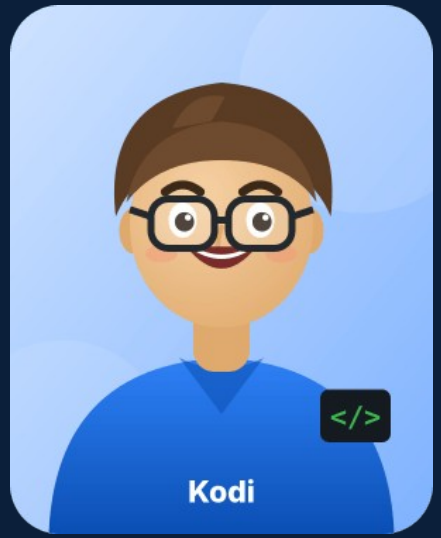
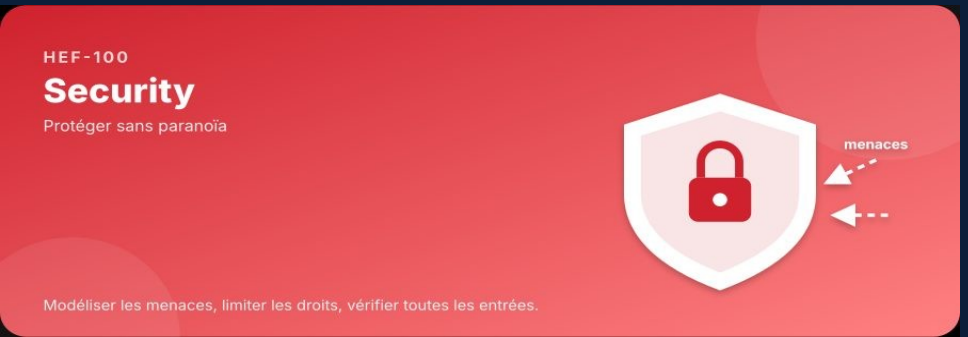
HEF-100 · MISSION 09 / 10

Journalisation, détection et réponse à incident

On ne peut pas défendre ce qu'on ne voit pas. Et quand ça arrive, on a besoin d'un plan, pas d'improvisation.



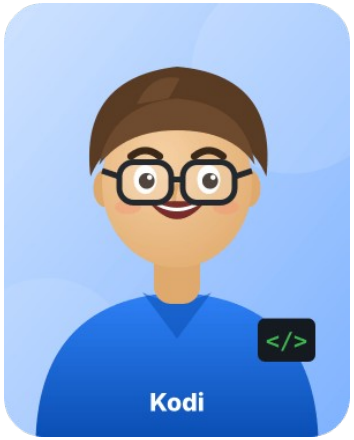


Badge : Veilleur et répondant

100 min · intermédiaire

L'histoire

Un matin, un formateur de HashBox signale que son compte a envoyé des messages étranges pendant la nuit. L'équipe cherche : quand la connexion a-t-elle eu lieu ? Depuis où ? Qu'a fait ce compte ? Les journaux existent, mais ils ne contiennent pas l'adresse IP, ils sont écrasés après 24 heures, et personne ne sait qui est responsable de la réponse.



> *« Une enquête d'incident repose sur trois choses, dit Sam : des traces fiables, des outils pour les lire, et un plan connu à l'avance. Il manque les trois. »*

Tu apprends à journaliser pour la sécurité, à détecter des comportements suspects et à mener une réponse à incident structurée.

Ta mission

À la fin, je serai capable de…

1. définir quoi journaliser pour la sécurité (et quoi ne pas journaliser)
2. analyser des journaux pour détecter force brute, bourrage d'identifiants et voyage impossible
3. protéger l'intégrité des journaux avec une chaîne de hachés
4. décrire les phases de la réponse à incident (préparation, détection, confinement, éradication, récupération, retour d'expérience)
5. mesurer MTTD et MTTR et rédiger un post-mortem sans blâme

Défi d'échauffement

Défi d'échauffement

Un cambriolage a eu lieu dans un magasin. Pour reconstituer ce qui s'est passé, quels éléments voudrais-tu avoir (caméras, journal des alarmes, badge des employés, caisse) ? Que se passe-t-il si les enregistrements sont effacés après 24 h ou si le cambrioleur peut les modifier ? Écris les propriétés indispensables d'un bon enregistrement.

Le concept

Que journaliser pour la sécurité ?

Catégorie	Exemples d'événements
Authentification	succès, échecs, verrouillage, changement de mot de passe, MFA
Autorisation	accès refusés, changements de droits
Données sensibles	lectures, exports, suppressions
Administration	actions privilégiées, changements de configuration
Application	erreurs inattendues, validation d'entrée échouée
Système	démarrage/arrêt de services, nouvelles clés, nouveaux comptes

Le concept

Que journaliser pour la sécurité ? (suite)

Pour chaque événement : **qui** (identifiant, pas le mot de passe), **quoi**, **quand** (horodatage UTC précis), **d'où** (IP, agent), **sur quelle ressource**, **résultat**, **identifiant de corrélation** (suivre une requête à travers les services).

À ne jamais journaliser : mots de passe, jetons, clés, numéros de carte complets, données personnelles inutiles (mission 5).

Des journaux utiles

- **structurés** (JSON) pour être analysés par des machines ;
- **centralisés** (hors de la machine compromise) ;
- **horodatés de façon synchronisée** (NTP) ;
- **conservés** assez longtemps (30 à 365 jours selon les règles) ;
- **protégés** : écriture seule, intégrité vérifiable ;

Le concept

Des journaux utiles (suite)

- **surveillés** : des alertes, pas seulement du stockage.

Détection : quelques motifs classiques

Motif	Signal	Exemple de règle
Force brute	beaucoup d'échecs sur un compte depuis une source	≥ 5 échecs en 60 s
Bourrage d'identifiants (*credential stuffing*) / **pulvérisation**	un mot de passe essayé sur beaucoup de comptes	une IP, ≥ 10 comptes différents en 10 min
Connexion après échecs	succès juste après une rafale d'échecs	compromission probable
Voyage impossible	deux connexions éloignées géographiquement en peu de temps	Lyon puis Sydney 20 min plus tard
Heures inhabituelles	activité hors des horaires normaux du compte	3 h du matin

Le concept

Détection : quelques motifs classiques (suite)

Motif	Signal	Exemple de règle
Élévation de privilèges	un compte reçoit des droits administrateur	événement `role_change`
Exfiltration	volume anormal de lectures ou d'exports	100 fois la moyenne

Intégrité des journaux : la chaîne de hachés

Pour détecter la modification ou la suppression de lignes, on chaîne les enregistrements : chaque ligne inclut le **haché de la précédente**. Modifier une ligne casse toutes les suivantes.

```
import hashlib, json

def ajouter(chaine, evenement):
    precedent = chaine[-1]["hash"] if chaine else "0" * 64
```

Le concept

Intégrité des journaux : la chaîne de hachés (suite)

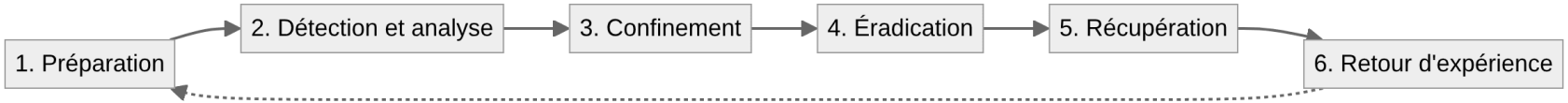
```
    corps = json.dumps(evenement, sort_keys=True)
    h = hashlib.sha256((precedent + corps).encode()).hexdigest()
    chaine.append({"evenement": evenement, "precedent": precedent, "hash": h})

def verifier(chaine):
    precedent = "0" * 64
    for i, ligne in enumerate(chaine):
        attendu = hashlib.sha256((precedent + json.dumps(ligne["evenement"], sort_
   keys=True)).encode()).hexdigest()
        if ligne["precedent"] != precedent or ligne["hash"] != attendu:
            return i
        precedent = ligne["hash"]
    return None
```

(Pour être résistant à un attaquant qui contrôle la machine, on envoie aussi régulièrement le dernier haché à un système externe.)

Le concept

Réponse à incident (modèle NIST)



Description du schéma : six étapes en cycle : préparation, détection et analyse, confinement, éradication, récupération, puis retour d'expérience qui améliore la préparation.

Phase	Objectif	Exemples d'actions
Préparation	être prêt avant	plan, rôles, contacts, journaux, sauvegardes testées, exercices
Détection et analyse	savoir qu'il y a un incident et son ampleur	alertes, triage, chronologie

Le concept

Réponse à incident (modèle NIST) (suite)

Phase	Objectif	Exemples d'actions
Confinement	stopper la propagation	isoler la machine, révoquer des sessions et des clés, bloquer une IP
Éradication	supprimer la cause	corriger la faille, supprimer les accès de l'attaquant
Récupération	revenir en service sûr	restaurer, surveiller de près
Retour d'expérience	apprendre	post-mortem sans blâme, actions correctives

Règles d'or : conserver les **preuves** (copies, hachés) avant de nettoyer ; **communiquer** (interne, clients, autorités selon les obligations légales, par exemple notification d'une violation de données personnelles sous 72 h en Europe) ; **documenter** chaque action avec l'heure.

Le concept

Mesures

Indicateur	Sens
MTTD (*mean time to detect*)	délai moyen entre le début d'un incident et sa détection
MTTR (*mean time to respond/recover*)	délai moyen de réponse ou de rétablissement
Taux de faux positifs	alertes inutiles (fatigue)

Le post-mortem sans blâme

Un bon post-mortem décrit la **chronologie**, les **causes** (techniques et organisationnelles), ce qui a bien marché, ce qui a manqué, et des **actions** avec responsables et dates. Il cherche à comprendre le **système**, pas à désigner un coupable.

Le concept

Le post-mortem sans blâme (suite)

🧠 ***Astuce*** *: écris ton plan de réponse* ***avant*** *d'en avoir besoin, avec les numéros de téléphone, et exerce-le une fois par an.*

Exemple pas à pas

Problème : analyser un journal d'authentification pour détecter force brute, pulvérisation et voyage impossible, et protéger le journal contre la falsification.

```
import hashlib, json, math, re
from collections import defaultdict
from datetime import datetime, timedelta

JOURNAL = '''
2025-03-10T02:14:01Z LOGIN_FAIL user=ada ip=203.0.113.9 geo=FR
2025-03-10T02:14:05Z LOGIN_FAIL user=ada ip=203.0.113.9 geo=FR
2025-03-10T02:14:09Z LOGIN_FAIL user=ada ip=203.0.113.9 geo=FR
2025-03-10T02:14:13Z LOGIN_FAIL user=ada ip=203.0.113.9 geo=FR
2025-03-10T02:14:17Z LOGIN_FAIL user=ada ip=203.0.113.9 geo=FR
2025-03-10T02:14:21Z LOGIN_OK   user=ada ip=203.0.113.9 geo=FR
2025-03-10T09:00:00Z LOGIN_OK   user=bob ip=198.51.100.4 geo=FR
2025-03-10T09:10:00Z LOGIN_OK   user=bob ip=192.0.2.77 geo=AU
2025-03-10T03:00:00Z LOGIN_FAIL user=zoe ip=192.0.2.200 geo=RU
2025-03-10T03:00:03Z LOGIN_FAIL user=eve ip=192.0.2.200 geo=RU
```

Exemple pas à pas (suite)

```
2025-03-10T03:00:06Z LOGIN_FAIL user=sam ip=192.0.2.200 geo=RU
2025-03-10T03:00:09Z LOGIN_FAIL user=cleo ip=192.0.2.200 geo=RU
2025-03-10T03:00:12Z LOGIN_FAIL user=dan ip=192.0.2.200 geo=RU
2025-03-10T03:00:15Z LOGIN_FAIL user=max ip=192.0.2.200 geo=RU
'''

MOTIF = re.compile(r"(\S+) (LOGIN_OK|LOGIN_FAIL)\s+user=(\w+) ip=(\S+) geo=(\w+)")
CENTRES = {"FR": (46.2, 2.2), "AU": (-25.3, 133.8), "RU": (61.5, 105.3)}

def lire(texte):
    evts = []
    for m in MOTIF.finditer(texte):
        evts.append({"t": datetime.strptime(m.group(1), "%Y-%m-%dT%H:%M:%SZ"), "ty
   pe": m.group(2), "user": m.group(3), "ip": m.group(4), "geo": m.group(5)})
    return sorted(evts, key=lambda e: e["t"])

def distance_km(a, b):
    (la1, lo1), (la2, lo2) = map(lambda x: tuple(map(math.radians, x)), (a, b))
    h = math.sin((la2 - la1) / 2) ** 2 + math.cos(la1) * math.cos(la2) * math.sin(
```

Exemple pas à pas (suite)

```
    (lo2 - lo1) / 2) ** 2
     return 6371 * 2 * math.asin(math.sqrt(h))

def detecter(evts):
     alertes = []
     echecs = defaultdict(list)
     for e in evts:
         if e["type"] == "LOGIN_FAIL":
             echecs[(e["user"], e["ip"])].append(e["t"])
         elif e["type"] == "LOGIN_OK":
             recents = [t for t in echecs[(e["user"], e["ip"])] if e["t"] - t <= ti
    medelta(seconds=60)]
             if len(recents) >= 5:
                 alertes.append(("critique", f"connexion réussie de {e['user']} apr
    ès {len(recents)} échecs en 60 s depuis {e['ip']} : compromission probable"))
     par_ip = defaultdict(dict)
     for e in evts:
         if e["type"] == "LOGIN_FAIL":
             par_ip[e["ip"]][e["user"]] = e["t"]
```

Exemple pas à pas

Exemple pas à pas (suite)

```
    for ip, comptes in par_ip.items():
        temps = sorted(comptes.values())
        if len(comptes) >= 5 and temps[-1] - temps[0] <= timedelta(minutes=10):
            alertes.append(("haute", f"pulvérisation de mots de passe : {len(compt
  es)} comptes visés depuis {ip}"))
    derniere = {}
    for e in evts:
        if e["type"] == "LOGIN_OK":
            if e["user"] in derniere:
                avant = derniere[e["user"]]
                km, heures = distance_km(CENTRES[avant["geo"]], CENTRES[e["geo"]])
  , (e["t"] - avant["t"]).total_seconds() / 3600
                if heures > 0 and km / heures > 900:
                    alertes.append(("haute", f"voyage impossible pour {e['user']}
  : {km:.0f} km en {heures * 60:.0f} min"))
            derniere[e["user"]] = e
    return alertes

for niveau, message in detecter(lire(JOURNAL)):
```

Exemple pas à pas (suite)

```
    print(f"[{niveau}] {message}")

# ---------- intégrité des journaux ----------
def ajouter(chaine, evenement):
    precedent = chaine[-1]["hash"] if chaine else "0" * 64
    h = hashlib.sha256((precedent + json.dumps(evenement, sort_keys=True)).encode(
   )).hexdigest()
    chaine.append({"evenement": evenement, "precedent": precedent, "hash": h})

def verifier(chaine):
    precedent = "0" * 64
    for i, ligne in enumerate(chaine):
        attendu = hashlib.sha256((precedent + json.dumps(ligne["evenement"], sort_
   keys=True)).encode()).hexdigest()
        if ligne["precedent"] != precedent or ligne["hash"] != attendu:
            return i
        precedent = ligne["hash"]
    return None
```

Exemple pas à pas (suite)

```
chaine = []
for i in range(5):
    ajouter(chaine, {"n": i, "action": "export", "user": "ada"})
assert verifier(chaine) is None
chaine[2]["evenement"]["user"] = "bob"                    # falsification
assert verifier(chaine) == 2                               # détectée à la lign
   e 2
del chaine[3]                                               # suppression d'une
    ligne
assert verifier(chaine) is not None
print("journal falsifié détecté")
```

Lecture : trois alertes sont produites automatiquement : la compromission probable du compte de `ada` (connexion réussie après 5 échecs en moins d'une minute), la pulvérisation depuis `192.0.2.200` (6 comptes en 15 secondes), et le voyage impossible de `bob` (France puis Australie en 10 minutes). La chaîne de hachés détecte une ligne modifiée ou supprimée.

À toi, avec aide

Calcule les indicateurs de réponse d'une équipe à partir de 5 incidents fictifs :

Incident	Début réel	Détecté à	Confiné à	Rétabli à
1	08:00	08:30	09:00	12:00
2	22:00	06:00 (lendemain)	06:30	10:00
3	14:00	14:05	14:20	15:00
4	03:00	15:00	15:30	18:00
5	10:00	10:10	10:40	11:30

Calcule le MTTD (début → détection), le MTTC (détection → confinement), le MTTR (détection → rétablissement), et ressors la cause principale du mauvais résultat global.

À toi, avec aide · réponse

Indices progressifs

Indice 1

Utilise `datetime` ou convertis les heures en minutes depuis minuit en gérant le lendemain.

Indice 2

L'incident 2 et l'incident 4 ont un très long délai de détection (de nuit).

Indice 3

Moyenne et médiane peuvent diverger fortement : donne les deux.

À toi, avec aide · réponse

Réponse

```
from datetime import datetime
import statistics

def t(h, jour=0):
    hh, mm = map(int, h.split(":"))
    return jour * 1440 + hh * 60 + mm

INCIDENTS = [
    (t("08:00"), t("08:30"), t("09:00"), t("12:00")),
    (t("22:00"), t("06:00", 1), t("06:30", 1), t("10:00", 1)),
    (t("14:00"), t("14:05"), t("14:20"), t("15:00")),
    (t("03:00"), t("15:00"), t("15:30"), t("18:00")),
    (t("10:00"), t("10:10"), t("10:40"), t("11:30")),
]
mttd = [d - s for s, d, c, r in INCIDENTS]
mttc = [c - d for s, d, c, r in INCIDENTS]
mttr = [r - d for s, d, c, r in INCIDENTS]
for nom, v in (("MTTD", mttd), ("MTTC", mttc), ("MTTR", mttr)):
    print(f"{nom}: moyenne {statistics.mean(v):6.1f} min, médiane {statistics.medi
```

À toi, avec aide · réponse

Réponse (suite)

```
    an(v):6.1f} min, max {max(v)} min")
```

Les incidents 2 et 4 (nuit) ont des délais de détection de 480 et 720 minutes : ils dominent la moyenne (MTTD moyen ≈ 249 min contre une médiane de 30). Une fois détectés, le confinement est rapide (≈ 27 min en moyenne). **La cause principale est l'absence de surveillance/alertes la nuit**, pas la capacité de réponse : améliorer la détection (alertes, astreinte) a l'effet le plus fort.

À toi, en solo

Moteur de règles de détection. Écris un petit moteur dans lequel chaque règle est une fonction qui reçoit la liste des événements (dictionnaires) et renvoie des alertes. Implémente au moins 6 règles :

1. force brute suivie d'un succès ;
2. pulvérisation de mots de passe (une IP, plusieurs comptes) ;
3. voyage impossible ;
4. connexion à une heure inhabituelle pour ce compte (apprise sur l'historique : plage où le compte se connecte 95 % du temps) ;
5. élévation de privilèges (événement `ROLE_CHANGE` vers admin sans ticket associé) ;
6. export massif (nombre de lignes exportées > moyenne + 3 écarts-types du compte).

Génère un jeu de 2 000 événements normaux (graine fixée) dans lequel tu **injectes** 5 scénarios d'attaque, puis mesure rappel (attaques trouvées) et précision (alertes réellement justifiées).

Pistes de correction

```
import random, statistics
from collections import defaultdict
from datetime import datetime, timedelta

def generer(graine=1):
    rng = random.Random(graine)
    debut = datetime(2025, 3, 1, 8, 0, 0)
    evts = []
    for i in range(2000):
        user = f"u{rng.randint(1, 40)}"
        t = debut + timedelta(days=rng.randint(0, 9), hours=rng.randint(0, 9), min
   utes=rng.randint(0, 59))   # horaires de bureau 8h-17h
        evts.append({"t": t, "type": rng.choices(["LOGIN_OK", "LOGIN_FAIL", "EXPOR
   T"], [80, 5, 15])[0], "user": user,
                     "ip": f"10.0.{rng.randint(0, 5)}.{rng.randint(1, 50)}", "geo"
   : "FR", "lignes": rng.randint(10, 100)})
    return evts, debut

def injecter(evts, debut):
```

Pistes de correction (suite)

```
    attaques = set()
    t0 = debut + timedelta(days=3, hours=14)
    for k in range(6):
        evts.append({"t": t0 + timedelta(seconds=5 * k), "type": "LOGIN_FAIL", "us
er": "u1", "ip": "203.0.113.9", "geo": "FR", "lignes": 0})
    evts.append({"t": t0 + timedelta(seconds=40), "type": "LOGIN_OK", "user": "u1"
, "ip": "203.0.113.9", "geo": "FR", "lignes": 0})
    attaques.add("force_brute")
    for k in range(8):
        evts.append({"t": t0 + timedelta(minutes=30, seconds=3 * k), "type": "LOGI
N_FAIL", "user": f"u{10 + k}", "ip": "198.51.100.66", "geo": "FR", "lignes": 0}
)
    attaques.add("pulverisation")
    evts.append({"t": debut + timedelta(days=4, hours=-5), "type": "LOGIN_OK", "us
er": "u2", "ip": "10.0.1.1", "geo": "FR", "lignes": 0})
    attaques.add("heure_inhabituelle")
    evts.append({"t": debut + timedelta(days=5, hours=3), "type": "ROLE_CHANGE", "
user": "u3", "ip": "10.0.2.2", "geo": "FR", "lignes": 0, "vers": "admin", "tick
et": None})
```

Pistes de correction (suite)

```
    attaques.add("elevation")
    evts.append({"t": debut + timedelta(days=6, hours=2), "type": "EXPORT", "user"
   : "u4", "ip": "10.0.3.3", "geo": "FR", "lignes": 50000})
    attaques.add("export_massif")
    return sorted(evts, key=lambda e: e["t"]), attaques

def regle_force_brute(evts):
    res, echecs = [], defaultdict(list)
    for e in evts:
        k = (e["user"], e["ip"])
        if e["type"] == "LOGIN_FAIL":
            echecs[k].append(e["t"])
        elif e["type"] == "LOGIN_OK" and sum(1 for t in echecs[k] if e["t"] - t <=
    timedelta(seconds=60)) >= 5:
            res.append(("force_brute", e["user"]))
    return res

def regle_pulverisation(evts):
    par_ip = defaultdict(dict)
```

Pistes de correction (suite)

```
    for e in evts:
        if e["type"] == "LOGIN_FAIL":
            par_ip[e["ip"]][e["user"]] = e["t"]
    return [("pulverisation", ip) for ip, c in par_ip.items() if len(c) >= 5 and m
   ax(c.values()) - min(c.values()) <= timedelta(minutes=10)]

def regle_heure(evts):
    heures = defaultdict(list)
    for e in evts:
        if e["type"] == "LOGIN_OK":
            heures[e["user"]].append(e["t"].hour)
    res = []
    for user, hs in heures.items():
        if len(hs) < 10:
            continue
        for h in set(hs):
            if hs.count(h) == 1 and (h < 7 or h > 19):
                res.append(("heure_inhabituelle", user))
    return res
```

Pistes de correction (suite)

```
def regle_elevation(evts):
    return [("elevation", e["user"]) for e in evts if e["type"] == "ROLE_CHANGE" a
   nd e.get("vers") == "admin" and not e.get("ticket")]

def regle_export(evts):
    par_user = defaultdict(list)
    for e in evts:
        if e["type"] == "EXPORT":
            par_user[e["user"]].append(e["lignes"])
    res = []
    for user, valeurs in par_user.items():
        if len(valeurs) < 5:
            continue
        pic = max(valeurs)
        reste = list(valeurs)
        reste.remove(pic)
        m, s = statistics.mean(reste), statistics.pstdev(reste) or 1
        if pic > m + 3 * s:
```

Pistes de correction (suite)

```
            res.append(("export_massif", user))
    return res

evts, debut = generer()
evts, attaques = injecter(evts, debut)
alertes = set()
for regle in (regle_force_brute, regle_pulverisation, regle_heure, regle_elevation
   , regle_export):
    alertes |= set(regle(evts))
trouvees = {nom for nom, _ in alertes}
rappel = len(trouvees & attaques) / len(attaques)
precision = len([a for a in alertes if a[0] in attaques]) / len(alertes)
print(f"rappel {rappel:.0%}, précision {precision:.0%}, alertes {len(alertes)}")
assert rappel == 1.0
```

Rappel et précision sont les deux faces d'un moteur de détection : un rappel élevé avec une précision faible noie l'équipe sous les fausses alertes ; l'inverse laisse passer des attaques. On règle les seuils en mesurant les deux.

Pièges classiques

Piège	Pourquoi c'est un problème	Comment l'éviter
Journaux sans IP ni identifiant	Impossible d'enquêter	Qui, quoi, quand, d'où, résultat
Journaliser des secrets	Les journaux deviennent un trésor pour l'attaquant	Masquer avant d'écrire
Alertes trop nombreuses	La fatigue d'alerte les fait ignorer	Régler seuils, mesurer faux positifs
Nettoyer avant de collecter les preuves	Enquête et obligations légales compromises	Copier, hacher, puis éradiquer

Mini-boss

Exercice de réponse à incident : « le compte du formateur ». Simule un incident de bout en bout :

1. génère un journal de 3 jours avec activité normale et un scénario d'attaque en 5 étapes (hameçonnage → connexion depuis une IP étrangère → élévation de privilèges → export massif → suppression de journaux) ;
2. écris l'**analyseur** qui reconstruit la **chronologie** de l'incident (événements corrélés par compte et IP) ;
3. utilise la **chaîne de hachés** pour montrer que des lignes ont été supprimées ;
4. écris le **plan de réponse** exécutable : confinement (révoquer les sessions, bloquer l'IP, désactiver le compte), éradication (réinitialiser les secrets), récupération, avec un journal horodaté des actions ;
5. calcule MTTD/MTTR de ta simulation ;
6. rédige un **post-mortem sans blâme** d'une page : chronologie, causes, ce qui a marché, ce qui a manqué, 5 actions avec responsable et échéance.

Mini-boss · réponse

Pistes de correction

Un bon travail : une classe `Incident` qui accumule `(horodatage, acteur, action)`; la chronologie est obtenue en filtrant les événements du compte et de l'IP suspectés et en les triant; la détection de suppression utilise `verifier()` qui renvoie l'indice de la première rupture dans la chaîne; le plan de réponse est une liste d'actions `Action(nom, fonction, preuve)` exécutées dans l'ordre (conserver les preuves → confiner → éradiquer → récupérer) avec journalisation; le post-mortem sépare faits et interprétations, cite des causes organisationnelles (aucune alerte de nuit, MFA absent sur les comptes de formateurs) et propose des actions mesurables (MFA obligatoire sous 30 jours, journaux envoyés vers un stockage externe en écriture seule, exercice semestriel).

Quiz éclair

Quiz éclair

1. Un journal de sécurité utile contient :

A. uniquement les erreurs du programme
B. qui, quoi, quand, d'où et le résultat
C. les mots de passe des utilisateurs

2. Une chaîne de hachés sert à :

A. chiffrer les journaux
B. compresser les journaux
C. détecter la modification ou la suppression de lignes d'un journal

3. La phase de confinement vise à :

A. stopper la propagation de l'incident

Quiz éclair

Quiz éclair (suite)

B. restaurer les sauvegardes
C. rédiger le rapport

4. 🔁 *Révision mission 08* : Le durcissement d'un système consiste à :

A. ajouter des fonctions
B. réduire la surface d'attaque et sécuriser la configuration
C. ralentir le serveur

5. 🔁 *Révision mission 08* : Dans un conteneur, on évite :

A. d'utiliser une image minimale
B. d'épingler les versions
C. d'exécuter le processus en root

Quiz éclair · réponse

Réponses

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

Je réfléchis

1. Si ton compte principal était compromis cette nuit, comment le saurais-tu, et qui préviendrais-tu ?
2. Pourquoi un post-mortem « sans blâme » produit-il de meilleures améliorations qu'une recherche de coupable ?

À retenir

À retenir

1. Journaliser qui, quoi, quand, d'où, résultat — sans secrets.
2. Détecter : force brute, pulvérisation, voyage impossible, exfiltration.
3. Chaîne de hachés pour l'intégrité des journaux.
4. Réponse : préparation, détection, confinement, éradication, récupération, retour d'expérience.
5. MTTD, MTTR, post-mortem sans blâme.

MISSION ACCOMPLIE

+100 XP

Badge débloqué : Veilleur et répondant

Prochaine mission : Audit et rapport de sécurité

