qui sur Internet peut soumettre le formulaire;
- La chaîne envoyée est copiée fidèlement dans un buffer.

3.2 Owning the bits

Un serveur web public copie le contenu d'un formulaire en mémoire pour le traiter:

- N'importe qui sur Internet peut soumettre le formulaire;
- La chaîne envoyée est copiée fidèlement dans un buffer.

L'attaquant.e choisit exactement quels bytes apparaissent dans la mémoire du serveur: iel **possède les bits** (*owns the bits*).

Tant que la chaîne est traitée sans danger, aucun problème !

3.2 Owning the bits

Mais si lea dév a dimensionné le buffer à 100 bytes, “parce que personne n’écrit jamais aussi long”...

3.2 Owning the bits

Mais si lea dév a dimensionné le buffer à 100 bytes, “parce que personne n’écrit jamais aussi long”...

En envoyant une chaîne démesurée, l’attaquant.e possède maintenant les bits **au-delà du buffer**, y compris des zones mémoire très intéressantes.

C’est le classique **dépassement de tampon** (*buffer overflow*).

3.3 Anatomie d'une attaque

Une attaque est une chaîne d'événements:

- **Surface d'attaque** (*attack surface*): les interfaces exposées, où l'attaque peut commencer;

3.3 Anatomie d'une attaque

Une attaque est une chaîne d'événements:

- **Surface d'attaque** (*attack surface*): les interfaces exposées, où l'attaque peut commencer;
- **Chemin d'attaque** (*attack path*): la séquence d'actions menée par l'attaquant.e;

3.3 Anatomie d'une attaque

Une attaque est une chaîne d'événements:

- **Surface d'attaque** (*attack surface*): les interfaces exposées, où l'attaque peut commencer;
- **Chemin d'attaque** (*attack path*): la séquence d'actions menée par l'attaquant.e;
- **Vulnérabilité(s)**: les faiblesses exploitées en chemin;

3.3 Anatomie d'une attaque

Une attaque est une chaîne d'événements:

- **Surface d'attaque** (*attack surface*): les interfaces exposées, où l'attaque peut commencer;
- **Chemin d'attaque** (*attack path*): la séquence d'actions menée par l'attaquant.e;
- **Vulnérabilité(s)**: les faiblesses exploitées en chemin;
- **Surface d'impact** (*impact surface*): l'ensemble des dégâts finaux que l'attaquant.e peut causer.

3.3 Anatomie d'une attaque

3.3.1 Braquer une banque

Question Vous devez braquer une banque. Quels chemins d'attaque imaginez-vous ? Quelles faiblesses exploitent-ils ?

3.3 Anatomie d'une attaque

Par la porte principale: entrer en client.e, distraire lea caissier.ère grâce à un.e complice, se servir dans le tiroir-caisse.

Faiblesses: tiroir-caisse à portée des client.e.s, inattention, absence de surveillance, fuite possible.

3.3 Anatomie d'une attaque

Par la porte principale: entrer en client.e, distraire lea caissier.ère grâce à un.e complice, se servir dans le tiroir-caisse.

Faiblesses: tiroir-caisse à portée des client.e.s, inattention, absence de surveillance, fuite possible.

Par la porte arrière: une entrée de service est restée ouverte, la salle des coffres aussi.

Faiblesses: porte non verrouillée, coffre ouvert, argent qui traîne.

3.3 Anatomie d'une attaque

Sans braquage: un.e employé.e malhonnête glisse discrètement du liquide dans son sac avant de rentrer chez ellui.

C'est une **menace interne** (*insider threat*): impossible à éliminer, les employé.e.s ont *besoin* d'accéder à l'argent pour travailler.

On la **mitige** par la redevabilité: audits, inspections, traçabilité des transactions.

3.4 La surface d'attaque

Tout ce qu'un.e utilisateur.rice (ou attaquant.e) peut employer pour influencer le comportement du système.

Pour la banque: portes, fenêtres, murs, toit, tunnels, conduits techniques...

3.4 La surface d'attaque

Tout ce qu'un.e utilisateur.rice (ou attaquant.e) peut employer pour influencer le comportement du système.

Pour la banque: portes, fenêtres, murs, toit, tunnels, conduits techniques...

Pour un programme typique:

- Données reçues du réseau;
- Fichiers d'entrée;
- Champs de formulaires web;
- Entrées en base de données;
- Configuration.

3.4 La surface d'attaque

La surface d'attaque inclut aussi tout ce que l'attaquant.e peut **observer** de l'extérieur: pages publiques, scripts, messages d'erreur...

Et pour un logiciel open source: le code lui-même, qui révèle comment accéder au système et l'influencer.

3.5 Attaques directes et indirectes

Attaque directe l'entrée fournie sur la surface d'attaque déclenche directement la vulnérabilité. C'est le cas le plus courant.

Attaque indirecte l'entrée provoque des changements d'état, qui en provoquent d'autres... jusqu'aux dégâts, plusieurs étapes plus loin.

3.5 Attaques directes et indirectes

Exemple d'attaque indirecte:

1. Un service web valide toutes les entrées, rejette la requête malveillante... et la **log**;

3.5 Attaques directes et indirectes

Exemple d'attaque indirecte:

1. Un service web valide toutes les entrées, rejette la requête malveillante... et la **log**;
2. La requête a été conçue pour produire une entrée de log **malformée**;

3.5 Attaques directes et indirectes

Exemple d'attaque indirecte:

1. Un service web valide toutes les entrées, rejette la requête malveillante... et la **log**;
2. La requête a été conçue pour produire une entrée de log **malformée**;
3. Un cronjob d'analyse de logs lit cette entrée, et une vulnérabilité s'y déclenche.

3.5 Attaques directes et indirectes

Exemple d'attaque indirecte:

1. Un service web valide toutes les entrées, rejette la requête malveillante... et la **log**;
2. La requête a été conçue pour produire une entrée de log **malformée**;
3. Un cronjob d'analyse de logs lit cette entrée, et une vulnérabilité s'y déclenche.

Le job d'analyse, “jamais exposé à Internet”, est pourtant atteignable depuis la surface d'attaque !

3.6 Suivre le flot du programme

Le chemin d'attaque se propage par le **flot de données** et le **flot de contrôle**:

```
if (buffer[i] > 10)
    val = 3;
else
    val = 25;
```

Si *i* vient de la surface d'attaque, l'attaquant.e influence *val*... alors que les deux variables n'apparaissent jamais dans la même instruction.

3.6 Suivre le flot du programme

Le flot du programme étend la portée de l'attaquant.e bien au-delà de ce qu'on imagine, même depuis une surface d'attaque minimale.

Minimiser la surface d'impact est donc une stratégie défensive à part entière.

3.7 La surface d'impact

L'endroit du code où l'exploit prend effet est un **point d'impact**. Par exemple:

- Exécution arbitraire (obtenir un shell);

3.7 La surface d'impact

L'endroit du code où l'exploit prend effet est un **point d'impact**. Par exemple:

- Exécution arbitraire (obtenir un shell);
- Élévation de privilèges;

3.7 La surface d'impact

L'endroit du code où l'exploit prend effet est un **point d'impact**. Par exemple:

- Exécution arbitraire (obtenir un shell);
- Élévation de privilèges;
- Usurpation d'identité (deviner un mot de passe faible);

3.7 La surface d'impact

L'endroit du code où l'exploit prend effet est un **point d'impact**. Par exemple:

- Exécution arbitraire (obtenir un shell);
- Élévation de privilèges;
- Usurpation d'identité (deviner un mot de passe faible);
- Rebond vers une autre attaque (phishing, puis usurpation);

3.7 La surface d'impact

L'endroit du code où l'exploit prend effet est un **point d'impact**. Par exemple:

- Exécution arbitraire (obtenir un shell);
- Élévation de privilèges;
- Usurpation d'identité (deviner un mot de passe faible);
- Rebond vers une autre attaque (phishing, puis usurpation);
- Fuite d'informations confidentielles.

3.8 Qui sont les attaquant.e.s ?

- Script kiddie qui s'amuse avec un tuto YouTube, APT, ...
- Motivations variées: argent, politique, défi technique.

3.8 Qui sont les attaquant.e.s ?



Exercices

Pour le scénario suivant, identifiez la surface d'attaque, le chemin d'attaque, les vulnérabilités et la surface d'impact :

Un site de e-commerce affiche dans une page “vu récemment” le nom des produits visités, stocké dans un cookie modifiable par l'utilisateur.rice et affiché sans échappement HTML.

1. Définir la sécurité
2. Modéliser les risques
3. The Attacker Mindset
- 4. En résumé**

4. En résumé

La difficulté dans la sécurisation d'un système d'information est qu'il faut que nos objectifs soient respectés, peu importe ce qu'entreprennent les attaquants.

4. En résumé

Il est par exemple très facile de garantir qu'une personne ait accès à un système: il suffit de lui demander.

En revanche, il est beaucoup plus complexe de garantir que cette personne uniquement puisse accéder au système.

4. En résumé

La sécurité est un processus **itératif**. A chaque itération, on essaie d'identifier le lien le plus fragile du système et de le renforcer.

Cela peut se faire par la modification du threat model, par la mise à jour des mécanismes (patcher un bug...) etc.

4. En résumé

Il faut noter qu'il est généralement **beaucoup plus complexe de défendre un système** que de l'attaquer:

- l'attaquant.e ne doit réussir qu'une **seule fois**;
- lea défenseur.euse doit réussir à **chaque coup**.

4. En résumé

Aucun système n'est sécurisé à 100 pour 100 !

4. En résumé

Une des approches pour sécuriser son SI, en tant que défenseur.euse, est de faire en sorte que le coût de l'attaque soit supérieur à la valeur de ce qu'il y a sur le système.

4. En résumé

Globalement, il est bon de garder en tête que:

- Si l'attaquant.e obtient un accès physique, c'est généralement *game over*.
- Sur le long terme, le chiffrement ne fait que rajouter de la latence vers une fin inévitable: le déchiffrement.
- Les *malwares* sont de partout, et sont bien plus évolués que les logiciels anti-virus.
- La porte d'entrée n'est pas que logiciel, elle peut être matérielle ou humaine.

© ⓘ ⓘ Hugo Blanc, 2026

Ce document peut être distribué librement, selon les termes de la version 4.0 de la licence Creative Commons Attribution-ShareAlike: <http://creativecommons.org/licenses/by-sa/4.0/>.

Vous êtes libres de reproduire, distribuer et communiquer ce document au public et de modifier ce document, selon les conditions suivantes :

- Paternité. Vous devez citer le nom de l'auteur original.
- Partage des Conditions Initiales à l'Identique. Si vous modifiez, transformez ou adaptez cette création, vous n'avez le droit de distribuer la création qui en résulte que sous un contrat identique à celui-ci.
- À chaque réutilisation ou distribution, vous devez faire apparaître clairement aux autres les conditions contractuelles de mise à disposition de cette création.

Chacune de ces conditions peut être levée si vous obtenez l'autorisation du titulaire des droits.