

### Obfuscation de code :

- Instructions inutiles/Arguments inutiles : ALourdi le programme
- Minification : pratique répandue, permettant de réduire la taille du code en renommant les variables et classes par les lettres, ralentie la lecture du code à la décompilation
- Ralentie l'application, car ces opérations

sont lourdes

### Chiffrement du programme :

- Ralentit le lancement du programme, le programme doit forcément être déchiffré pour être exécuté, on peut alors essayer de l'analyser

### Execution du code distant :

- Les applications web, nécessitent une connexion internet

## Comment s'en prémunir ? : La sécurité par l'obscurité

- Comment s'en prémunir ?



FIGURE – XKCD 257







- Instructions inutiles/Arguments inutiles :  
ALourdi le programme

- Minification : pratique répandue, permettant de réduire la taille du code en renommant les variables et classes par les lettres, ralentit la lecture du code à la décompilation
- Ralentit l'application, car ces opérations

sont lourdes

### Chiffrement du programme :

- Ralentir le lancement du programme, le programme doit forcément être déchiffré pour être exécuté, on peut alors essayer de l'analyser

### Execution du code distant :

- Les applications web, nécessite une connexion internet

## Comment s'en prémunir? : La sécurité par l'obscurité

- Comment s'en prémunir?

## Obscurcir son code

- ▶ Obfuscation de code :
  - ▶ Ajout d'instructions inutiles
  - ▶ Ajout d'arguments inutiles sur les méthodes



FIGURE – XKCD 257

- **Ralentie l'application, car ces opérations**

- Les applications web, nécessite une connexion internet

## 1 / 2









### Limites :

- Les opérations de déboguage et la réalisation de traces ne pourront plus être proposées sur la version commerciale de l'application. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
- La sécurité par l'opacité est un mythe! L'obfuscation protège le code source contre la piraterie intellectuelle durant un temps assez court. Mais elle ne protège pas des pirates voulant exploiter les failles de sécurité de l'application.

- L'augmentation de la complexité algorithmique et la modification des structures de données augmentent le temps d'exécution. Les patrons de conceptions choisis par le développeur disparaissent et peuvent être remplacés par d'autres moins efficaces.
- La qualité (algorithmique) du code source baisse considérablement et peut être un frein à la certification par des organismes tiers.
- L'appel à des API externes (notamment en Java) fait par le nom ne peut PAS être obfusqué, et donnent alors des indices aux pirates.

## Comment s'en prémunir ? : L'absurdité de l'obscurité

- Comment s'en prémunir?



### Limites :

- Les opérations de déboguage et la réalisation de traces ne pourront plus être proposées sur la version commerciale de l'application. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
- La sécurité par l'opacité est un mythe! L'obfuscation protège le code source contre la piraterie intellectuelle durant un temps assez court. Mais elle ne protège pas des pirates voulant exploiter les failles de sécurité de l'application.

- L'augmentation de la complexité algorithmique et la modification des structures de données augmentent le temps d'exécution. Les patrons de conceptions choisis par le développeur disparaissent et peuvent être remplacés par d'autres moins efficaces.
- La qualité (algorithmique) du code source baisse considérablement et peut être un frein à la certification par des organismes tiers.
- L'appel à des API externes (notamment en Java) fait par le nom ne peut PAS être obfusqué, et donnent alors des indices aux pirates.

## Comment s'en prémunir ? : L'absurdité de l'obscurité

• Comment s'en prémunir ?

### Principe de Kerckhoffs



### Limites :

- Les opérations de déboguage et la réalisation de traces ne pourront plus être proposées sur la version commerciale de l'application. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
- La sécurité par l'opacité est un mythe! L'obfuscation protège le code source contre la piraterie intellectuelle durant un temps assez court. Mais elle ne protège pas des pirates voulant exploiter les failles de sécurité de l'application.

- L'augmentation de la complexité algorithmique et la modification des structures de données augmentent le temps d'exécution. Les patrons de conceptions choisis par le développeur disparaissent et peuvent être remplacés par d'autres moins efficaces.
- La qualité (algorithmique) du code source baisse considérablement et peut être un frein à la certification par des organismes tiers.
- L'appel à des API externes (notamment en Java) fait par le nom ne peut PAS être obfusqué, et donnent alors des indices aux pirates.

## Comment s'en prémunir ? : L'absurdité de l'obscurité

• Comment s'en prémunir?

### Principe de Kerckhoffs

"Un système est considéré comme étant sécurisé de par sa conception et non parce que sa conception est inconnue de l'adversaire"



### Limites :

- Les opérations de déboguage et la réalisation de traces ne pourront plus être proposées sur la version commerciale de l'application. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
- La sécurité par l'opacité est un mythe! L'obfuscation protège le code source contre la piraterie intellectuelle durant un temps assez court. Mais elle ne protège pas des pirates voulant exploiter les failles de sécurité de l'application.

- L'augmentation de la complexité algorithmique et la modification des structures de données augmentent le temps d'exécution. Les patrons de conceptions choisis par le développeur disparaissent et peuvent être remplacés par d'autres moins efficaces.
- La qualité (algorithmique) du code source baisse considérablement et peut être un frein à la certification par des organismes tiers.
- L'appel à des API externes (notamment en Java) fait par le nom ne peut PAS être obfusqué, et donnent alors des indices aux pirates.

## Comment s'en prémunir ? : L'absurdité de l'obscurité

• Comment s'en prémunir?

### Principe de Kerckhoffs

"Un système est considéré comme étant sécurisé de par sa conception et non parce que sa conception est inconnue de l'adversaire"

### Les limites de l'obfuscation



### Limites :

- Les opérations de déboguage et la réalisation de traces ne pourront plus être proposées sur la version commerciale de l'application. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
- La sécurité par l'opacité est un mythe! L'obfuscation protège le code source contre la piraterie intellectuelle durant un temps assez court. Mais elle ne protège pas des pirates voulant exploiter les failles de sécurité de l'application.

- L'augmentation de la complexité algorithmique et la modification des structures de données augmentent le temps d'exécution. Les patrons de conceptions choisis par le développeur disparaissent et peuvent être remplacés par d'autres moins efficaces.
- La qualité (algorithmique) du code source baisse considérablement et peut être un frein à la certification par des organismes tiers.
- L'appel à des API externes (notamment en Java) fait par le nom ne peut PAS être obfusqué, et donnent alors des indices aux pirates.

## Comment s'en prémunir ? : L'absurdité de l'obscurité

• Comment s'en prémunir?

### Principe de Kerckhoffs

"Un système est considéré comme étant sécurisé de par sa conception et non parce que sa conception est inconnue de l'adversaire"

### Les limites de l'obfuscation

- Débogage difficile



### Limites :

- Les opérations de déboguage et la réalisation de traces ne pourront plus être proposées sur la version commerciale de l'application. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
- La sécurité par l'opacité est un mythe! L'obfuscation protège le code source contre la piraterie intellectuelle durant un temps assez court. Mais elle ne protège pas des pirates voulant exploiter les failles de sécurité de l'application.

- L'augmentation de la complexité algorithmique et la modification des structures de données augmentent le temps d'exécution. Les patrons de conceptions choisis par le développeur disparaissent et peuvent être remplacés par d'autres moins efficaces.
- La qualité (algorithmique) du code source baisse considérablement et peut être un frein à la certification par des organismes tiers.
- L'appel à des API externes (notamment en Java) fait par le nom ne peut PAS être obfusqué, et donnent alors des indices aux pirates.

## Comment s'en prémunir ? : L'absurdité de l'obscurité

• Comment s'en prémunir?

### Principe de Kerckhoffs

"Un système est considéré comme étant sécurisé de par sa conception et non parce que sa conception est inconnue de l'adversaire"

### Les limites de l'obfuscation

- ▶ Débogage difficile
- ▶ Protection temporaire



### Limites :

- Les opérations de déboguage et la réalisation de traces ne pourront plus être proposées sur la version commerciale de l'application. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
- La sécurité par l'opacité est un mythe! L'obfuscation protège le code source contre la piraterie intellectuelle durant un temps assez court. Mais elle ne protège pas des pirates voulant exploiter les failles de sécurité de l'application.

- L'augmentation de la complexité algorithmique et la modification des structures de données augmentent le temps d'exécution. Les patrons de conceptions choisis par le développeur disparaissent et peuvent être remplacés par d'autres moins efficaces.
- La qualité (algorithmique) du code source baisse considérablement et peut être un frein à la certification par des organismes tiers.
- L'appel à des API externes (notamment en Java) fait par le nom ne peut PAS être obfusqué, et donnent alors des indices aux pirates.

## Comment s'en prémunir ? : L'absurdité de l'obscurité

• Comment s'en prémunir?

### Principe de Kerckhoffs

"Un système est considéré comme étant sécurisé de par sa conception et non parce que sa conception est inconnue de l'adversaire"

### Les limites de l'obfuscation

- ▶ Débogage difficile
- ▶ Protection temporaire
- ▶ Potentiel perte de performances





### Limites :

- Les opérations de déboguage et la réalisation de traces ne pourront plus être proposées sur la version commerciale de l'application. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
- La sécurité par l'opacité est un mythe! L'obfuscation protège le code source contre la piraterie intellectuelle durant un temps assez court. Mais elle ne protège pas des pirates voulant exploiter les failles de sécurité de l'application.

- L'augmentation de la complexité algorithmique et la modification des structures de données augmentent le temps d'exécution. Les patrons de conceptions choisis par le développeur disparaissent et peuvent être remplacés par d'autres moins efficaces.
- La qualité (algorithmique) du code source baisse considérablement et peut être un frein à la certification par des organismes tiers.
- L'appel à des API externes (notamment en Java) fait par le nom ne peut PAS être obfusqué, et donnent alors des indices aux pirates.

## Comment s'en prémunir ? : L'absurdité de l'obscurité

• Comment s'en prémunir?

### Principe de Kerckhoffs

"Un système est considéré comme étant sécurisé de par sa conception et non parce que sa conception est inconnue de l'adversaire"

### Les limites de l'obfuscation

- Débogage difficile
- Protection temporaire
- Potentiel perte de performances
- Qualité du code en baisse



### Limites :

- Les opérations de déboguage et la réalisation de traces ne pourront plus être proposées sur la version commerciale de l'application. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
- La sécurité par l'opacité est un mythe! L'obfuscation protège le code source contre la piraterie intellectuelle durant un temps assez court. Mais elle ne protège pas des pirates voulant exploiter les failles de sécurité de l'application.

- L'augmentation de la complexité algorithmique et la modification des structures de données augmentent le temps d'exécution. Les patrons de conceptions choisis par le développeur disparaissent et peuvent être remplacés par d'autres moins efficaces.
- La qualité (algorithmique) du code source baisse considérablement et peut être un frein à la certification par des organismes tiers.
- L'appel à des API externes (notamment en Java) fait par le nom ne peut PAS être obfusqué, et donnent alors des indices aux pirates.

## Comment s'en prémunir ? : L'absurdité de l'obscurité

• Comment s'en prémunir?

### Principe de Kerckhoffs

"Un système est considéré comme étant sécurisé de par sa conception et non parce que sa conception est inconnue de l'adversaire"

### Les limites de l'obfuscation

- ▶ Débogage difficile
- ▶ Protection temporaire
- ▶ Potentiel perte de performances
- ▶ Qualité du code en baisse
- ▶ Appel à des bibliothèques externes non obfusquables

