Programmation orientée sécurité

Olivier Levillain

Le présent document présente un module intitulé « Programmation orientée sécurité », dont l'objectif est d'intégrer les aspects sécurité liés au développement logiciel. Ce module a déjà été joué à plusieurs reprises, dans des conditions relativement proches :

— sous la forme d'un module de 18 heures dispensé en M2 à l'université Paris Sud (cours « Programmation orientée Sécurité » du parcours « Fondamentaux de l'informatique et ingénierie logicielle »), depuis 2015 ;
— sous la forme d'un module 26 heures dispensé en deuxième année de cycle ingénieur à l'INSA de Rennes (cours 4INFOS8ProSec), depuis 2017.

1 Description

Le cours « Programmation orientée sécurité » vise à décrire les grandes catégories de vulnérabilités logicielles, à détailler leur impact sur la sécurité d'un programme, et à expliquer les moyens de s'en protéger. Au-delà du test et de la correction des failles, proprement dites, l'accent est mis tout au long du cours sur les moyens d'être pro-actif pour assurer la sécurité pendant le développement logiciel.

Le cours couvre à la fois la description théorique des vulnérabilités, des exemples concrets de failles logicielles et la présentation de contre-mesures (spécifiques ou génériques). De plus, le module contient un certain nombre de travaux pratiques, dont un TP noté, ainsi qu'un projet bibliographique.

Note : Bien entendu, en fonction du nombre d'heures disponibles, certains contextes particuliers comme le web ne sont pas abordés en détails. De même, le projet bibliographique n'est proposé que dans la version longue du module. Le reste du document présente la version complète du module.

2 Pré-requis

Des bases en programmation sont nécessaires pour pouvoir suivre ce cours. Pour le TP noté, la connaissance d'un langage en particulier est nécessaire (le choix du langage est laissé aux étudiants, mais doit être validé avec l'enseignant [1]).

Des notions d'architecture des ordinateurs sont également utiles, pour mieux appréhender les aspects liés à la gestion mémoire d'un programme. De même, des connaissances en système peuvent permettre de comprendre les questions liées au déploiement d'une application dans un environnement d'exécution donné ou à la gestion des droits.

3 Objectifs pédagogiques

À la fin du cours, les étudiants devront comprendre les enjeux du développement sécurisé, ainsi que les menaces courantes. Ils auront acquis certains réflexes et bonnes pratiques dans leur manière de développer, afin qu'ils soient capables d'intégrer la sécurité au plus tôt dans le développement logiciel.

De plus, il serait souhaitable que les élèves développent pendant ce cours un esprit critique leur permettant d'adapter leur connaissance aux nouvelles menaces.

1. En pratique, les langages proposés par les étudiants sont C, C++, Java, Python. Dans certaines classes, des étudiants ont également choisi OCaml et Ruby. L'objet de la validation par l'enseignant a surtout pour objectif de garantir qu'il sera possible de compiler et de relire le programme.

4 Politique d'évaluation

L'évaluation se fait sur la base des éléments suivants :

— un TP noté (coefficient 1), commencé pendant une séance de TP, et que les élèves peuvent amender pendant une semaine après la séance. Le TP est réalisé individuellement ;
— un projet bibliographique (coefficient 1), par binôme (ou trinôme). L'objectif est d'analyser une vulnérabilité ou un mécanisme de défense à partir d'un avis de sécurité ou d'un billet de blog. Le projet donne lieu à une soutenance suivie de questions ;
— un contrôle écrit de 2h (coefficient 1), sans ordinateur ni accès aux notes de cours.

5 Emploi du temps détaillé

Pour la session de février-mars 2018 à l'INSA, voici les différents créneaux :

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 « *Mind your languages* »
	10h15 - 12h15	TP sur les dépassements de tampon
22 février	8h - 12h15	TP noté sur un *parser*
14 mars	8h - 10h	Cours sur les aspects web
	10h15 - 12h15	Retour sur le TP noté
15 mars	8h - 12h15	Soutenances de projet

6 Détail des interventions

Introduction

Ce premier cours a pour but de présenter les enjeux de la sécurité dans le développement logiciel. À travers des exemples issus de secteurs d'activité variés (automobile, systèmes industriels, santé, etc.), l'accent est mis sur la responsabilité du développeur dans la sécurité du produit final.

Cours/TP sur les vulnérabilités classiques

Plusieurs créneaux sont dédiés à cette part du module, pour un volume horaire d'environ 6 heures mêlant cours magistral et travaux pratiques. Une quinzaine d'exemples sont soumis à la sagacité des élèves, qui doivent étudier des morceaux de code, y détecter une faille de sécurité, et proposer une manière de corriger le problème repéré. Les exemples traités touchent à des problèmes assez divers, par exemple :

— injection *shell* ;
— erreur de logique dans le traitement d'erreur ;
— *directory traversal* ;
— dépassement de tampon dans la pile ;
— *timing attack*.

Sur ce dernier type d'attaque (un programme vérifiant un mot de passe dont le temps d'exécution dépend du secret), un défi est proposé aux étudiants avec un mot de passe différent (et sans accès aux sources).

Bien que les exemples de code présentés soient simples et un peu artificiel, des exemples plus conséquents sont présentés dans d'autres cours du module.

Note : Pour cette partie du cours, une amélioration prévue est de rendre les exemples un peu plus réalistes et de proposer une plus grande partie du TP sous la forme de défis. Il faudra cependant faire attention à ce que les élèves ne se consacrent pas uniquement aux aspects offensifs.

Bonnes pratiques

Cette intervention a pour objet les premières réponses à apporter aux problèmes présentés. Au-delà des conseils classiques sur les erreurs à ne pas commettre (vérifier les entrées utilisateurs, faire attention à la gestion des secrets, etc.), des grands principes sont présentés (réduction du périmètre exposé, défense en profondeur, etc.).

De plus, des conseils sur le bon usage du compilateur sont présentés. Par exemple, les élèves sont invités à toujours utiliser `-Wall -Wextra -Werror` au minimum dans tous leurs projets en C[2]. L'avantage de ce genre de bonnes pratiques, outre le fait de permettre la détection des problèmes dans le code très tôt dans le cycle de développement, c'est que cette recommandation ne fait perdre que peu de temps, dès lors qu'elle devient un réflexe et qu'elle est appliquée dès le début d'un projet.

Enfin, des éléments de méthodologie plus générale sont présentés (lisibilité du code, gestion de version, tests). Cependant, comme il s'agit d'une présentation assez courte, cette dernière partie pourrait mériter d'être un peu rallongée, éventuellement avec une application dans un TP.

Mind your languages

Dans le cadre des travaux menés à l'ANSSI sur les langages, un article a été rédigé[3] et de nombreuses présentations ont été réalisées autour de la sécurité des langages et des pièges dans lesquels un développeur, même averti, peut tomber.

Ce cours-conférence, présenté de façon ludique, présente des exemples de code surprenants (mais valides), qui peuvent mener à des failles de sécurité exploitables. L'objet de cette intervention n'est *pas* d'élire un langage qui serait *meilleur* que les autres, ni de critiquer les langages utilisés dans les exemples[4]

C'est l'occasion de présenter un certain nombre de vulnérabilités réelles aux étudiants, et d'insister sur l'importance de bien connaître et comprendre le langage qu'on utilise.

TP sur les dépassements de tampon

Il est assez classique, lorsque l'on parle de développement sécurisé et de failles classiques, de s'attarder sur le débordement de tampon *(buffer overflow)*. En effet, cet approfondissement est l'occasion de faire toucher du doigt aux élèves le modèle mémoire utilisé en pratique par les programmes qu'ils écrivent.

Pour cela, le TP consiste notamment à observer les cadres de pile dans `gdb` et à mettre en évidence l'écrasement d'une valeur dans la pile à l'aide d'exemples choisis.

Bien entendu, une séance ne suffit pas à aborder de nombreux aspects liés au débordement de tampon, et le TP se concentre sur les dépassements dans la pile.

TP noté sur un *parser*

Afin de faire prendre conscience aux élèves de la complexité du développement logiciel, il leur est demandé d'implémenter un *parser* pour un format de fichiers, Mini-PNG, inspiré du format d'images classique PNG. La spécification de Mini-PNG est volontairement mal écrite, et inclut de nombreuses ambiguïtés. Les attendus de ce TP sont multiples :

— l'écriture d'un outil fonctionnel dans le langage de leur choix ;
— l'application de bonnes pratiques lors du développement ;
— le développement d'un regard critique sur la spécification pour proposer des améliorations. Les élèves sont en effet encouragés à expliquer les hypothèses qu'ils ont dû faire en codant le *parser*.

Par exemple, il est toujours amusant de voir des élèves décoder un entier 32 bits dans un fichier, sans se soucier que l'ordre des octets ne soit spécifié nulle part !

2. Évidemment, ce conseil est à transposer aux autres langages, en essayant toujours de profiter au maximum des retours des outils de la chaîne de compilation et de test (compilateurs, interpréteurs, analyseurs statiques) pour améliorer la qualité du code.

3. Éric Jaeger, Olivier Levillain, *Mind your Language(s) : A Discussion about Languages and Security*, 35. IEEE Security and Privacy Workshops, SPW (LangSec) 2014, pp.140-151. `https://paperstreet.picty.org/yeye/2014/conf-spw-JaegerL14/`

4. À ce moment du discours, il m'arrive néanmoins d'ajouter une exception concernant JavaScript et PHP...

Il est également toujours étonnant de rencontrer un élève qui propose, dans un cours de développement sécurisé, de s'accommoder d'une image où il manquerait des données (il suffit pourtant de considérer que les octets manquants sont des zéros !).

Ce TP, commencé lors d'une séance de TP en classe, peut ensuite être terminé par les élèves pendant une semaine. La correction se fait en deux partie : d'une part une note calculée de manière relativement automatique sur le comportement du programme face à des fichiers bien ou mal formés, et d'autre part une relecture du code, qui permet de faire des remarques sur le style du code et l'application (ou non) des bonnes pratiques ; cette seconde partie ne fait pas au sens strict l'objet d'une note, mais permet de faire comprendre de manière concrète certaines erreurs classiques, au-delà de leur présentation théorique.

Lorsque c'est possible, il fait ensuite l'objet d'un retour complet, incluant une proposition de correction et la présentation d'erreurs de programmation possibles. Ces erreurs peuvent avoir été constatées dans les rendus des élèves, mais également au sein d'implémentations incorrectes écrites spécialement pour illustrer ce TP.

Enfin, si le temps le permet, une présentation rapide du format PDF est proposée. L'objectif est de montrer que s'ils ont eu des difficultés à implémenter un format simple comme Mini-PNG, dont la spécification fait 2 pages, il serait sans doute délicat de s'attaquer à un format dont la spécification fait plusieurs milliers de page, et qui est également assez mal écrite sur de nombreux points...

Cours sur les aspects web

La sécurité des développement web est évidemment un sujet important aujourd'hui, car le web est présent dans de nombreuses applications *cloud* mais aussi dans les applications mobiles ou comme moyen de communiquer des objets connectés. Il a donc semblé pertinent lors de la dernière session du module d'en parler plus en détails.

Le cours commence par définir le web, en faisant un petit historique sur le protocole HTTP et les différentes technologies impliquées (HTML, CSS, JavaScript). On en déduit les éléments suivants :

— le web repose sur un modèle sans état a priori ;
— le code ne dispose pas de point d'entrée clairement identifié ;
— les applications évoluent en milieu hostile ;
— les frontières de confiance sont floues.

Ensuite, certains problèmes et certaines failles sont décrites, en particulier les questions liées à l'authentification et à la gestion des sessions, les injections variées (injections SQL, XSS, LFI).

Cet ajout récent au module ne fait pas l'objet d'un TP pour l'instant, mais il serait utile d'intégrer une partie pratique à ce cours, afin de faire toucher la sordide réalité de la sécurité des applications web aux étudiants.

Projets bibliographiques

Il est demandé aux élèves de préparer une présentation sur une faille logicielle, réalisée lors de la dernière séance de cours. Les objectifs de ce projet sont les suivants :

— chercher de l'information sur une faille ;
— présenter une démarche de recherche ;
— expliquer la vulnérabilité et son impact ;
— analyser les réponses à apporter (court et long terme), sans se limiter aux modifications du code.

Les projets sont réalisés en binôme et font l'objet d'une présentation de 15-20 minutes, suivie d'une séance de questions. Afin de montrer aux étudiants ce qui est attendu, un exemple de soutenance est présenté lors du cours introductif. L'exemple utilisé est *Heartbleed*, la fameuse faille OpenSSL qui a fait couler beaucoup d'encre en 2014.

Voici quelques 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).

L'évaluation porte sur la prestation à l'oral, mais aussi sur la profondeur de l'analyse proposée et la rigueur du travail effectué.

7 Bibliographie

Aucune lecture n'est nécessaire préalablement à ce cours, mais voici quelques liens utiles pour les étudiants intéressés :

- `oss-security` (`http://www.openwall.com/lists/oss-security/`) est une liste de diffusion sur laquelle sont publiées des vulnérabilités et des correctifs de sécurité concernant des logiciels libres ;
- le site web du CERT-FR (`http://www.cert.ssi.gouv.fr/`) ou la liste de diffusion correspondante permet d'obtenir des informations sur les vulnérabilités existantes, ainsi que de l'information plus large sur l'actualité de la SSI ;
- *Smashing Stack For Fun and Profit*, l'article historique décrivant l'exploitation d'un dépassement de tampon (`http://insecure.org/stf/smashstack.html`) ;
- les études de l'ANSSI sur les langages de programmation : JavaSec (`https://www.ssi.gouv.fr/javasec`) et LaFoSec (`https://www.ssi.gouv.fr/lafosec`).

8 Conclusin et perspectives

Après avoir joué six fois ce module, les retours sont globalement positifs. Tout d'abord, même si certains points présentés peuvent sembler évident au premier abord, il apparaît que les sujets ne sont pas toujours connus, compris, ou assimilés par les élèves.

L'objectif de ce cours est de faire prendre conscience de la difficulté de développer de manière sécurisée et de proposer des éléments de réponse. Il semble essentiel pour arriver à ce résultat de passer par de la pratique et par l'acquisition de bons réflexes.

En quatre ans, le cours a déjà un peu évolué (remplacement du TP noté, ajout d'un cours lié au web, introduction du projet bibliographique). Comme indiqué dans les présentations détaillées du cours, certains aspects pourraient sans doute encore évoluer, pour améliorer le module :

- rendre le TP sur les vulnérabilités classiques plus ludique (sans perdre de vue le côté défensif) ;
- intégrer plus d'aspects liés à la méthodologie de développement dans le cours sur les bonnes pratiques, et inclure ces aspects dans un TP ;
- ajouter une partie pratique autour du cours sur le web.

Note : Les contenus pédagogiques sont disponibles sur demande, et devraient prochainement faire l'objet d'une consolidation pour être partagés de manière plus simple.