#UYBHYS24

Unlock your brain, Harden your system

2024 - Brest

Insecure time-based secret in web applications and Sandwich Attack exploitation

Tom CHAMBARETAUD aka Aethlios

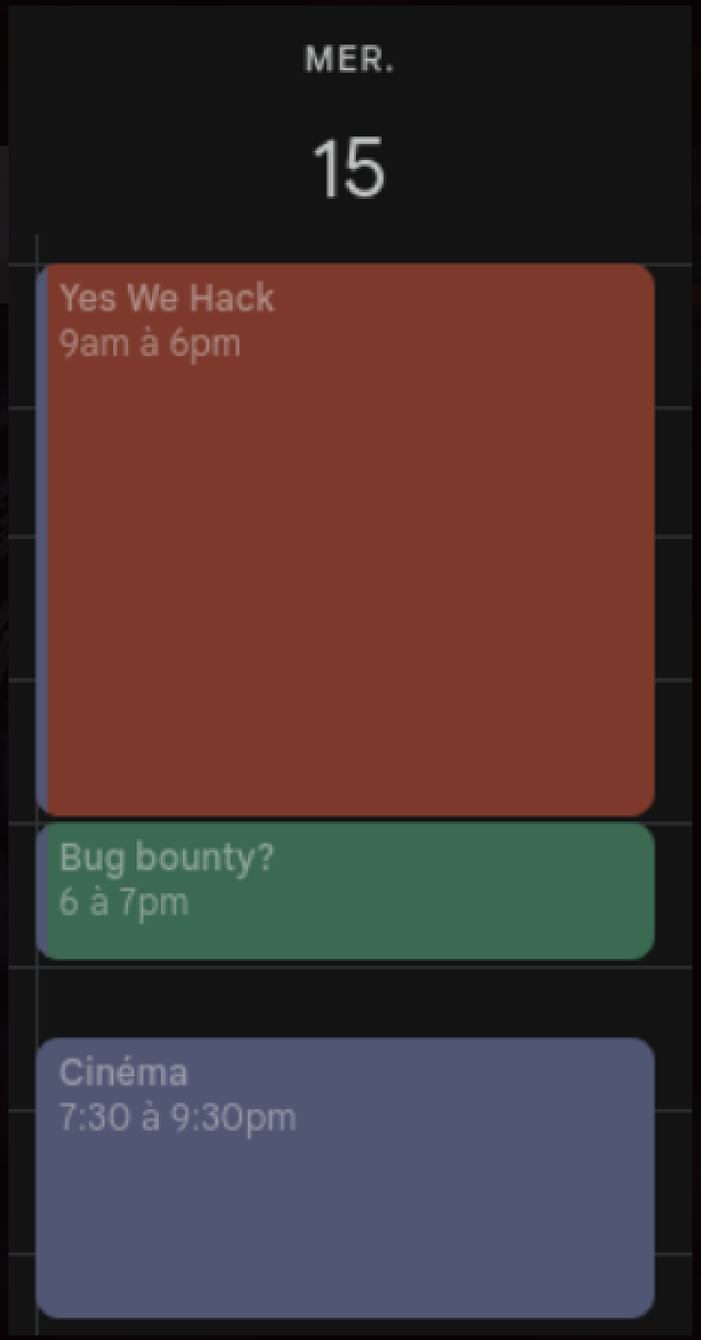
Whoami

Tom CHAMBARETAUD aka **Aethlios**

- Tech lead & Security analyst chez YesWeHack
- Bug bounty hunter indépendant
- Fan de cinéma

Yes We Hack

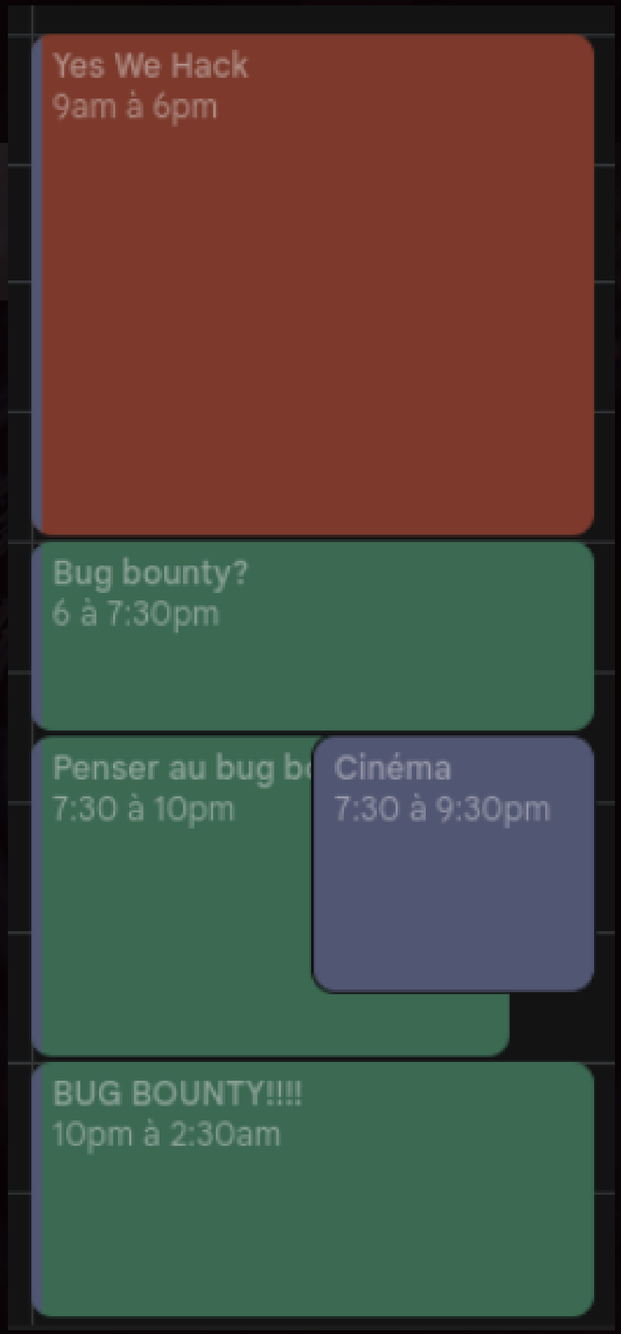
I - Context - Bug bounty x Cinéma



I.1 - Just one more turn

Tokens de réinitialisation séquentiellement générés:

- `/reset/password/token/64120736722fa`
- `/reset/password/token/64120736722fd`



I.2 -Hypothèse confirmée!

```
import datetime

def date_from_token(token):
    return (
        datetime.datetime.fromtimestamp(
            float(str(
                int(token[:8],16)
            ))
        ),
        str(int(token[-5:],16))
    )

>>> d, microtime = date_from_token("64120736722fa")
>>> print(f"{d} {microtime}")
# 2023-03-15 18:58:14 467706
```

Et si ce n'était pas un cas isolé ?

Et si on avait plein de serveurs qui utilisaient des secrets basés sur le temps ?

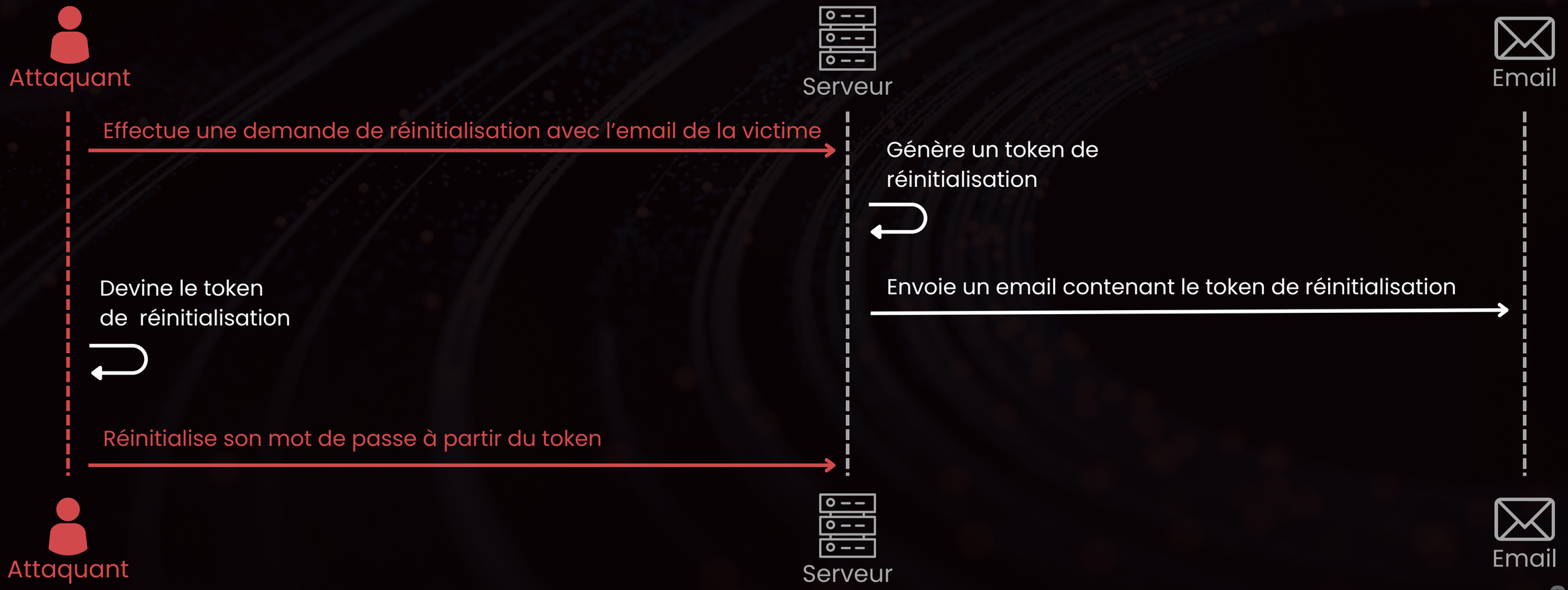
Et si la sécurité était finalement réellement un échec et que je pouvais gagner plus de bounties ?

ça fait beaucoup de "et si" oui oui

I.2 - Fonctionnalité de réinitialisation de mot de passe

I.3 - Si je connais le temps, c'est gagné

I.4 - Et je peux connaître le temps



I.5 – Le format de ces tokens

```python
import math

def uniqid(timestamp: float):
    sec = math.floor(timestamp)
    usec = round(1000000 * (timestamp - sec))
    return "%8x%05x" % (sec, usec)

def reverse_uniqid(value: str):
    return float(
        str(int(value[:8], 16))
        + "."
        + str(int(value[8:], 16))
    )
```

Uniqid

```
tokens = ["64120736722fa", "64120736722fd"]
for token in tokens:
    t = float(reverse_uniqid(token))
    d = datetime.datetime.fromtimestamp(t)
    print(f"{token} - {t} => {d}")

# 64120736722fa - 1678903094.467706 => 2023-03-15 18:58:14.467706
# 64120736722fd - 1678903094.467709 => 2023-03-15 18:58:14.467709
```

- Basé sur le temps en **microsecondes**
- Pas d'aléatoire par défaut

uniqid

(PHP 4, PHP 5, PHP 7, PHP 8)
uniqid — Generate a time-based identifier

Description

```
uniqid(string $prefix = "", bool $more_entropy = false): string
```

Gets an identifier based on the current time with microsecond precision, prefixed with the given **prefix** and optionally appending a randomly generated value.

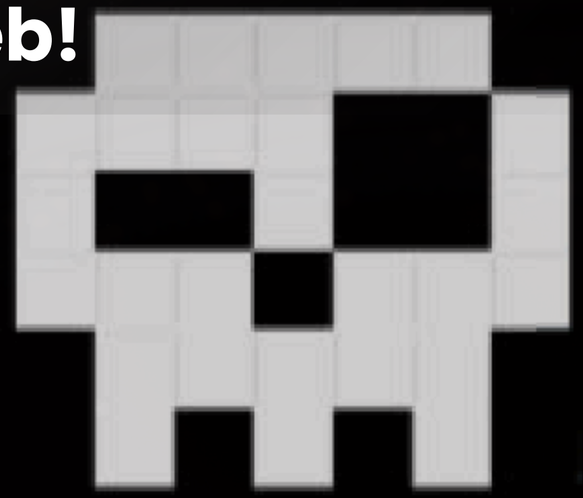
Caution This function does not generate cryptographically secure values, and *must not* be used for cryptographic purposes, or purposes that require returned values to be unguessable.

If cryptographically secure randomness is required, the Random\Randomizer may be used with the Random\Engine\Secure engine. For simple use cases, the random_int() and random_bytes() functions provide a convenient and secure API that is backed by the operating system's CSPRNG.

Warning This function does not guarantee the uniqueness of the return value because the value is based on the current time in microseconds or the current time with a small amount of random data appended if **more_entropy** is **true**.

I.6 - C'est vieux comme le mond... comme le web!

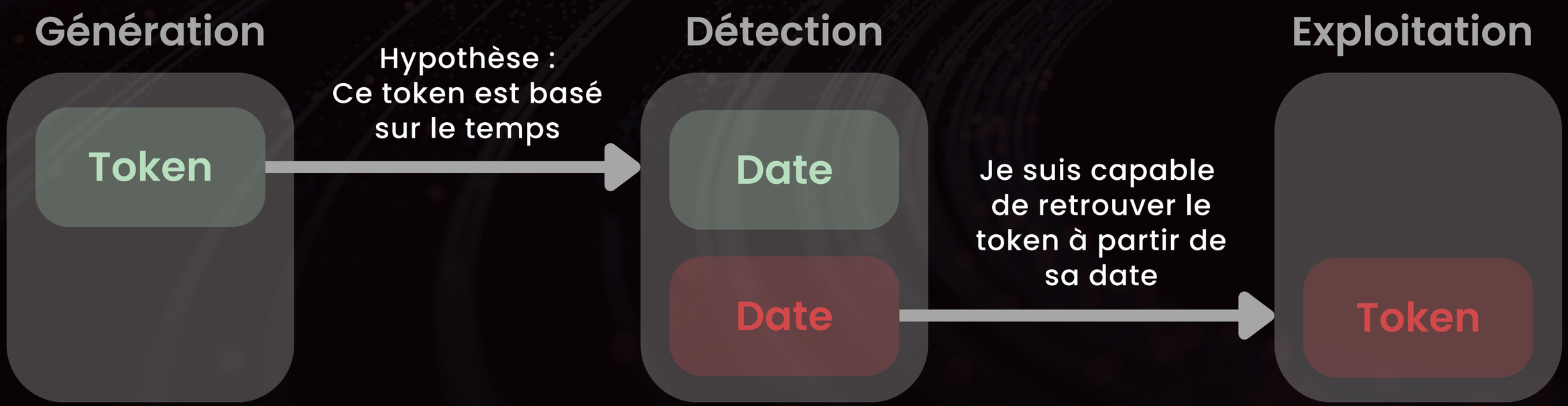
Démystification d'exploits visant des applications web
=> par *Apophis* paru dans le HackerzVoice #1 d'**Octobre 2008**



> La valeur du «cookie_seed» peut être connue car il n'est pas réellement aléatoire: cette valeur est le hash md5 de l'heure (en microsecondes) de la date de création du forum (qui est publique).

II - Formalisons tout ça!

II.1 - Les différentes phases



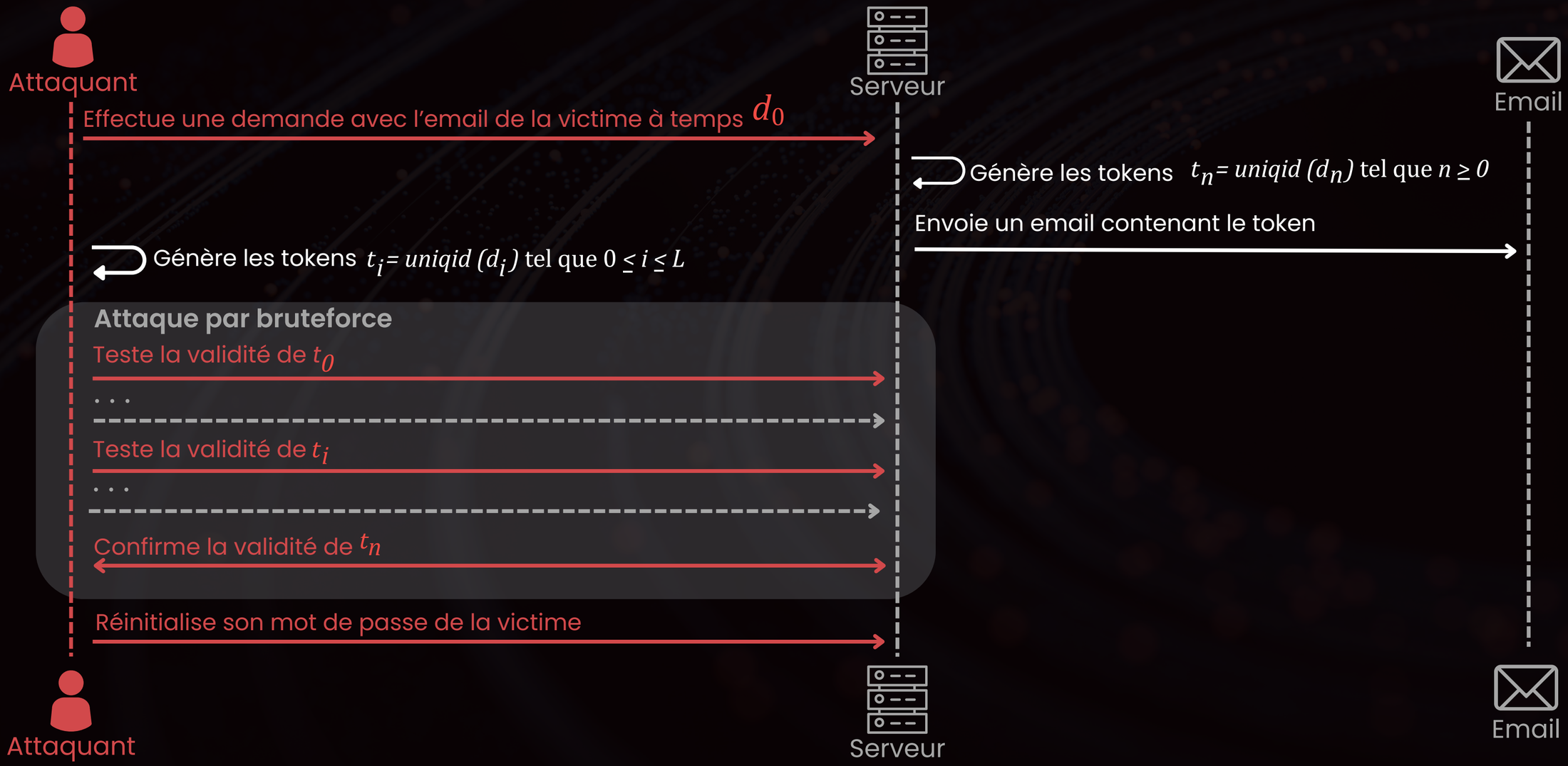
II.2 - Détection - Scénario naïf

- Générez un **token de réinitialisation** avec votre compte.
- Récupérez une **date approximative** de génération et votre **token.**
- Trouvez une corrélation entre votre **token** et sa **date précise** de génération à partir de la date approximative.

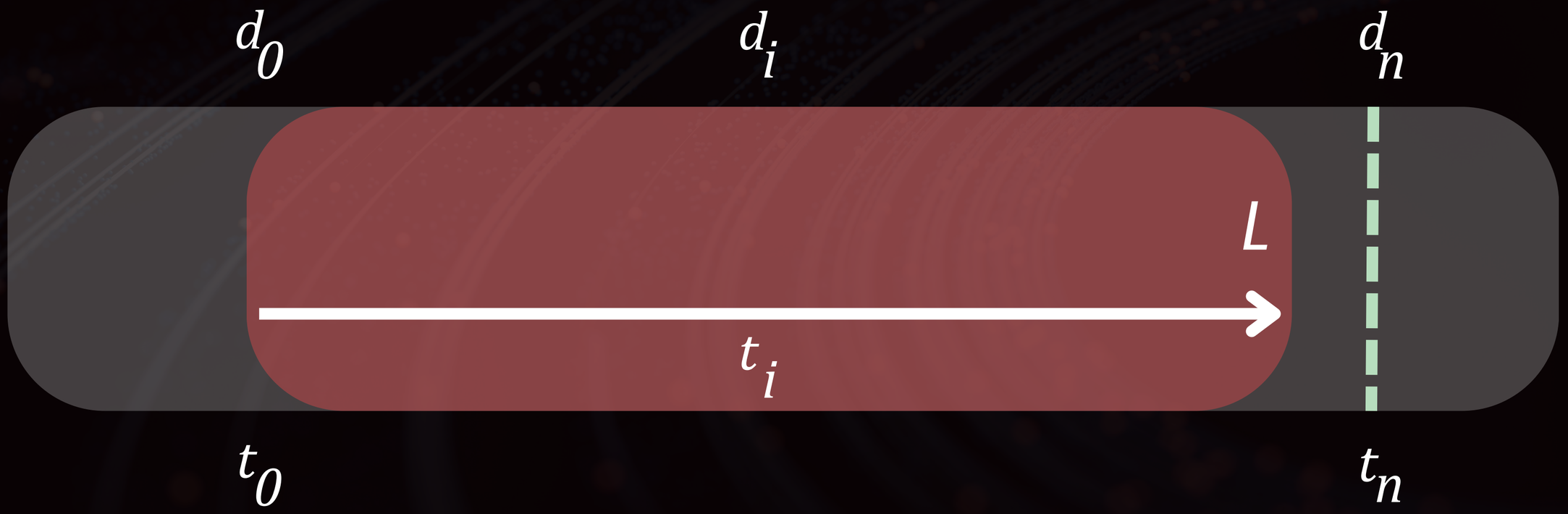
Axiome:

Si l'hypothèse est vraie : `date <- detection(token, ~date)`

II.2 - Attaque par force brute - Scénario naïf



II.3 – Oui, je bruteforce, mais jusqu'à quand?

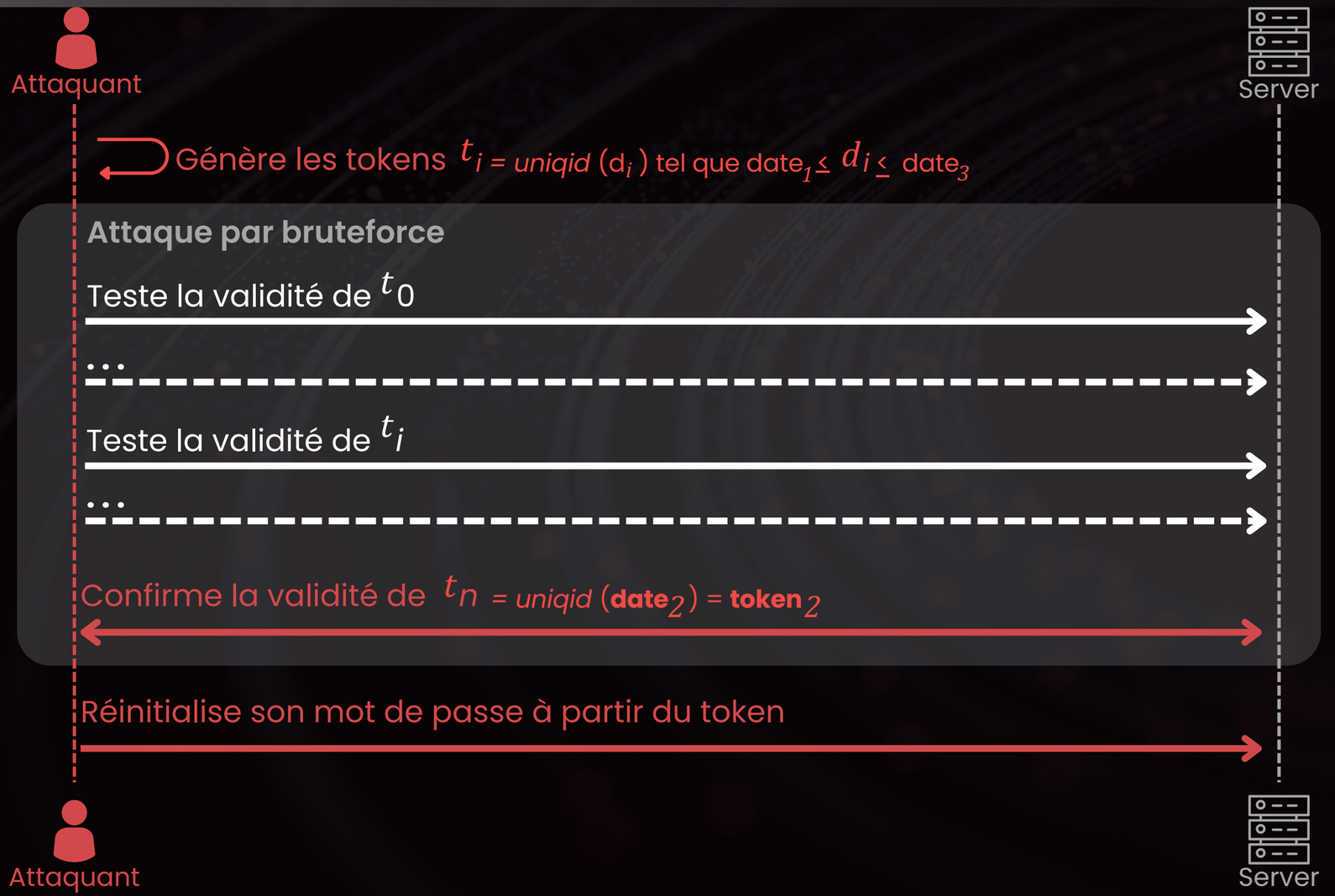


II.4 - Exploitation - Scénario de sandwich


```
date <- detection(token, ~date)
```

II.5 - Exploitation - Scénario de sandwich - Schéma 1

II.5 - Exploitation - Scénario de sandwich - Schéma 2



Eh hop, sandwiched!

III - Automatisation

III.1 - Pourquoi faire?

- Open source
- Y’en a marre du guessing de format des tokens
- Recherche en blackbox
- Le fun ?

III.2 - Automatisation de la détection

Et si ça **aussi**, c'était un token vulnérable ?

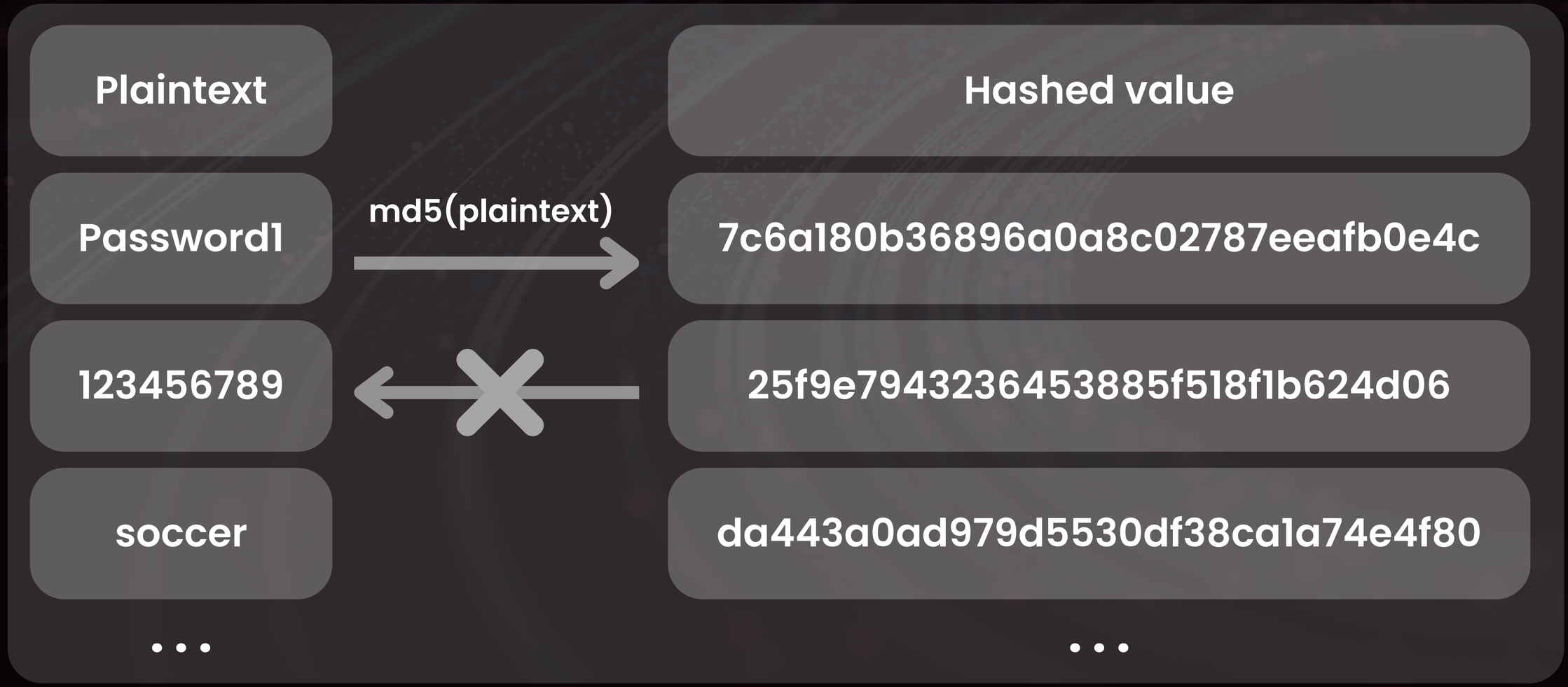
`b8e1d3b8af4cea584d73cb56cf83acc9`

- CVE-2022-39216 sur **ITOP** par .@Blaklis

```
// This token allows the user to change the password without knowing the previous one
$sToken = substr(md5(APPROOT.uniqid()), 0, 16);
$sToken = bin2hex(random_bytes(32));
$oUser->Set('reset_pwd_token', $sToken);
```

III.3 - Détection offline d'un hash basé sur le temps

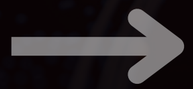
III.4 - Rainbow table



III.5 - By cracking passwor... timestamp ?!

Rainbow table

Plaintext	Hashed value
Password1	7c6a180b36896a0a8c02787eeafb0e4c
123456789	25f9e7943236453885f518f1b624d06
soccer	da443a0ad979d5530df38ca1a74e4f80
...	...



Génération des tokens possibles

Date	Hashed token value
167890394.467706	1ac2ab50c0baee093cd9825f30911c42
167890394.467707	a1317bfad62a7792944d4e2c7d6b3b3
167890394.467708	b292328ffab6aa1dc51f11ce356ccb01
...	...

III.4 - Différence avec le cracking de password

Pas de pré-calcul possible:

- `rockyou.txt` : 14,341,564 de mots de passe
- 1 seconde pré-calculé: 1,000,000 de millisecondes

Stockage d'un journée pré-calculée en SHA256:

22 Téra-octets

IV.1 Reset Tolkien - Démonstration

IV.2 - Créateur de scénario depuis 2024

```php
function generate_token($email){
    return md5(uniqid() . $email);
}
```

- config.yml :

```yaml
float-uniqid:
  description: "Uniqid timestamp"
  level: 2
  timestamp_type: float
  formats:
    - uniqid
```

IV.3 - Encoding / Hashing fonctions

```
base32                  md5
base64                  sha1
urlencode               sha224
hexint                  sha256
hexstr                  sha384
uniqid                  sha512
uuidv1                  sha3_224
shortuuid               sha3_256
mongodb_objectid        sha3_384
datetime                sha3_512
datetimeRFC2822         blake_256
                        blake_512
```

V - MongoDB Object ID

V.1 - Trouvé en bug bounty - Épisode 2

Tokens de réinitialisation séquentiellement générés :

- `/emailvalidation/65c7e6f47ded1f0fef0c1006`
- `/emailvalidation/65c7e6f47ded1f0fef0c1007`

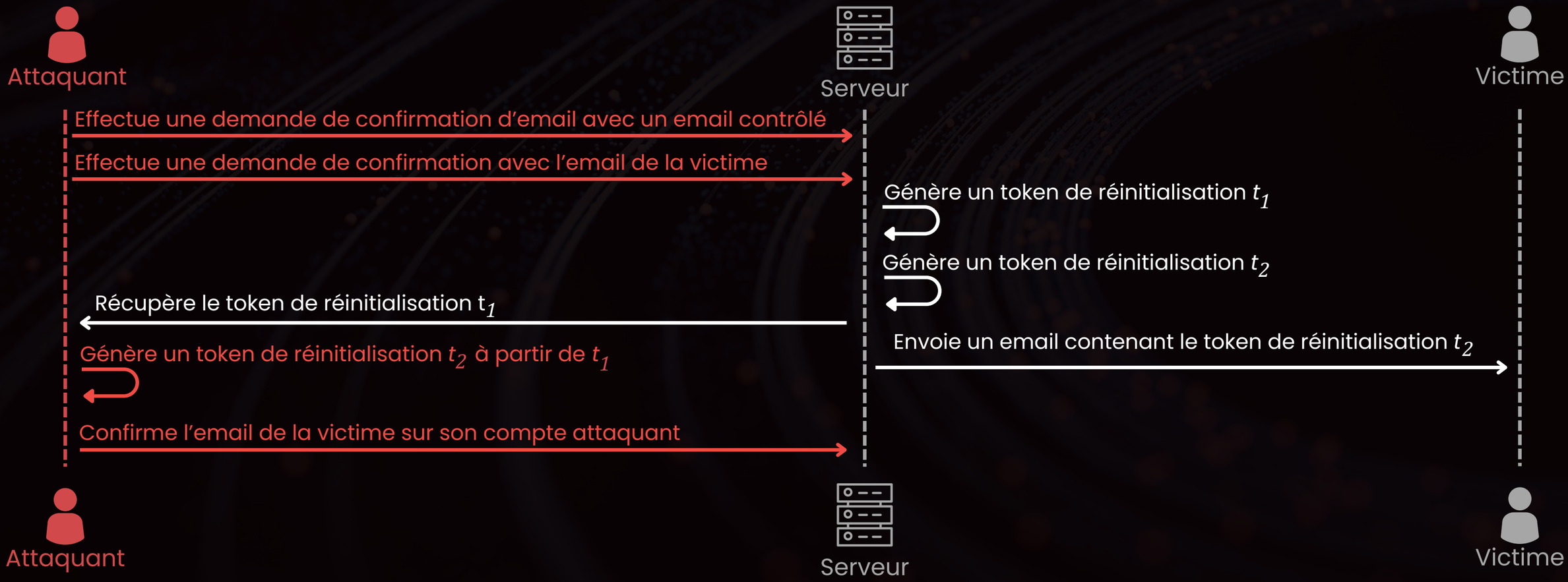
V.2 - Format du token

```
def MongoDB_ObjectID(timestamp, process, counter):
    return "%08x%10x%06x" % (
        timestamp,
        process,
        counter,
    )

def reverse_MongoDB_ObjectID(token):
    timestamp = int(token[0:8], 16)
    process = int(token[8:18], 16)
    counter = int(token[18:24], 16)
    return timestamp, process, counter
```

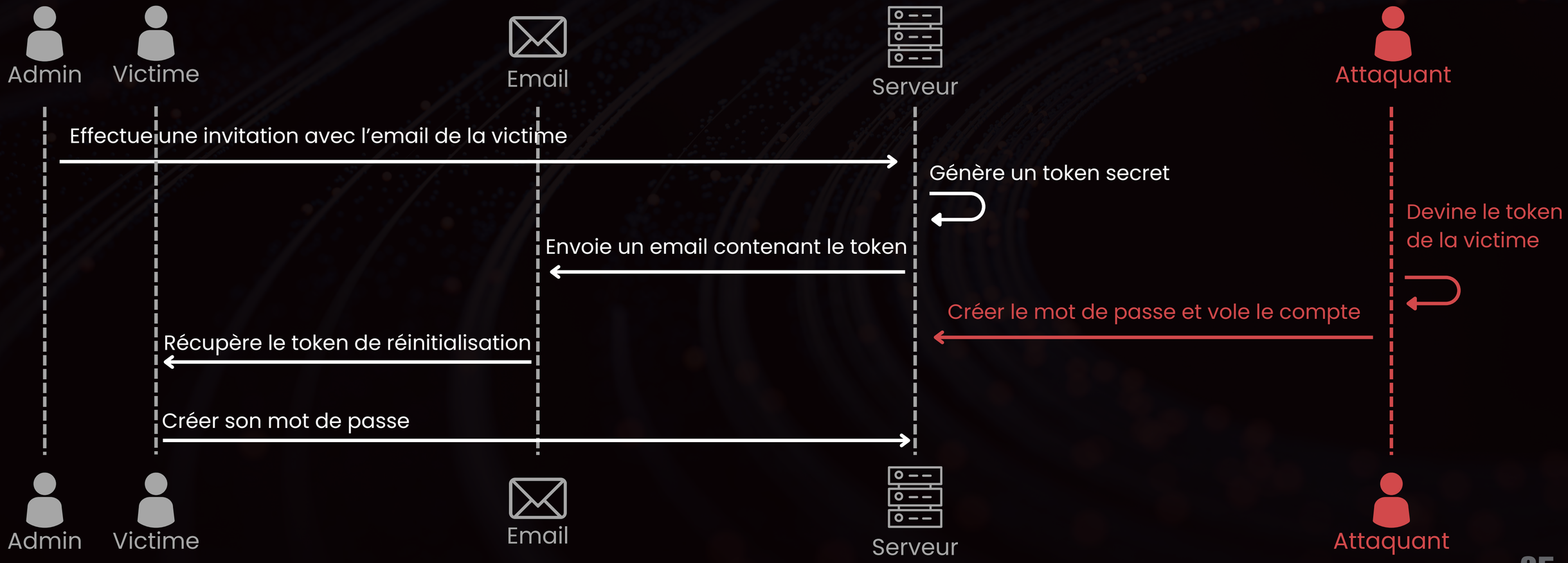
V.2 - Format du token

```python
tokens = ["65c7e6f47ded1f0fef0c1006", "65c7e6f47ded1f0fef0c1007"]
for token in tokens:
    (timestamp, process, counter) = reverse_MongoDB_ObjectID(token)
    print(f"{token}: {timestamp} - {process} - {counter}")

# 65c7e6f47ded1f0fef0c1006: 1707599604 - 540849147887 - 790534
# 65c7e6f47ded1f0fef0c1007: 1707599604 - 540849147887 - 790535
```

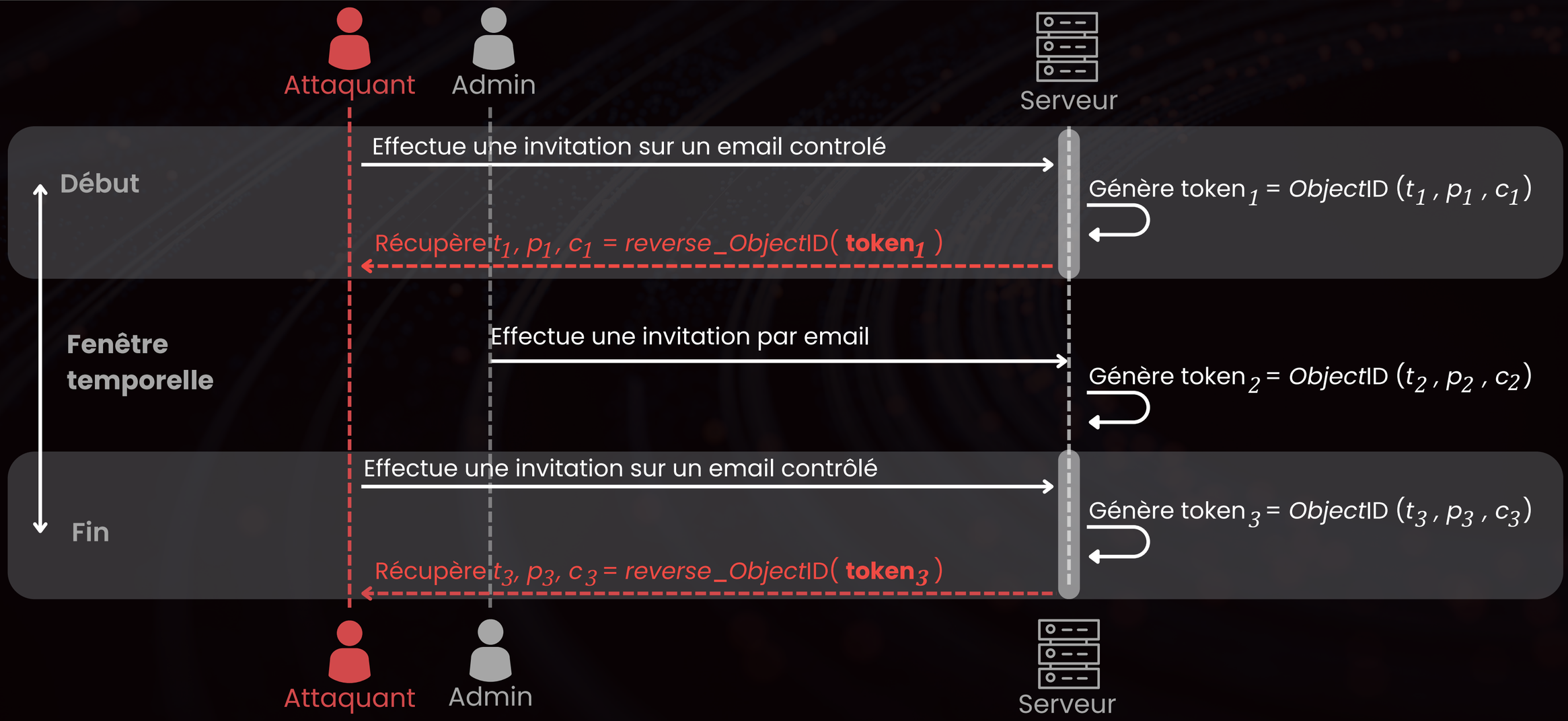
V.3 - Fonctionnalité de confirmation d'email



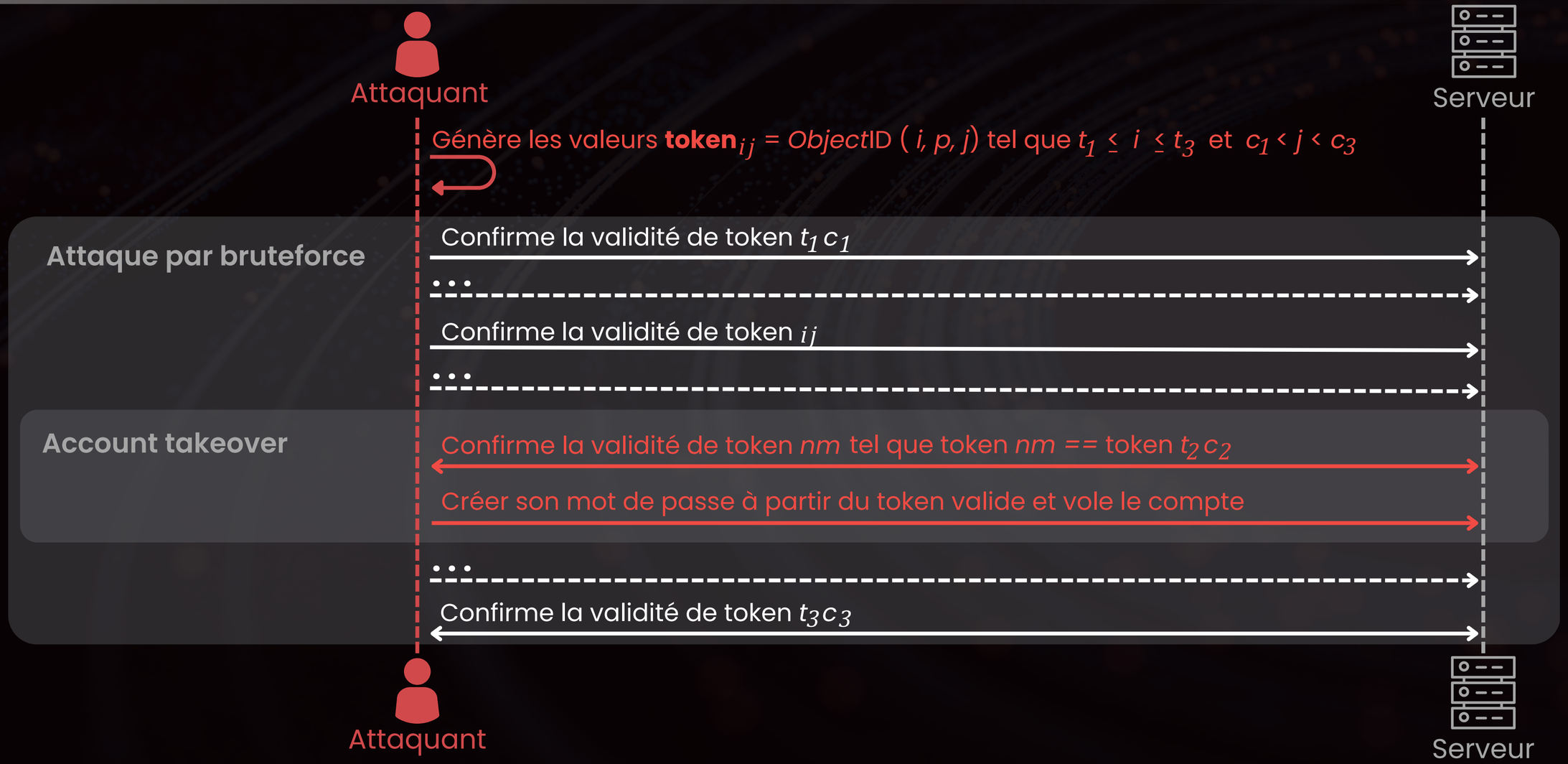
V.4 - Fonctionnalité d'invitation à un groupe



V.5 - Sandwich sur une invitation - Première partie

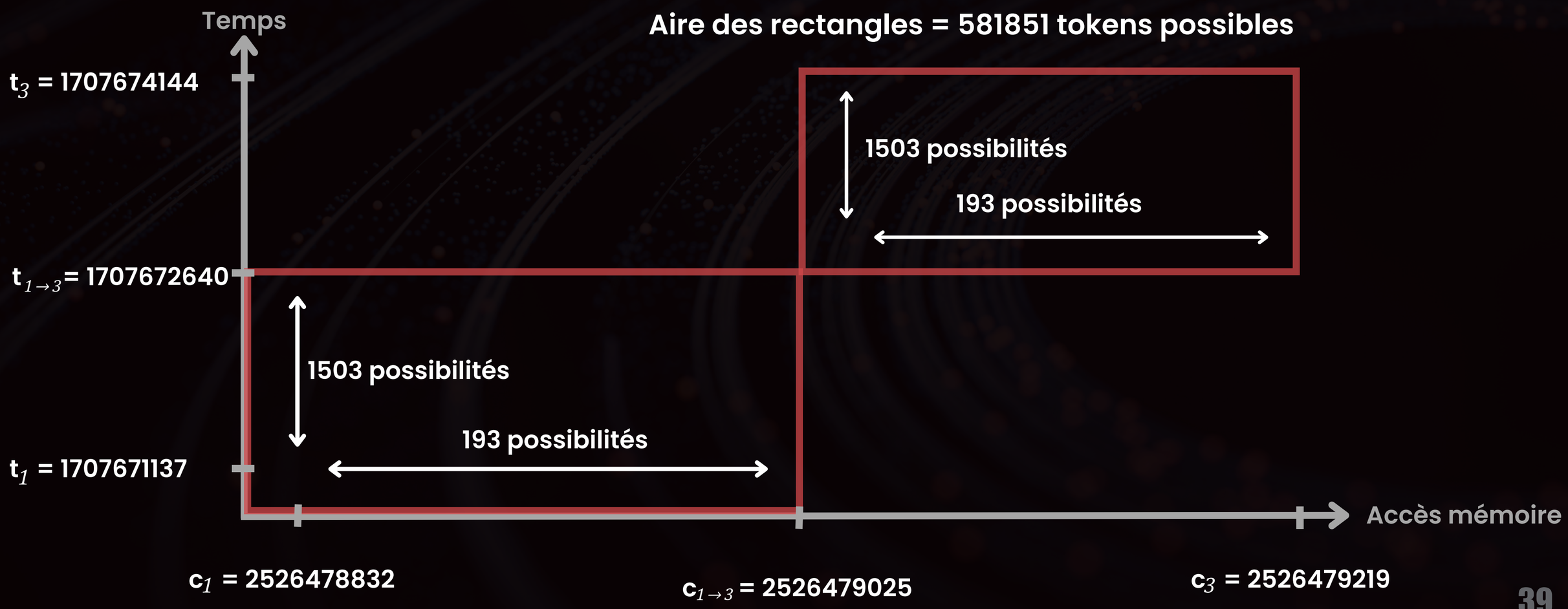


V.5 - Sandwich sur une invitation - Deuxième partie



V.6 -Complexité - O(n2)

V.7 - Complexité - Optimisation



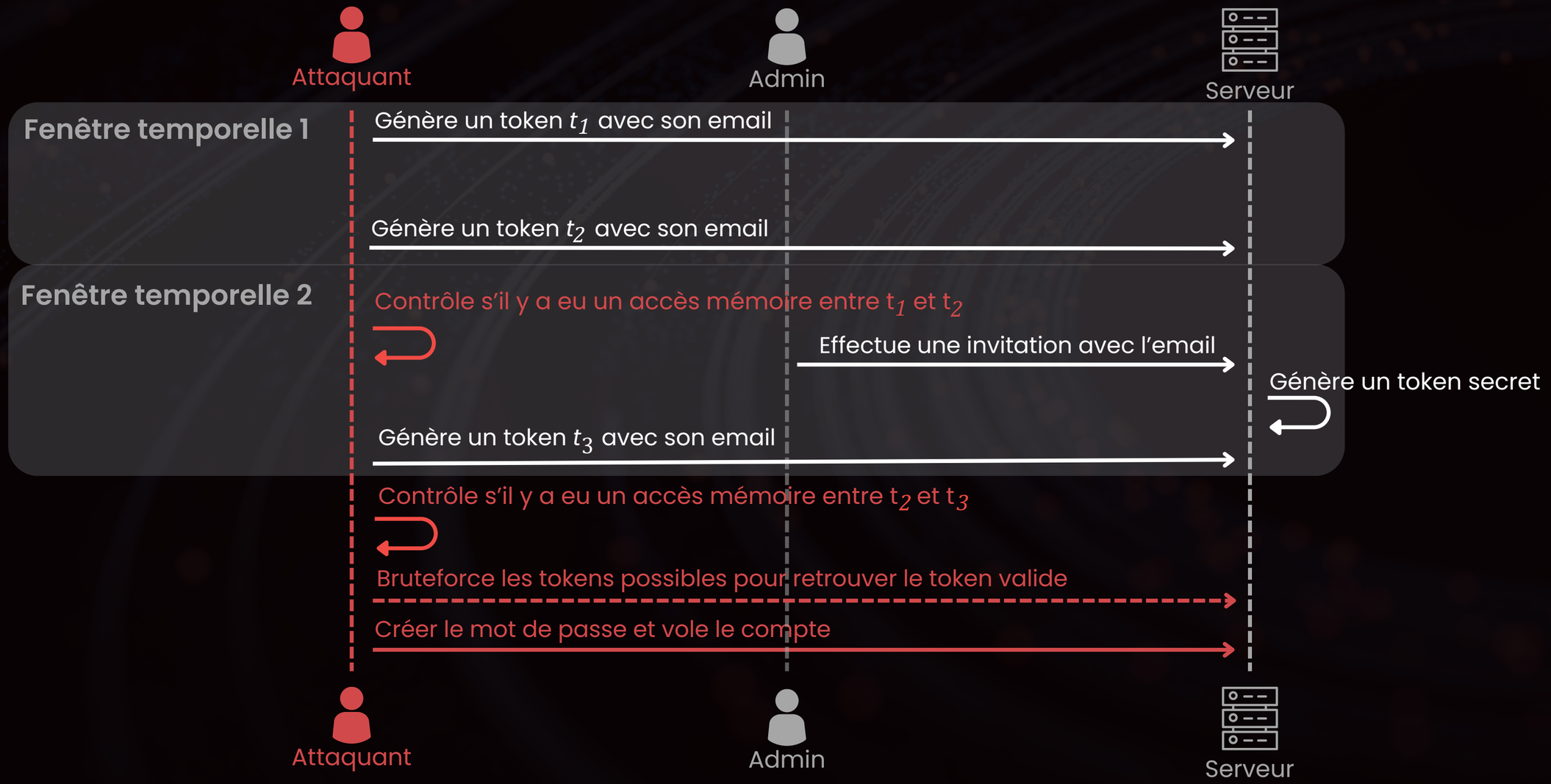
V.7 - Complexité - Taille optimisée des fenêtres temporelles

Nombre de tokens intermédiaires	Taille de chaque fenêtre temporelle	Nombre de tokens possibles	Requêtes par seconde pour les tester tous durant 50 minutes
0	50 minutes	1163707	387.9 req/s
1	25 minutes	581851	193.95 req/s
3	12 minutes et 30 secondes	290920	96.97 req/s
...	...	...	...
255	11.71875 secondes	4067	1.36 req/s

VI - Monitoring de l'évolution du compteur

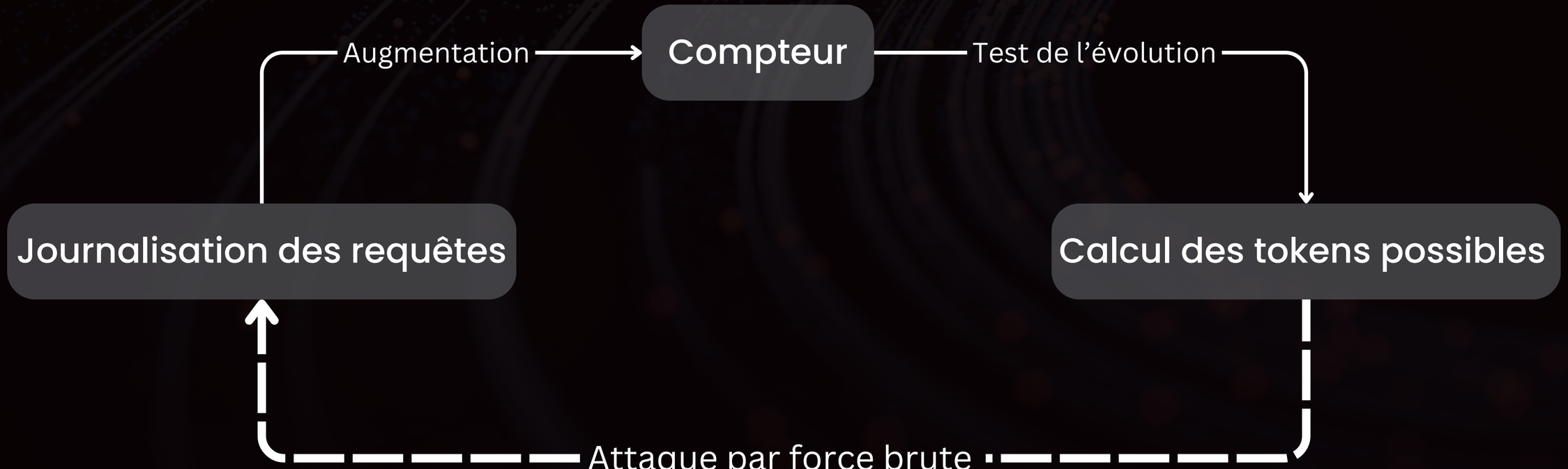
- Si le compteur **évolue de manière constante:**
  - Pas besoin de vérifier tous les tokens
- Si le compteur **évolue de manière inhabituelle:**
  - On vérifie tous les tokens pour récupérer les **valides**

VI.1 - Monitoring de l'évolution du compteur - Théorie



VI.2 - Monitoring de l'évolution du compteur - En pratique...

- L’évolution du compteur dépend des écritures mémoires, quelque soit le type de data.
- Si la base de données MongoDB est utilisée pour journaliser les requêtes:



VI.3 - Et si c'était un bon exemple de ce qui ne va pas?

VII - Conclusion

Oui la sécurité est un échec.

VII - La vrai conclusion

Websites are riddled with timing oracles eager to divulge their innermost secrets. It's time we started listening to them.

Les sites web sont truffés d'oracles du temps qui ne demandent qu'à divulguer leurs secrets les plus intimes. Il est temps que nous commencions à les écouter.

James Kettle, Director of Research of Portswigger

Des questions?

Merci de votre attention.