Cacher de code : Obfuscation de code :

- Instructions inutiles/Arguments inutiles : ALourdi le programme
- Minification : pratique répandue, permettant de réduire la taille du code en renommantles variables et classes par une lettre, ralentie la lecture du code à la décompilation
- Ralenti l'application, car ces opérations sont lourdes

Chiffrement du programme :

- Ralenti 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, mais nécessite une connexion internet

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

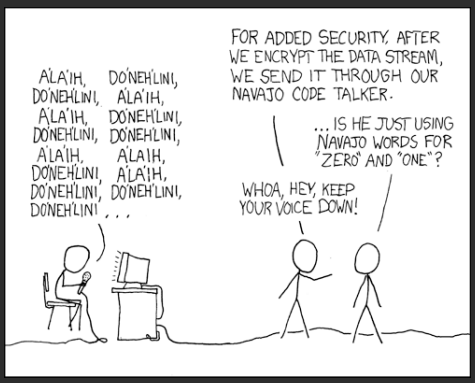


FIGURE – XKCD 257

Cacher de code : Obfuscation de code :

- Instructions inutiles/Arguments inutiles : ALourdi le programme
- Minification : pratique répandue, permettant de réduire la taille du code en renommantles variables et classes par une lettre, ralentie la lecture du code à la décompilation
- Ralenti l'application, car ces opérations sont lourdes

Chiffrement du programme :

- Ralenti 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, mais nécessite une connexion internet

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

Obscurcire son code

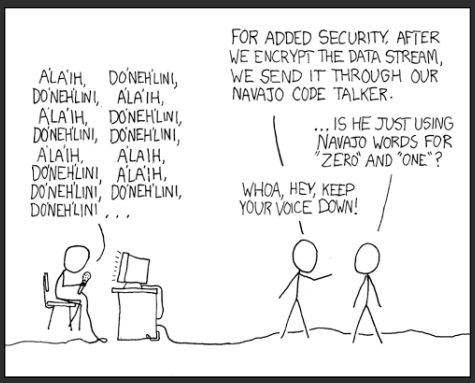


FIGURE – XKCD 257

Cacher de code : Obfuscation de code :

- Instructions inutiles/Arguments inutiles : ALourdi le programme
- Minification : pratique répandue, permettant de réduire la taille du code en renommantles variables et classes par une lettre, ralentie la lecture du code à la décompilation
- Ralenti l'application, car ces opérations sont lourdes

Chiffrement du programme :

- Ralenti 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, mais nécessite une connexion internet

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

Obscurcire son code

- Obfuscation de code :

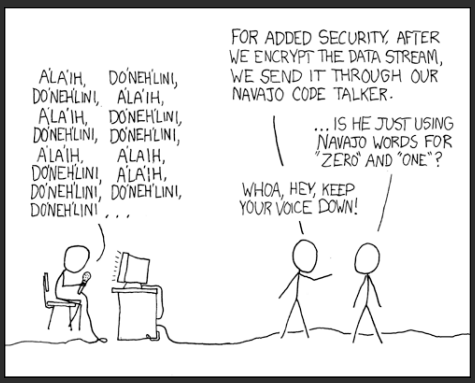


FIGURE – XKCD 257

Cacher de code : Obfuscation de code :

- Instructions inutiles/Arguments inutiles : ALourdi le programme
- Minification : pratique répandue, permettant de réduire la taille du code en renommantles variables et classes par une lettre, ralentie la lecture du code à la décompilation
- Ralenti l'application, car ces opérations sont lourdes

Chiffrement du programme :

- Ralenti 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, mais nécessite une connexion internet

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

Obscurcire son code

- Obfuscation de code :
  - Ajout d'instructions inutiles

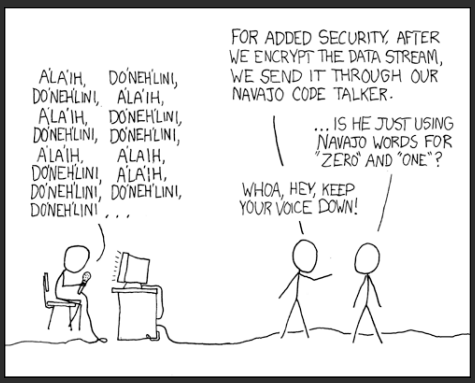


FIGURE – XKCD 257

Cacher de code : Obfuscation de code :

- Instructions inutiles/Arguments inutiles : ALourdi le programme
- Minification : pratique répandue, permettant de réduire la taille du code en renommantles variables et classes par une lettre, ralentie la lecture du code à la décompilation
- Ralenti l'application, car ces opérations sont lourdes

Chiffrement du programme :

- Ralenti 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, mais nécessite une connexion internet

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

Obscurcire son code

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

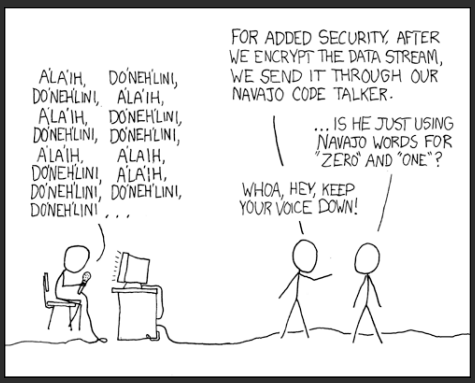


FIGURE – XKCD 257

Cacher de code : Obfuscation de code :

- Instructions inutiles/Arguments inutiles : ALourdi le programme
- Minification : pratique répandue, permettant de réduire la taille du code en renommantles variables et classes par une lettre, ralentie la lecture du code à la décompilation
- Ralenti l'application, car ces opérations sont lourdes

Chiffrement du programme :

- Ralenti 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, mais nécessite une connexion internet

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

Obscurcire son code

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

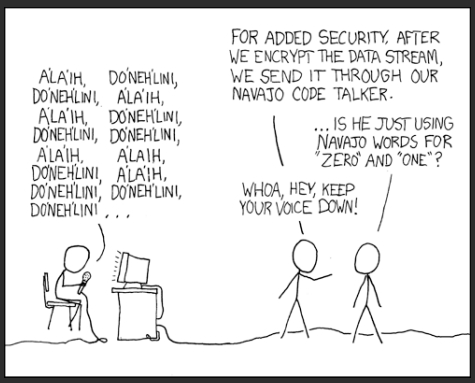


FIGURE – XKCD 257

Cacher de code : Obfuscation de code :

- Instructions inutiles/Arguments inutiles : ALourdi le programme
- Minification : pratique répandue, permettant de réduire la taille du code en renommantles variables et classes par une lettre, ralentie la lecture du code à la décompilation
- Ralenti l'application, car ces opérations sont lourdes

Chiffrement du programme :

- Ralenti 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, mais nécessite une connexion internet

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

Obscurcire son code

- Obfuscation de code :
  - Ajout d'instructions inutiles
  - Ajout d'arguments inutiles sur les méthodes
  - Minimication du code
  - Génération dynamique de string

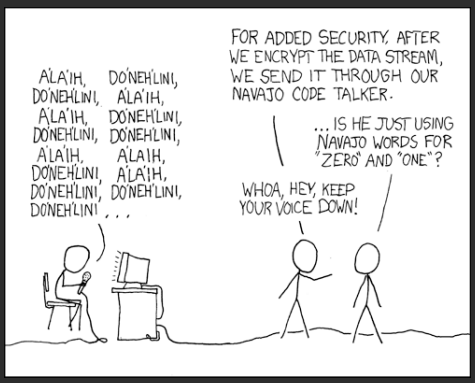


FIGURE – XKCD 257

Cacher de code : Obfuscation de code :

- Instructions inutiles/Arguments inutiles : ALourdi le programme
- Minification : pratique répandue, permettant de réduire la taille du code en renommantles variables et classes par une lettre, ralentie la lecture du code à la décompilation
- Ralenti l'application, car ces opérations sont lourdes

Chiffrement du programme :

- Ralenti 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, mais nécessite une connexion internet

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

Obscurcire son code

- Obfuscation de code :
  - Ajout d'instructions inutiles
  - Ajout d'arguments inutiles sur les méthodes
  - Minimication du code
  - Génération dynamique de string
- Chiffrement du programme

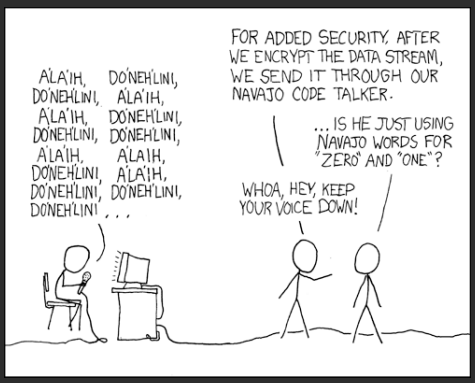


FIGURE – XKCD 257

Cacher de code : Obfuscation de code :

- Instructions inutiles/Arguments inutiles : ALourdi le programme
- Minification : pratique répandue, permettant de réduire la taille du code en renommantles variables et classes par une lettre, ralentie la lecture du code à la décompilation
- Ralenti l'application, car ces opérations sont lourdes

Chiffrement du programme :

- Ralenti 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, mais nécessite une connexion internet

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

Obscurcire son code

- Obfuscation de code :
  - Ajout d'instructions inutiles
  - Ajout d'arguments inutiles sur les méthodes
  - Minimication du code
  - Génération dynamique de string
- Chiffrement du programme
- Exécution de code distant

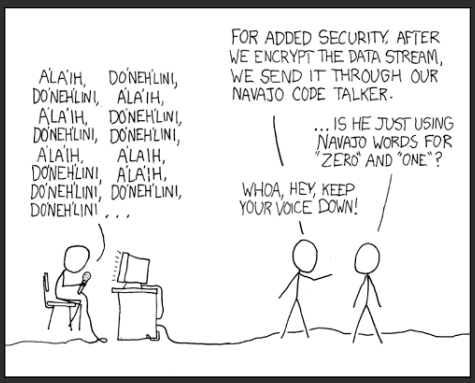
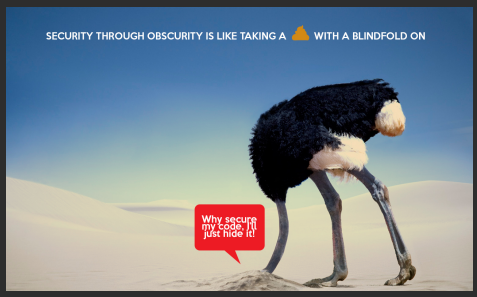


FIGURE – XKCD 257

Principe de Kerckhoffs :
Un système est considéré commeétant sécurisé de par sa conceptionet non parce que sa conception estinconnue de l'adversaire

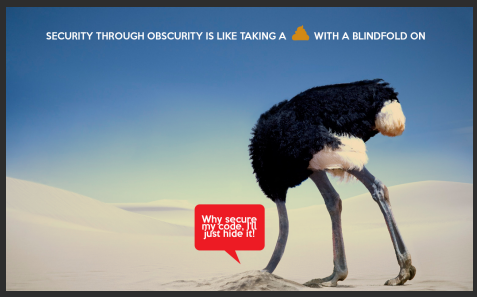
Comment s'en prémunir ? : L'absurdité de l'obscurité



Principe de Kerckhoffs :
Un système est considéré commeétant sécurisé de par sa conceptionet non parce que sa conception estinconnue de l'adversaire

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

Principe de Kerckhoffs



Principe de Kerckhoffs :
Un système est considéré commeétant sécurisé de par sa conceptionet non parce que sa conception estinconnue de l'adversaire

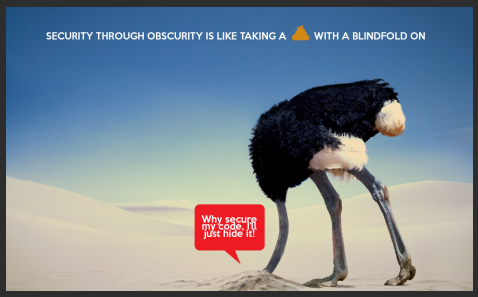
Limites :
• Débogage et récupération de log plus difficile. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
• L'obfuscation protège le code source contre la piraterie intellectuelle durant un temps assez court. Mais elle ne protège pas des pirates voulant exploiter les failles de sécurité de l'application.
• L'augmentation de la complexité algorithmique et la modification des structures de données augmentent le temps d'exécution.
• 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 fait par le nom ne peut pas être obfusqué.

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"



Principe de Kerckhoffs :
Un système est considéré commeétant sécurisé de par sa conceptionet non parce que sa conception estinconnue de l'adversaire

Limites :
• Débogage et récupération de log plus difficile. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
• 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.
• 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 fait par le nom ne peut pas être obfusqué.

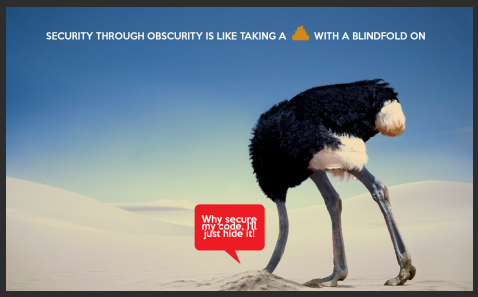
Comment s'en prémunir ? : L'absurdité de l'obscurité

• **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



Principe de Kerckhoffs :
Un système est considéré commeétant sécurisé de par sa conceptionet non parce que sa conception estinconnue de l'adversaire

Limites :
• Débogage et récupération de log plus difficile. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
• 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.
• 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 fait par le nom ne peut pas être obfusqué.

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

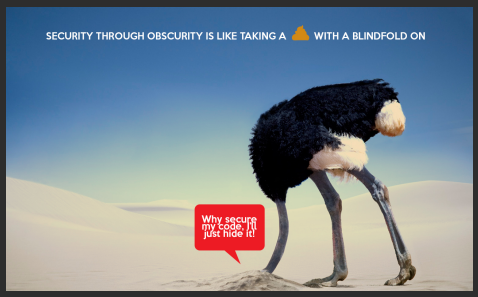
• **Comment s'en prémunir ?**

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



Principe de Kerckhoffs :
Un système est considéré commeétant sécurisé de par sa conceptionet non parce que sa conception estinconnue de l'adversaire

Limites :
• Débogage et récupération de log plus difficile. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
• 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.
• 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 fait par le nom ne peut pas être obfusqué.

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

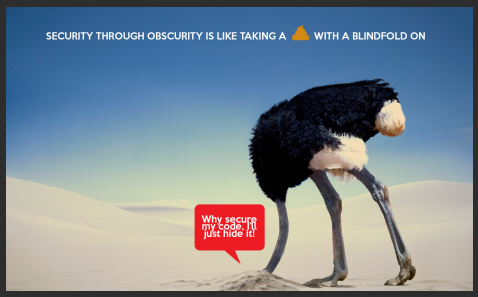
• **Comment s'en prémunir ?**

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



Principe de Kerckhoffs :
Un système est considéré commeétant sécurisé de par sa conceptionet non parce que sa conception estinconnue de l'adversaire

Limites :
• Débogage et récupération de log plus difficile. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
• 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.
• 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 fait par le nom ne peut pas être obfusqué.

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

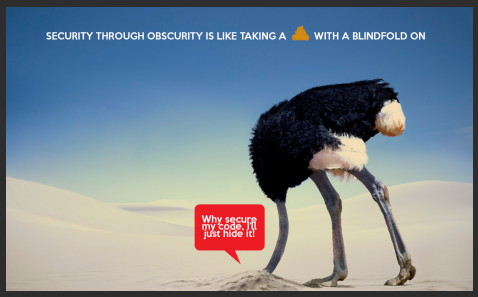
• **Comment s'en prémunir ?**

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



Principe de Kerckhoffs :
Un système est considéré commeétant sécurisé de par sa conceptionet non parce que sa conception estinconnue de l'adversaire

Limites :
• Débogage et récupération de log plus difficile. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
• 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.
• 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 fait par le nom ne peut pas être obfusqué.

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

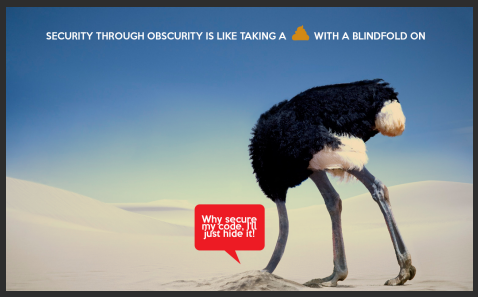
• **Comment s'en prémunir ?**

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



Principe de Kerckhoffs :
Un système est considéré commeétant sécurisé de par sa conceptionet non parce que sa conception estinconnue de l'adversaire

Limites :
• Débogage et récupération de log plus difficile. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
• 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.
• 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 fait par le nom ne peut pas être obfusqué.

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

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 librairies externes non obfuscables