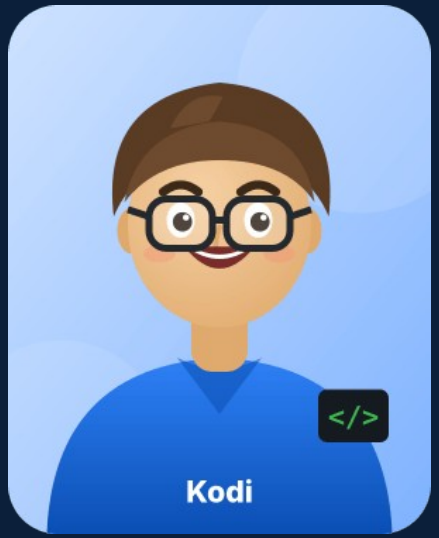
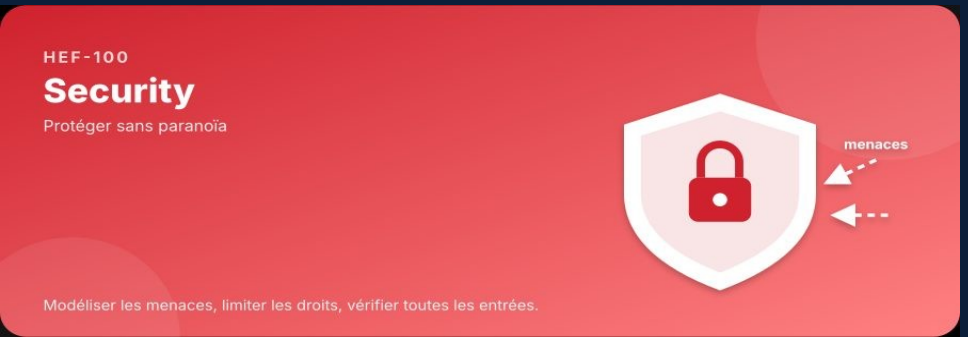
HEF-100 · MISSION 01 / 10

Principes de sécurité et surface d'attaque

La sécurité n'est pas un produit qu'on ajoute à la fin : c'est une façon de concevoir, avec quelques principes simples.



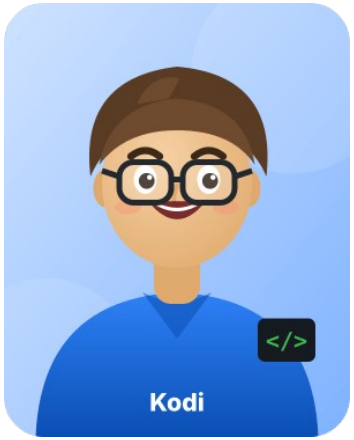


Badge : Gardien des principes

80 min · intermédiaire

L'histoire

Un matin, Maya reçoit un message : un chercheur en sécurité a trouvé sur le serveur de test de HashBox une page d'administration accessible depuis internet, protégée par le mot de passe `admin / admin`. Rien n'a été volé, mais l'équipe est sonnée.



> *« Personne n'a voulu mal faire, dit Sam. Mais personne n'a non plus demandé : qu'est-ce qu'on expose, à qui, et que se passe-t-il si ça tombe entre de mauvaises mains ? Voilà la sécurité : ces trois questions, posées tôt. »*

Dans cette première mission, tu apprends le vocabulaire, les principes fondateurs et la notion de surface d'attaque.

L'histoire

L'histoire (suite)

⚖ ***Cadre éthique de tout le module*** *: on apprend à* ***défendre****. Les techniques sont démontrées sur du code que tu écris toi-même ou sur des exemples fictifs. N'attaque jamais un système sans* ***autorisation écrite*** *de son propriétaire : c'est illégal, même « pour tester ».*

Ta mission

À la fin, je serai capable de…

1. définir confidentialité, intégrité et disponibilité (CIA)
2. distinguer actif, menace, vulnérabilité, risque et contrôle
3. énoncer les principes de moindre privilège, défense en profondeur et sécurité par défaut
4. identifier la surface d'attaque d'une application
5. calculer l'effet de plusieurs couches de défense

Défi d'échauffement

Défi d'échauffement

Tu possèdes une maison. Liste **tout ce qu'un cambrioleur pourrait utiliser** pour entrer (portes, fenêtres, garage, double de clé chez le voisin, wifi des caméras…). Puis classe-les : lesquelles protèges-tu le mieux ? Laquelle est la plus faible ? Si tu ne pouvais renforcer que **deux** points, lesquels choisirais-tu, et pourquoi ?

Le concept

Le vocabulaire

Terme	Définition	Exemple
Actif (*asset*)	ce qui a de la valeur	base d'utilisateurs, clé de signature
Menace	cause potentielle d'un dommage	un attaquant externe, un employé négligent
Vulnérabilité	faiblesse exploitable	mot de passe par défaut, champ non validé
Attaque / exploit	exploitation d'une vulnérabilité	injection SQL
Risque	probabilité × impact d'un dommage	fuite de 100 000 e-mails
Contrôle (*mesure*)	ce qui réduit le risque	authentification forte, chiffrement, journalisation

Le concept

La triade CIA

Propriété	Question	Atteinte typique
Confidentialité	seuls les autorisés lisent-ils ?	fuite de données
Intégrité	les données sont-elles exactes et non modifiées ?	altération d'une commande
Disponibilité	le service répond-il quand on en a besoin ?	déni de service, rançongiciel

On y ajoute souvent : **authenticité** (qui est vraiment l'auteur ?), **non-répudiation** (peut-on nier avoir fait ?), **traçabilité**.

Les principes fondateurs

Le concept

Les principes fondateurs (suite)

Principe	Idée	Exemple
Moindre privilège	chaque personne ou programme n'a que les droits nécessaires	le service web n'est pas administrateur de la base
Défense en profondeur	plusieurs couches indépendantes de protection	pare-feu + authentification + validation + chiffrement + journaux
Sécurité par défaut	la configuration livrée est sûre	pas de mot de passe par défaut, ports fermés
Échec sûr (*fail-safe*)	en cas d'erreur, on refuse	si le contrôle d'accès plante, on n'autorise pas
Séparation des tâches	une action critique demande deux personnes ou deux étapes	validation d'un déploiement
Simplicité	moins de code, moins de fonctions, moins de failles	désactiver ce qui ne sert pas

Le concept

Les principes fondateurs (suite)

Principe	Idée	Exemple
Ne pas faire confiance aux entrées	tout ce qui vient de l'extérieur est suspect	valider, encoder, limiter
Pas de sécurité par l'obscurité	la sécurité ne repose pas sur le secret du fonctionnement (principe de Kerckhoffs)	l'algorithme est public, seule la clé est secrète

La surface d'attaque

La **surface d'attaque** est l'ensemble des points par lesquels un attaquant peut tenter d'agir : ports ouverts, routes d'API, formulaires, comptes, dépendances, fichiers de configuration, employés (hameçonnage), fournisseurs.

Le concept

La surface d'attaque (suite)

> ***Réduire la surface d'attaque*** *est l'une des mesures les plus rentables : ce qui n'existe pas ne peut pas être attaqué.*

Pour la cartographier : lister **les entrées** (réseau, fichiers, interfaces, personnes), **les actifs** qu'elles atteignent, et **les protections** qui séparent les deux.

La défense en profondeur, en chiffres

Si chaque couche indépendante arrête une attaque avec la probabilité p, la probabilité qu'une attaque **traverse** n couches est $(1 - p)^n$.

```
def probabilite_de_passage(efficacites):
    p = 1.0
    for e in efficacites:
        p *= (1 - e)
```

Le concept

La défense en profondeur, en chiffres (suite)

```
    return p

print(probabilite_de_passage([0.9]))                # une seule couche à 90 % : 0
   .1
print(probabilite_de_passage([0.7, 0.7, 0.7]))      # trois couches à 70 % : 0.02
   7
```

Trois couches moyennes valent mieux qu'une couche « très bonne » — **si elles sont réellement indépendantes**. Si elles reposent toutes sur le même secret ou la même hypothèse, elles tombent ensemble.

Les acteurs de menace

Acteur	Motivation	Moyens
Opportuniste / automatisé	gain facile	scans de masse, mots de passe par défaut
Criminel organisé	argent (rançongiciel, fraude)	campagnes ciblées, ingénierie sociale

Le concept

Les acteurs de menace (suite)

Acteur	Motivation	Moyens
Initié (employé, prestataire)	rancune, erreur, appât du gain	accès légitime
Étatique	espionnage, sabotage	ressources importantes
Hacktiviste	cause	déni de service, défiguration

On ne défend pas contre « tout le monde » : on calibre les contrôles selon **qui** peut s'intéresser à **quoi**.

La sécurité est un compromis

Chaque contrôle a un coût (argent, temps, ergonomie). Une sécurité inutilisable est contournée. Le but est un **risque acceptable**, pas un risque nul.

Le concept

La sécurité est un compromis (suite)

🧠 ***Astuce*** *: devant toute fonctionnalité, pose trois questions : Qu'est-ce qu'on expose ? À qui ? Que se passe-t-il si c'est détourné ?*

Exemple pas à pas

Problème : cartographier la surface d'attaque de HashBox et estimer l'effet des protections.

```
# Inventaire des points d'entrée de l'application (fictive)
POINTS_ENTREE = [
    {"nom": "page de connexion", "public": True,  "authentifie": False, "actif": "
   comptes",   "protections": ["limitation de débit", "journalisation"]},
    {"nom": "API des cours",     "public": True,  "authentifie": True,  "actif": "
   contenu",   "protections": ["jeton", "validation"]},
    {"nom": "page /admin",       "public": True,  "authentifie": False, "actif": "
   tout",      "protections": []},
    {"nom": "base de données",   "public": False, "authentifie": True,  "actif": "
   données",   "protections": ["réseau privé", "compte dédié"]},
    {"nom": "console SSH",       "public": True,  "authentifie": True,  "actif": "
   serveur",   "protections": ["clé", "liste d'IP"]},
    {"nom": "import CSV",        "public": False, "authentifie": True,  "actif": "
   données",   "protections": []},
]
```

Exemple pas à pas

Exemple pas à pas (suite)

```
def surface(points):
    exposes = [p for p in points if p["public"]]
    sans_auth = [p for p in exposes if not p["authentifie"]]
    nus = [p for p in exposes if not p["protections"]]
    return {
        "points": len(points),
        "exposés à internet": len(exposes),
        "exposés sans authentification": [p["nom"] for p in sans_auth],
        "exposés sans aucune protection": [p["nom"] for p in nus],
    }

for cle, valeur in surface(POINTS_ENTREE).items():
    print(f"{cle:32s} {valeur}")
```

Lecture : le rapport met en évidence `page /admin` : **exposée à internet, sans authentification, sans protection**. C'est le point à fermer en premier : le retirer d'internet réduit la surface d'attaque plus que n'importe quel renforcement ailleurs.

Exemple pas à pas (suite)

Effet d'une défense en profondeur :

```
def probabilite_de_passage(efficacites):
    p = 1.0
    for e in efficacites:
        p *= (1 - e)
    return p

scenarios = {
    "aucune protection": [],
    "pare-feu seul (80 %)": [0.8],
    "pare-feu + authentification (80 %, 90 %)": [0.8, 0.9],
    "+ validation + journaux/blocage (80 %, 90 %, 70 %, 50 %)": [0.8, 0.9, 0.7, 0.
   5],
}
for nom, couches in scenarios.items():
    print(f"{nom:58s} attaque réussie : {100 * probabilite_de_passage(couches):6.2
   f} %")
```

Exemple pas à pas (suite)

Résultat : de 100 % sans protection à environ 0,3 % avec quatre couches. Mais ce calcul suppose des couches **indépendantes** : si la même faille de configuration les contourne toutes, l'avantage disparaît.
Une règle de mots de passe, mesurée :

```
import math

def entropie_bits(mot_de_passe):
    alphabet = 0
    if any(c.islower() for c in mot_de_passe): alphabet += 26
    if any(c.isupper() for c in mot_de_passe): alphabet += 26
    if any(c.isdigit() for c in mot_de_passe): alphabet += 10
    if any(not c.isalnum() for c in mot_de_passe): alphabet += 32
    return len(mot_de_passe) * math.log2(alphabet) if alphabet else 0

for mdp in ("admin", "Motdepasse1", "correct-cheval-pile-agrafe"):
    print(f"{mdp:30s} ≈ {entropie_bits(mdp):5.1f} bits")
```

Exemple pas à pas (suite)

C'est une **estimation optimiste** (un mot courant comme « Motdepasse1 » se devine bien plus vite qu'un tirage aléatoire), mais elle montre qu'**allonger** un mot de passe aide plus que de multiplier les symboles.

À toi, avec aide

Une petite association utilise : un site WordPress public, un tableur de membres sur un partage cloud accessible à « toute personne disposant du lien », une adresse e-mail partagée par 5 bénévoles avec un mot de passe unique, et un ordinateur de bureau avec des fichiers non chiffrés.

1. Liste les **actifs**.
2. Liste les **points d'entrée** (surface d'attaque).
3. Pour chacun, nomme la propriété CIA la plus menacée.
4. Propose les **trois premières mesures** (par ordre de rentabilité).

Indices progressifs

Indice 1

Pense aux actifs de valeur : liste des membres (données personnelles), accès au site, adresse e-mail.

Indice 2

Le lien « toute personne disposant du lien » est un point d'entrée public.

Indice 3

Les mesures les moins chères et les plus efficaces : supprimer l'accès public, mots de passe individuels + gestionnaire, mises à jour.

À toi, avec aide · réponse

Réponse

- **Actifs** : données personnelles des membres, contenu et accès du site, boîte e-mail (réinitialisation de mots de passe), ordinateur de bureau.
- **Surface d'attaque** : site WordPress (plugins, thèmes, comptes), lien partagé du tableur, e-mail partagé (hameçonnage, mot de passe réutilisé), poste de bureau (accès physique, logiciels non à jour).
- **Propriétés menacées** : tableur → **confidentialité** (quiconque a le lien lit) ; site → **intégrité** (défiguration) et **disponibilité** ; e-mail partagé → **authenticité/traçabilité** (on ne sait pas qui a fait quoi) ; poste → **confidentialité** et **disponibilité** (vol, panne, rançongiciel).
- **Trois premières mesures** : (1) restreindre le tableur à des personnes nommées avec authentification ; (2) comptes individuels avec authentification à deux facteurs sur l'e-mail et le site ; (3) mises à jour automatiques + sauvegardes chiffrées hors ligne.

À toi, en solo

Ton outil d'inventaire. Écris `rapport_surface(points)` qui, à partir d'une liste de points d'entrée (dictionnaires comme dans l'exemple), calcule pour chaque point un **score d'exposition** :

- +3 si public ; +2 si pas d'authentification ; +2 si aucune protection ; +1 si l'actif est « tout » ou « données » ; −1 par protection (minimum 0) ;

puis classe les points du plus au moins exposé et affiche les trois premiers avec une phrase de recommandation générée (par exemple « retirer de l'accès public » si public et sans authentification).

Teste avec au moins 8 points d'entrée inventés, dont un cas évident à fermer immédiatement.

À toi, en solo · réponse

Pistes de correction

```
def score(p):
    s = 0
    if p["public"]: s += 3
    if not p["authentifie"]: s += 2
    if not p["protections"]: s += 2
    if p["actif"] in ("tout", "données"): s += 1
    s -= len(p["protections"])
    return max(s, 0)

def recommandation(p):
    if p["public"] and not p["authentifie"]:
        return "retirer de l'accès public ou exiger une authentification"
    if not p["protections"]:
        return "ajouter au moins une protection (limitation de débit, validation,
   journalisation)"
    if p["public"]:
        return "vérifier que les protections sont efficaces et testées"
    return "surveiller"
```

À toi, en solo · réponse

Pistes de correction (suite)

```
def rapport_surface(points, n=3):
    classes = sorted(points, key=score, reverse=True)
    return [(p["nom"], score(p), recommandation(p)) for p in classes[:n]]

POINTS = [
    {"nom": "page /admin", "public": True, "authentifie": False, "actif": "tout",
   "protections": []},
    {"nom": "API publique", "public": True, "authentifie": True, "actif": "contenu
   ", "protections": ["jeton", "limitation de débit"]},
    {"nom": "base de données", "public": False, "authentifie": True, "actif": "don
   nées", "protections": ["réseau privé"]},
    {"nom": "formulaire de contact", "public": True, "authentifie": False, "actif"
   : "contenu", "protections": ["captcha"]},
    {"nom": "import CSV", "public": False, "authentifie": True, "actif": "données"
   , "protections": []},
    {"nom": "SSH", "public": True, "authentifie": True, "actif": "tout", "protecti
   ons": ["clé", "liste d'IP"]},
    {"nom": "sauvegardes", "public": False, "authentifie": True, "actif": "données
   ", "protections": ["chiffrement", "accès restreint"]},
```

Pistes de correction (suite)

```
    {"nom": "page de statut", "public": True, "authentifie": False, "actif": "cont
   enu", "protections": ["lecture seule"]},
]
for nom, s, reco in rapport_surface(POINTS):
    print(f"{s:2d}  {nom:22s} {reco}")
assert rapport_surface(POINTS)[0][0] == "page /admin"
```

Le calcul est volontairement simple : un score n'est qu'une aide à la priorisation, pas une vérité.

Pièges classiques

Piège	Pourquoi c'est un problème	Comment l'éviter
Croire que « personne ne trouvera »	La sécurité par l'obscurité tombe vite (scans automatiques)	Supposer que tout ce qui est exposé sera trouvé
Tout protéger de la même façon	Moyens gaspillés, points critiques sous-protégés	Protéger selon la valeur de l'actif et la menace
Une seule couche « très forte »	Si elle tombe, tout tombe	Défense en profondeur avec couches indépendantes
Ajouter la sécurité à la fin	Corrections coûteuses, architecture inadaptée	Penser sécurité dès la conception

Mini-boss

Mini-boss

Revue de surface d'attaque de HashBox. Décris dans un dictionnaire l'architecture de HashBox : site web, API, base de données, stockage de fichiers, serveur de mails, outil d'administration, sauvegardes, pipeline de déploiement, comptes des employés, fournisseurs (paiement, e-mail).

1. Pour chaque élément : exposition (internet, réseau interne, local), authentification, protections, actif, propriétés CIA concernées.
2. Écris le code qui calcule le score d'exposition et une **matrice** éléments × propriétés CIA.
3. Applique le principe de **moindre privilège** : liste pour chaque compte technique (service web, sauvegarde, déploiement) les accès réellement nécessaires et ceux à supprimer.
4. Propose un plan en trois vagues (cette semaine, ce mois, ce trimestre) avec le nombre de points d'entrée supprimés ou protégés à chaque vague.
5. Rédige une page d'explication adressée à la direction : risques, coût, bénéfice, ce qu'on accepte de ne pas traiter et pourquoi.

Mini-boss · réponse

Pistes de correction

Un bon dossier : au moins 12 éléments décrits ; une fonction `matrice_cia(points)` qui marque chaque élément C/I/D selon l'actif ; une table de comptes techniques avec « accès actuels » et « accès nécessaires », par exemple le service web n'a besoin que de `SELECT/INSERT/UPDATE` sur trois tables, pas du compte propriétaire de la base ; un plan en vagues cohérent (fermer l'administration publique et les mots de passe par défaut d'abord ; MFA et gestion des secrets ensuite ; segmentation réseau et tests d'intrusion ensuite) ; une note pour la direction qui parle de risque et de coût (pas de jargon), qui nomme des risques acceptés (par exemple un fournisseur de paiement certifié) et prévoit une revue périodique.

Quiz éclair

Quiz éclair

1. La triade CIA désigne :

A. confidentialité, intégrité, disponibilité
B. contrôle, identité, accès
C. chiffrement, intégrité, authentification

2. La surface d'attaque, c'est :

A. la taille du code
B. l'ensemble des points par lesquels un attaquant peut agir
C. le nombre d'utilisateurs

3. Le principe de moindre privilège dit de :

A. donner tous les droits aux administrateurs

Quiz éclair

Quiz éclair (suite)

B. supprimer les mots de passe
C. n'accorder que les droits nécessaires

4. 🔁 *Révision HEF-090, mission 09* : La récupération d'erreur du parseur consiste à :

A. avancer jusqu'à un point de synchronisation puis reprendre
B. s'arrêter à la première erreur
C. ignorer toutes les erreurs

5. 🔁 *Révision HEF-090, mission 09* : Un bon message d'erreur indique :

A. seulement « erreur »
B. où, quoi, pourquoi et comment corriger
C. un code interne

Quiz éclair · réponse

Réponses

1. **A**.
2. **B**.
3. **C**.
4. **A**.
5. **B**.

Je réfléchis

Je réfléchis

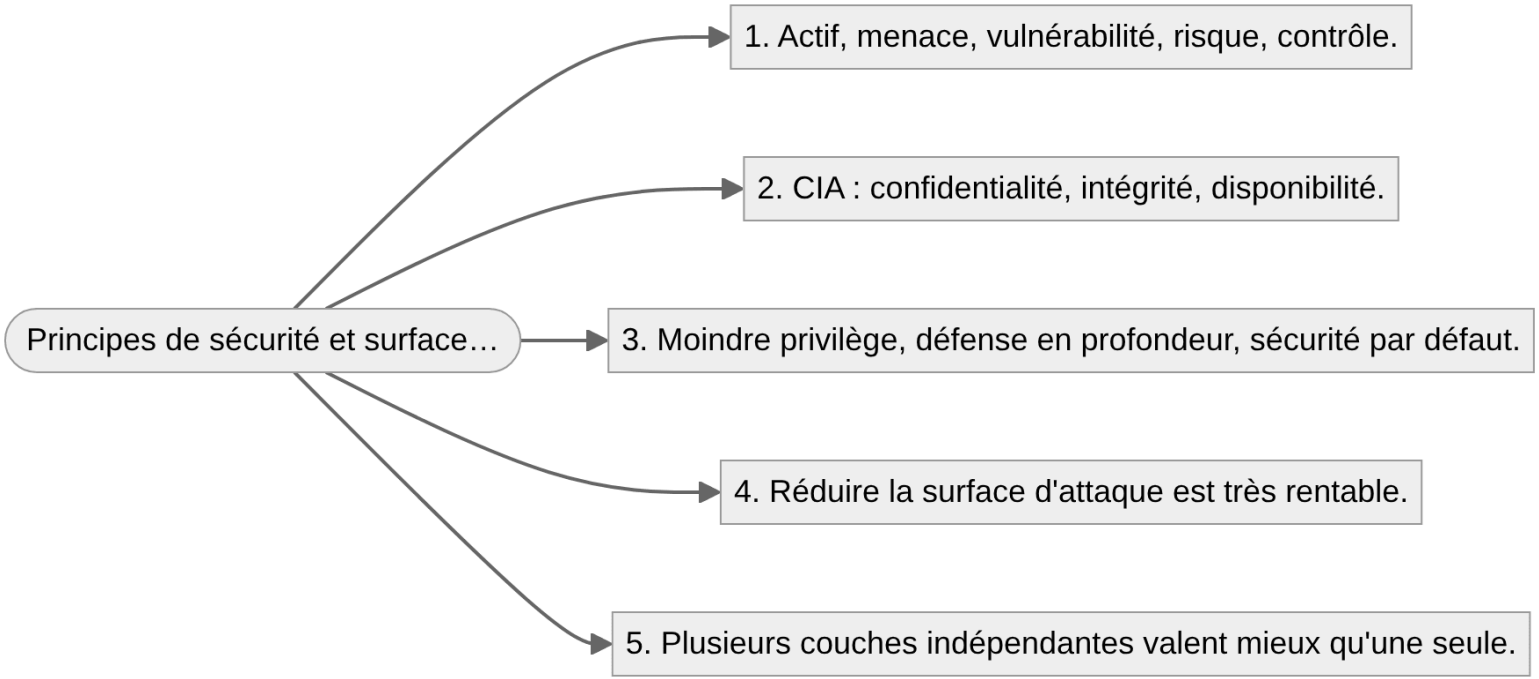
1. Quels éléments de ta vie numérique (comptes, appareils, partages) forment ta propre surface d'attaque ?
2. Quel contrôle simple et peu coûteux ferait le plus baisser ton risque dès aujourd'hui ?

À retenir

1. Actif, menace, vulnérabilité, risque, contrôle.
2. CIA : confidentialité, intégrité, disponibilité.
3. Moindre privilège, défense en profondeur, sécurité par défaut.
4. Réduire la surface d'attaque est très rentable.
5. Plusieurs couches indépendantes valent mieux qu'une seule.

À retenir

Le schéma de la mission



MISSION ACCOMPLIE

+100 XP

Badge débloqué : Gardien des principes

Prochaine mission : La modélisation des menaces

