

Programmation orientée sécurité

Olivier Levillain

Agence nationale de la sécurité des systèmes d'information

23 mai 2018

Plan

Généralités

Description des différents cours

Écriture d'un parser MiniPNG

Conclusion

Généralités

Description des différents cours

Écriture d'un parser MiniPNG

Conclusion

Description des modules

Cours sur la programmation sécurisée

- ▶ Programmation orientée sécurité (POS) à l'université Paris Sud
 - ▶ cours optionnel dans un M2
 - ▶ 18 heures (sur 21)
 - ▶ depuis 2014

- ▶ Programmation sécurisée à l'INSA de Rennes
 - ▶ option en 2^e année du cycle d'ingénieur
 - ▶ 26 heures
 - ▶ depuis 2017

Objectifs pédagogiques

À la fin du cours, les étudiants devraient

- ▶ comprendre les enjeux du développement sécurisé
- ▶ connaître les menaces courantes
- ▶ avoir acquis des réflexes et des bonnes pratiques
- ▶ être mieux armés pour intégrer la sécurité au plus tôt dans le développement logiciel
- ▶ *avoir développé un esprit critique pour s'adapter aux nouvelles menaces*

Pré-requis

- ▶ Des bases en programmation sont nécessaires
- ▶ Pour le TP noté, le choix du langage est laissé aux étudiants
- ▶ Des notions d'architecture des ordinateurs sont également utiles
- ▶ Des connaissances plus générales en système aussi

Politique d'évaluation

L'évaluation repose sur

- ▶ un TP noté (individuel)
- ▶ un projet bibliographique (en binôme)
 - ▶ *seulement dans la version longue (INSA) du module*
- ▶ un contrôle écrit

Généralités

Description des différents cours

Écriture d'un parser MiniPNG

Conclusion

Introduction (2h)

Thèmes abordés

- ▶ *présentation du module*
- ▶ enjeux de la sécurité dans le développement logiciel
- ▶ exemples dans divers secteurs d'activités

- ▶ objectif : prise de conscience de la responsabilité du développeur dans la sécurité d'un produit

Cours/TP sur les vulnérabilités classiques (6h)

Modalité

- ▶ alternance de cours et de travaux pratiques
- ▶ étude d'exemples simples de code
- ▶ objectif : détecter et corriger les problèmes

Problèmes abordés

- ▶ injections en tout genre
- ▶ problèmes de logique
- ▶ débordements de tampon
- ▶ *race conditions*
- ▶ *timing attacks* (+ défi)
- ▶ *string format attacks*
- ▶ *directory traversal*

Bonnes pratiques (2h)

Description des premières réponses à apporter

- ▶ conseils classiques
 - ▶ vérifier les entrées utilisateurs
 - ▶ faire attention à la gestion des secrets
 - ▶ etc.
- ▶ présentation de grands principes
 - ▶ réduction du périmètre exposé
 - ▶ défense en profondeur

Conseils sur le bon usage des outils

- ▶ `-Wall -Wextra -Werror`
- ▶ objectif : inculquer des réflexes pour détecter certains problèmes au plus tôt

Quelques éléments de méthodologie

- ▶ lisibilité du code
- ▶ gestion de version
- ▶ tests

Mind your languages et autres TP

Mind your languages (2h)

- ▶ conférence issue des travaux menés à l'ANSSI sur les langages
- ▶ pièges des langages décrits de manière ludique
- ▶ *présentation à RESSI 2015 et supports disponibles*¹

TP sur les débordements de tampon (2h)

- ▶ débordement de tampon dans la pile uniquement
- ▶ 2h ne permettent que d'effleurer le sujet

TP noté sur un *parser* (6h + travail personnel)

1. <https://paperstreet.picty.org/yeye/2014/conf-spw-JaegerL14/>

Cours sur les aspects web (2h)

Nouveauté 2018

- ▶ définition du web (HTTP, HTML, CSS, JS)
- ▶ description du contexte web pour le développeur
 - ▶ modèle sans état a priori
 - ▶ pas de point d'entrée clairement identifié
 - ▶ évolution des applications en milieu hostile
 - ▶ frontières de confiance floues

Présentation de quelques failles

- ▶ questions liées à l'authentification et aux sessions
- ▶ injections variées (SQLi, XSS, LFI)

Attention, ce n'est qu'une introduction modeste au sujet

Projets bibliographiques (4h + travail personnel)

Présentation sur une faille logicielle (ou un mécanisme de défense)

- ▶ recherche d'informations sur une faille
- ▶ présentation de cette démarche de recherche
- ▶ explication de la vulnérabilité et de son impact
- ▶ analyse des réponses à apporter
 - ▶ court et long terme
 - ▶ sans se limiter aux modifications du code

Exemples de sujets :

- ▶ accès non autorisé à des fichiers privilégiés dans OpenSSH (CVE-2011-4327);
- ▶ biais RC4 pour décrypter une session TLS (<http://www.isg.rhul.ac.uk/tls/>);
- ▶ exécution arbitraire de code dans tnftp (CVE-2014-8517).

Généralités

Description des différents cours

Écriture d'un parser MiniPNG

Conclusion

Présentation générale du TP noté

Objectif : écrire un *parser* d'images au format MiniPNG

Présentation générale du TP noté

Objectif : écrire un *parser* d'images au format MiniPNG

Découpage du TP

1. écriture du *parser* de la structure d'images en noir et blanc
2. affichage des images lues
3. extension du *parser* à des images en couleurs

Présentation générale du TP noté

Objectif : écrire un *parser* d'images au format MiniPNG

Découpage du TP

1. écriture du *parser* de la structure d'images en noir et blanc
2. affichage des images lues
3. extension du *parser* à des images en couleurs

Compléments

- ▶ langage de développement assez libre (mais à valider)
- ▶ bien sûr, il faut réfléchir à la sécurité !
- ▶ il faut expliciter les hypothèses faites par rapport à la spécification

Spécification de MiniPNG (1/3)

Une image au format MiniPNG est constituée

- ▶ d'un marqueur Mini-PNG
- ▶ d'une liste de blocs

Spécification de MiniPNG (1/3)

Une image au format MiniPNG est constituée

- ▶ d'un marqueur Mini-PNG
- ▶ d'une liste de blocs

Format d'un bloc

Offset	Nom du champ	Taille du champ
0	Type de bloc	1 octet
1	Longueur du bloc /	4 octets
5	Contenu du bloc	/ octets

Spécification de MiniPNG (2/3)

Une image en noir et blanc contient

- ▶ un bloc d'en-tête H
- ▶ d'éventuels blocs de commentaires C
- ▶ un ou plusieurs blocs de données D

Spécification de MiniPNG (2/3)

Une image en noir et blanc contient

- ▶ un bloc d'en-tête H
- ▶ d'éventuels blocs de commentaires C
- ▶ un ou plusieurs blocs de données D

Bloc H

Offset	Nom du champ	Taille du champ
0	Largeur de l'image	4 octets
4	Hauteur de l'image	4 octets
8	Type de pixels (0 = N&B, 1 = gris...)	1 octet

Spécification de MiniPNG (3/3)

Bloc C

- ▶ une simple chaîne de caractères ASCII affichables

Spécification de MiniPNG (3/3)

Bloc C

- ▶ une simple chaîne de caractères ASCII affichables

Bloc D

- ▶ en N&B, chaque pixel doit être codé sur 1 bit
- ▶ en niveau de gris, chaque pixel est codé sur 1 octet
- ▶ en couleurs, chaque pixel est codé sur 3 octets (RVB)
- ▶ avec une palette, jusqu'à 256 couleurs peuvent être définies

Fourniture de fichiers

Des fichiers *a priori* bons

- ▶ des fichiers avec 1 ou 2 blocs D (et éventuellement des commentaires)
- ▶ un fichier avec un bloc D de plus de 256 octets
- ▶ un fichier N&B de taille 13x7
- ▶ un fichier avec un bloc H à la fin
- ▶ des fichiers en niveau de gris ou en couleur (avec ou sans palette)

Fourniture de fichiers

Des fichiers *a priori* bons

- ▶ des fichiers avec 1 ou 2 blocs D (et éventuellement des commentaires)
- ▶ un fichier avec un bloc D de plus de 256 octets
- ▶ un fichier N&B de taille 13x7
- ▶ un fichier avec un bloc H à la fin
- ▶ des fichiers en niveau de gris ou en couleur (avec ou sans palette)

Des fichiers *a priori* mauvais

- ▶ marqueur invalide
- ▶ absence d'en-tête
- ▶ absence de données
- ▶ trop de données

Discussion / correction

- ▶ Discussion sur les hypothèses manquantes
- ▶ Proposition d'amélioration de la spécification

Discussion / correction

- ▶ Discussion sur les hypothèses manquantes
- ▶ Proposition d'amélioration de la spécification
- ▶ Description au tableau des fonctions à implémenter

Discussion / correction

- ▶ Discussion sur les hypothèses manquantes
- ▶ Proposition d'amélioration de la spécification
- ▶ Description au tableau des fonctions à implémenter
- ▶ Écriture du code de manière simple et lisible

Discussion / correction

- ▶ Discussion sur les hypothèses manquantes
- ▶ Proposition d'amélioration de la spécification

- ▶ Description au tableau des fonctions à implémenter
- ▶ Écriture du code de manière simple et lisible
- ▶ Description d'implémentations vulnérables

Quelques points croustillants (1/2)

Échauffement

- ▶ quelle *endianness* ?
- ▶ quid d'un bloc de plus de 256 octets ?

Quelques points croustillants (1/2)

Échauffement

- ▶ quelle *endianness*?
- ▶ quid d'un bloc de plus de 256 octets?

Sur l'ordre des blocs. Que faire avec...

Quelques points croustillants (1/2)

Échauffement

- ▶ quelle *endianness*?
- ▶ quid d'un bloc de plus de 256 octets?

Sur l'ordre des blocs. Que faire avec...

- ▶ un fichier sans bloc H
 - ▶ un élève avait prévu des valeurs par défaut ?!

Quelques points croustillants (1/2)

Échauffement

- ▶ quelle *endianness*?
- ▶ quid d'un bloc de plus de 256 octets ?

Sur l'ordre des blocs. Que faire avec...

- ▶ un fichier sans bloc H
 - ▶ un élève avait prévu des valeurs par défaut ? !
- ▶ un fichier avec plusieurs blocs H

Quelques points croustillants (1/2)

Échauffement

- ▶ quelle *endianness* ?
- ▶ quid d'un bloc de plus de 256 octets ?

Sur l'ordre des blocs. Que faire avec...

- ▶ un fichier sans bloc H
 - ▶ un élève avait prévu des valeurs par défaut ? !
- ▶ un fichier avec plusieurs blocs H
- ▶ un fichier avec un bloc H après un bloc D

Quelques points croustillants (1/2)

Échauffement

- ▶ quelle *endianness*?
- ▶ quid d'un bloc de plus de 256 octets ?

Sur l'ordre des blocs. Que faire avec...

- ▶ un fichier sans bloc H
 - ▶ un élève avait prévu des valeurs par défaut ? !
- ▶ un fichier avec plusieurs blocs H
- ▶ un fichier avec un bloc H après un bloc D
- ▶ un fichier avec un bloc de type inconnu

Quelques points croustillants

Et le contenu ?

Quelques points croustillants

Et le contenu ?

- ▶ un bloc H fait toujours 9 octets. Pourquoi le vérifier ?

Quelques points croustillants

Et le contenu ?

- ▶ un bloc H fait toujours 9 octets. Pourquoi le vérifier ?
- ▶ quid d'un fichier auquel il manque des données
 - ▶ un élève proposait des pixels à 0 par défaut ? !

Quelques points croustillants

Et le contenu ?

- ▶ un bloc H fait toujours 9 octets. Pourquoi le vérifier ?
- ▶ quid d'un fichier auquel il manque des données
 - ▶ un élève proposait des pixels à 0 par défaut ? !
- ▶ quid d'une longueur négative ?

Quelques points croustillants

Et le contenu ?

- ▶ un bloc H fait toujours 9 octets. Pourquoi le vérifier ?
- ▶ quid d'un fichier auquel il manque des données
 - ▶ un élève proposait des pixels à 0 par défaut ? !
- ▶ quid d'une longueur négative ?
- ▶ que faire d'un commentaire non ASCII ?

Quelques points croustillants

Et le contenu ?

- ▶ un bloc H fait toujours 9 octets. Pourquoi le vérifier ?
- ▶ quid d'un fichier auquel il manque des données
 - ▶ un élève proposait des pixels à 0 par défaut ? !
- ▶ quid d'une longueur négative ?
- ▶ que faire d'un commentaire non ASCII ?
- ▶ quid d'une palette absente ou incomplète ?

Quelques points croustillants

Et le contenu ?

- ▶ un bloc H fait toujours 9 octets. Pourquoi le vérifier ?
- ▶ quid d'un fichier auquel il manque des données
 - ▶ un élève proposait des pixels à 0 par défaut ? !
- ▶ quid d'une longueur négative ?
- ▶ que faire d'un commentaire non ASCII ?
- ▶ quid d'une palette absente ou incomplète ?
- ▶ quid d'une image noir et blanc 13 par 7 ?

Généralités

Description des différents cours

Écriture d'un parser MiniPNG

Conclusion

Conclusion et perspectives

Retour sur les six sessions

- ▶ retours des élèves globalement positifs
- ▶ importance de rappeler *ce qui va sans dire*
- ▶ objectifs principaux : prise de conscience et acquisition de réflexes

Évolutions à venir

- ▶ rendre le TP sur les vulnérabilités classiques plus ludique
- ▶ faire reprendre le TP *buffer overflow* par un rennais ?
- ▶ développer la méthodologie de développement
- ▶ ajouter un TP autour du cours sur le web
- ▶ *Mind your languages* à réduire/supprimer ?
- ▶ séance initiale du TP noté à réduire ?

Questions ?

Merci de votre attention

`olivier.levillain@ssi.gouv.fr`

Les supports de cours, les exemples et les implémentations sont à votre disposition sur simple demande

Session de février-mars 2018 à l'INSA

7 février	8h - 10h	Introduction
	10h15 - 12h15	Cours/TP sur les vulnérabilités classiques
	13h30 - 15h30	Cours/TP sur les vulnérabilités classiques
8 février	8h - 10h	Bonnes pratiques
	10h15 - 12h15	Cours/TP sur les vulnérabilités classiques
21 février	8h - 10h	Cours conférence « <i>Mind your languages</i> »
	10h15 - 12h15	TP sur les dépassements de tampon
22 février	8h - 12h15	TP noté sur un <i>parser</i>
14 mars	8h - 10h	Cours sur les aspects web
	10h15 - 12h15	Retour sur le TP noté
15 mars	8h - 12h15	Soutenances de projet

Exemple de vulnérabilité classique (1/2)

```
#!/bin/sh

PASSWORD="azerty12"

read TYPED_PASSWORD

if [ $TYPED_PASSWORD = $PASSWORD ]; then
    echo Authentication OK
    [...]
else
    echo Authentication Failure
    exit 1
fi
```

Comment s'authentifier sans connaître le mot de passe ?

Exemple de vulnérabilité classique (2/2)

```
#include <stdio.h>

int main () {
    char* s = "toto";
    printf ("%s\n", s);
    s[1] = 'i';
    printf ("%s\n", s);
    return 0;
}
```

Quel est le problème ?

Comment demander au compilateur de nous le signaler ?