CyberEdu : quelques pistes pour enseigner la cybersécurité dans des cours de développement ou de réseau

Olivier Levillain

Télécom SudParis
Association CyberEdu

16 février 2021

Plan

Plan

Qui suis-je ? (1/2)

Olivier Levillain

Qui suis-je ? (1/2)

Olivier Levillain

- @pictyeye sur Twitter
- `https://paperstreet.picty.org/yeye`
- Président de l'association CyberEdu

Qui suis-je ? (1/2)

Olivier Levillain

- @pictyeye sur Twitter
- `https://paperstreet.picty.org/yeye`
- Président de l'association CyberEdu

Parcours

- stage en cryptographie sur une fonction de hachage
- membre du laboratoire « système » de l'ANSSI (2007-2012)
- responsable du laboratoire « réseau » de l'ANSSI (2012-2015)
- responsable du centre de formation de l'ANSSI (2015-2018)
- maître de conférences à Télécom SudParis (2018-)

Qui suis-je ? (2/2)

Recherche

- participation aux travaux sur les mécanismes bas-niveau x86
- études sur les langages de programmation depuis 2007
- travaux sur SSL/TLS (thèse soutenue en 2016)
- travaux sur les *parsers*

Qui suis-je ? (2/2)

Recherche

- participation aux travaux sur les mécanismes bas-niveau x86
- études sur les langages de programmation depuis 2007
- travaux sur SSL/TLS (thèse soutenue en 2016)
- travaux sur les *parsers*

Enseignement

- cryptographie appliquée : fonctions de hachage, cryptanalyse, chiffrement de disque, SSL/TLS
- cours sur la sécurité système Unix
- interventions en réseau et en sécurité des réseaux
- module « Programmation Orienté Sécurité »

CyberEdu en quelques mots

Quelques constats :

- obtenir un niveau de sécurité acceptable est difficile
- un expert en SSI ne peut rien face à une horde de développeurs/administrateurs non sensibilisés
- la sécurité est l'affaire de tous !

CyberEdu en quelques mots

Quelques constats :

- obtenir un niveau de sécurité acceptable est difficile
- un expert en SSI ne peut rien face à une horde de développeurs/administrateurs non sensibilisés
- la sécurité est l'affaire de tous !

Démarche

- inciter et accompagner l'intégration de la SSI dans les formations du supérieur en informatique
- intérêt pédagogique
- intérêt *marketting*

CyberEdu : quelques exemples concrets

CyberEdu : quelques exemples concrets

- Un administrateur réseau doit savoir que l'on peut changer l'adresse MAC d'une carte réseau

CyberEdu : quelques exemples concrets

- Un administrateur réseau doit savoir que l'on peut changer l'adresse MAC d'une carte réseau
- Un développeur C doit savoir ce qu'est un *buffer overflow*

CyberEdu : quelques exemples concrets

- Un administrateur réseau doit savoir que l'on peut changer l'adresse MAC d'une carte réseau
- Un développeur C doit savoir ce qu'est un *buffer overflow*
- Un admin sys doit savoir comment sont stockés les mots de passe

CyberEdu : quelques exemples concrets

- Un administrateur réseau doit savoir que l'on peut changer l'adresse MAC d'une carte réseau
- Un développeur C doit savoir ce qu'est un *buffer overflow*
- Un admin sys doit savoir comment sont stockés les mots de passe
- Un développeur web doit savoir ce qu'est une injection SQL

CyberEdu : quelques exemples concrets

- Un administrateur réseau doit savoir que l'on peut changer l'adresse MAC d'une carte réseau
- Un développeur C doit savoir ce qu'est un *buffer overflow*
- Un admin sys doit savoir comment sont stockés les mots de passe
- Un développeur web doit savoir ce qu'est une injection SQL
- Un informaticien doit savoir ce qu'est un certificat électronique

CyberEdu : la démarche et les actions

2013 – 2016 : lancement de CyberEdu par l'ANSSI

- supports pédagogiques (licence CC-BY)
- organisation de colloques de 3 jours à l'ANSSI au profit des formateurs

CyberEdu : la démarche et les actions

2013 – 2016 : lancement de CyberEdu par l'ANSSI

- supports pédagogiques (licence CC-BY)
- organisation de colloques de 3 jours à l'ANSSI au profit des formateurs

Depuis 2016 : l'association CyberEdu

- plusieurs colloques proposés (Poitiers, Colmar, Nantes)
- partenariat avec l'Afpa pour introduire de la cybersécurité dans plusieurs titres professionnels
- maintenance et développement des documents
- forum d'échanges entre spécialistes et non spécialistes de la sécurité
- label « CyberEdu » (91 formations référencées à ce jour)

Plan

MS17-010 : la menace des rançongiciels

En mai et juin 2017, deux attaques très médiatisées sur des rançongiciels

- exploitation d'une vulnérabilité critique dans Windows
- ... sur un service qui ne devrait pas être exposé
- ... pour lequel un correctif est disponible depuis mars

MS17-010 : la menace des rançongiciels

En mai et juin 2017, deux attaques très médiatisées sur des rançongiciels

- exploitation d'une vulnérabilité critique dans Windows
- ... sur un service qui ne devrait pas être exposé
- ... pour lequel un correctif est disponible depuis mars
- pourquoi la sécurité semble-t-elle un échec ?

Une voiture connectée

Charlie Miller et Chris Valasek (BlackHat2015) : prise de contrôle à distance d'une Jeep

- Cause : de nombreux services non sécurisés en écoute sur internet

Une voiture connectée

Charlie Miller et Chris Valasek (BlackHat2015) : prise de contrôle à distance d'une Jeep

- Cause : de nombreux services non sécurisés en écoute sur internet
- Combien de voitures (avions, usines...) reposent de manière critique sur du logiciel pour fonctionner ?

Santé et sécurité

Les *pacemakers* ont aujourd'hui des interfaces sans fil pour permettre un suivi en temps réel des patients

- Dick Cheney (vice-président des USA sous Georges W. Bush) a été convaincu par la NSA de désactiver ces interfaces
- Quelques publications académiques sur le sujet

Santé et sécurité

Les *pacemakers* ont aujourd'hui des interfaces sans fil pour permettre un suivi en temps réel des patients

- Dick Cheney (vice-président des USA sous Georges W. Bush) a été convaincu par la NSA de désactiver ces interfaces
- Quelques publications académiques sur le sujet
- Scénario utilisé dans une série américaine

Santé et sécurité

Les *pacemakers* ont aujourd'hui des interfaces sans fil pour permettre un suivi en temps réel des patients

- Dick Cheney (vice-président des USA sous Georges W. Bush) a été convaincu par la NSA de désactiver ces interfaces
- Quelques publications académiques sur le sujet
- Scénario utilisé dans une série américaine

L'informatique s'imisce de plus en plus dans le monde *réel*

La sécurité du numérique : un sujet omniprésent

Que nous apprennent ces exemples ?

- les systèmes informatiques...
- ... sont omniprésents
- ... sont complexes
- ... ont un impact sur le monde physique
- ... et sur des vies humaines

La sécurité du numérique : un sujet omniprésent

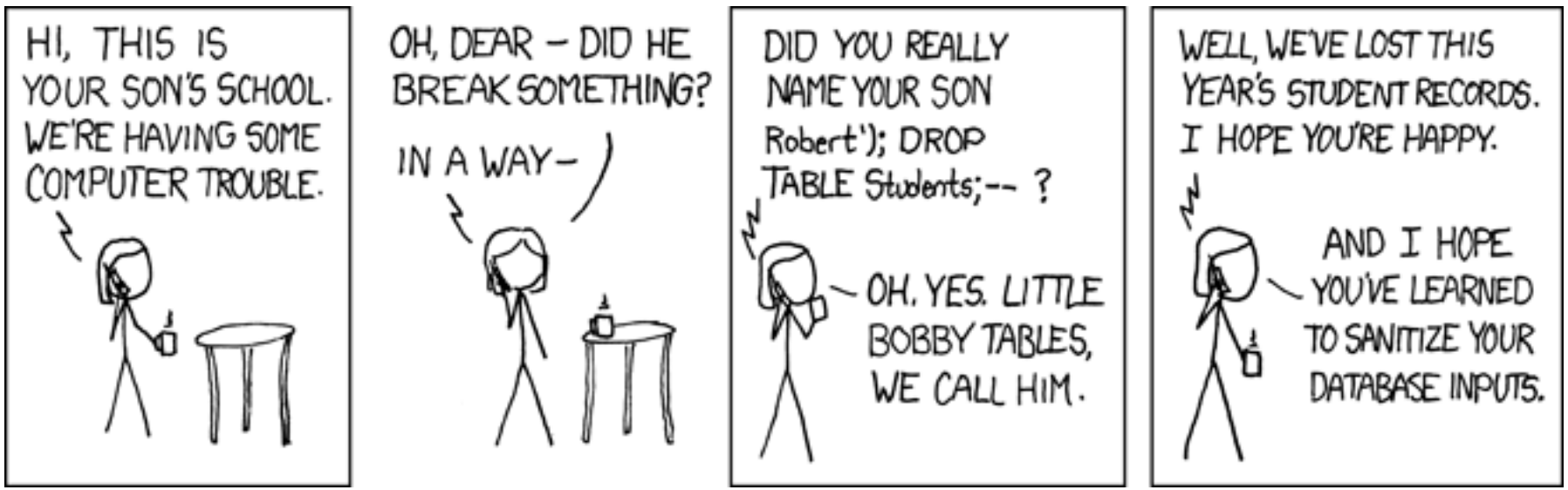
Que nous apprennent ces exemples ?

- les systèmes informatiques...
- ... sont omniprésents
- ... sont complexes
- ... ont un impact sur le monde physique
- ... et sur des vies humaines

Au-delà des spécialistes, il faut former et responsabiliser les acteurs des systèmes d'information

Plan

Connaissez-vous Bobby Tables ?



Source : `http://xkcd.com/327`

SQL Injections 101

Cas d'école

- une application web prenant en entrée
  - `$USER` le nom de l'utilisateur
  - `$PASS` son mot de passe

SQL Injections 101

Cas d'école

- une application web prenant en entrée
  - `$USER` le nom de l'utilisateur
  - `$PASS` son mot de passe
- la vérification est faite auprès d'une base de données SQL

```
$query = 'SELECT * FROM user ' .
         'WHERE username = "' . $USER . '" '
         'AND password = "' . $PASS . '"';
$result = sql_query (query);
if ($result) { /* Authentication OK */ }
```

SQL Injections 101

Cas d'école

- une application web prenant en entrée
  - `$USER` le nom de l'utilisateur
  - `$PASS` son mot de passe
- la vérification est faite auprès d'une base de données SQL

```
$query = 'SELECT * FROM user ' .
         'WHERE username = "' . $USER . '" '
         'AND password = "' . $PASS . '"';
$result = sql_query (query);
if ($result) { /* Authentication OK */ }
```

Explication de texte

- dès que la requête renvoie un résultat, on passe l'authentification
- que se passe-t-il si `$PASS` contient des guillemets ?

Réflexions sur les injections

Concept généralisable

- dès qu'un programme/script utilise une chaîne de caractères pour transmettre une structure complexe
- il y a possibilité de confusion si les caractères décrivant la structure (ici les guillemets) peuvent être contrôlés par l'attaquant

Réflexions sur les injections

Concept généralisable

- dès qu'un programme/script utilise une chaîne de caractères pour transmettre une structure complexe
- il y a possibilité de confusion si les caractères décrivant la structure (ici les guillemets) peuvent être contrôlés par l'attaquant

Contre-mesures classiques

- échapper tous les caractères de structure

Réflexions sur les injections

Concept généralisable

- dès qu'un programme/script utilise une chaîne de caractères pour transmettre une structure complexe
- il y a possibilité de confusion si les caractères décrivant la structure (ici les guillemets) peuvent être contrôlés par l'attaquant

Contre-mesures classiques

- échapper tous les caractères de structure
  - l'approche en liste noire ne marche généralement pas

Réflexions sur les injections

Concept généralisable

- dès qu'un programme/script utilise une chaîne de caractères pour transmettre une structure complexe
- il y a possibilité de confusion si les caractères décrivant la structure (ici les guillemets) peuvent être contrôlés par l'attaquant

Contre-mesures classiques

- échapper tous les caractères de structure
  - l'approche en liste noire ne marche généralement pas
- conserver la structure lors du passage (requêtes préparées)

Un cas d'école de *Buffer Overflow*

```
void f(int i, char c) {
    char buf[10];
    for (int j=0; j<i; j++)
        buf[j] = c;
    buf[i] = 0;
    puts(buf)
}
```

Un cas d'école de *Buffer Overflow*

```
void f(int i, char c) {
    char buf[10];
    for (int j=0; j<i; j++)
        buf[j] = c;
    buf[i] = 0;
    puts(buf)
}
```

Quelques remarques

- `f` affiche la chaîne de `i` caractères, tous égaux à `c`
- que fait `f(4, 'A')` ?
- que fait `f(20, 'A')` ?
- que peut faire un attaquant ?

Conséquences d'un débordement de tampon

Les conséquences dépendent

- de la fonction vulnérable,
- de l'architecture sous-jacente
- de la manière dont le programme a été construit

Conséquences d'un débordement de tampon

Les conséquences dépendent

- de la fonction vulnérable,
- de l'architecture sous-jacente
- de la manière dont le programme a été construit

Concrètement, l'attaquant peut

- planter le programme
- écraser des variables locales
- dérouter le flot de contrôle et exécuter du code de son choix

Jeu du chat et de la souris

- Exécution de code arbitraire dans la pile

Jeu du chat et de la souris

- Exécution de code arbitraire dans la pile
- Contre-mesure : la non exécution de la pile

Jeu du chat et de la souris

- Exécution de code arbitraire dans la pile
- Contre-mesure : la non exécution de la pile
- Le ROP

Jeu du chat et de la souris

- Exécution de code arbitraire dans la pile
- Contre-mesure : la non exécution de la pile
- Le ROP
- Contre-mesure : la randomisation

Jeu du chat et de la souris

- Exécution de code arbitraire dans la pile
- Contre-mesure : la non exécution de la pile
- Le ROP
- Contre-mesure : la randomisation
- Limites d'une randomisation partielle, d'une fuite d'information ou d'un `fork()` à répétition

Jeu du chat et de la souris

- Exécution de code arbitraire dans la pile
- Contre-mesure : la non exécution de la pile
- Le ROP
- Contre-mesure : la randomisation
- Limites d'une randomisation partielle, d'une fuite d'information ou d'un `fork()` à répétition
- Contre-mesure : les canaris

Jeu du chat et de la souris

- Exécution de code arbitraire dans la pile
- Contre-mesure : la non exécution de la pile
- Le ROP
- Contre-mesure : la randomisation
- Limites d'une randomisation partielle, d'une fuite d'information ou d'un `fork()` à répétition
- Contre-mesure : les canaris
- Perspectives : JOP, débordements dans le tas...

Failles du web : quelques mots sur le web

Caractéristique du web

- modèle client-serveur
- pas d'état a priori (mais il y a des *cookies*)
- pas de point d'entrée clairement identifié
- communication en clair par défaut (mais il y a TLS)
- grande complexité (langages, architecture)
- milieu hostile
  - des entrées utilisateurs partout
  - beaucoup de choses se passent sur le client
  - un flôt de contrôle incontrôlable
- frontières de confiances floues

Failles du web : XSS

Au-delà de l'injection SQL, le web connaît les attaques de type *cross-site scripting* consistent à injecter des contenus HTML ou JavaScript via des entrées utilisateurs non filtrées

- encore un nouveau type d'injection liée à une interprétation dans un contexte différent

Failles du web : XSS

Au-delà de l'injection SQL, le web connaît les attaques de type *cross-site scripting* consistent à injecter des contenus HTML ou JavaScript via des entrées utilisateurs non filtrées

- encore un nouveau type d'injection liée à une interprétation dans un contexte différent
- *Reflected XSS* : inclusion dans une page d'un paramètre reçu dans l'URL

Failles du web : XSS

Au-delà de l'injection SQL, le web connaît les attaques de type *cross-site scripting* consistent à injecter des contenus HTML ou JavaScript via des entrées utilisateurs non filtrées

- encore un nouveau type d'injection liée à une interprétation dans un contexte différent
- *Reflected XSS* : inclusion dans une page d'un paramètre reçu dans l'URL
- *Stored XSS* : stockage d'une entrée utilisateur dans une base de données, puis affichage non filtré de ces données
  - exemple : commentaires dans un blog
  - exemple : champs dans un certificat X.509 utilisé pour se connecter à un point d'accès Wifi, ensuite affiché dans une console d'administration

Failles du web : LFI

Exemple issu de la page Wikipédia :

```
<?php
    if ( isset( $_GET['language'] ) ) {
        include( $_GET['language'] . '.php' );
    }
?>
```

- C'est une bête injection PHP à partir d'une variable d'URL
- Le développeur fait l'hypothèse (plutôt hardie) que `language` ne peut être définie que par les valeurs du formulaire...

Plan

Nettoyage des entrées non filtrées

Plusieurs conséquences aux entrées utilisateur non filtrées

- les injections SQL
- les injections *shell*
- les injections diverses et variées
- les débordements de tampon
  - en lecture (*Heartbleed*) : fuite d'information
  - en écriture : modification d'une variable locale
  - en écriture : détournement du flot de contrôle

Comment contrer les injections ? (1/2)

- Interdire les caractères *spéciaux*
  - exemples en PHP pour les injection SQL :

Comment contrer les injections ? (1/2)

- Interdire les caractères *spéciaux*
  - exemples en PHP pour les injection SQL :
  - `mysql_escape_string` (obsolète)

Comment contrer les injections ? (1/2)

- Interdire les caractères *spéciaux*
  - exemples en PHP pour les injection SQL :
  - `mysql_escape_string` (obsolète)
  - `addslashes` (ne couvre que les guillemets et les apostrophes)

Comment contrer les injections ? (1/2)

- Interdire les caractères *spéciaux*
  - exemples en PHP pour les injection SQL :
  - `mysql_escape_string` (obsolète)
  - `addslashes` (ne couvre que les guillemets et les apostrophes)
  - `mysqli_real_escape_string` (couvre-t-elle réellement tous les cas, y compris les encodages alternatifs comme `\u0022` ?)

Comment contrer les injections ? (1/2)

- Interdire les caractères *spéciaux*
  - exemples en PHP pour les injection SQL :
  - `mysql_escape_string` (obsolète)
  - `addslashes` (ne couvre que les guillemets et les apostrophes)
  - `mysqli_real_escape_string` (couvre-t-elle réellement tous les cas, y compris les encodages alternatifs comme `\u0022` ?)
  - la liste noire peut marcher, mais il y a mieux

Comment contrer les injections ? (1/2)

- Interdire les caractères *spéciaux*
  - exemples en PHP pour les injection SQL :
  - `mysql_escape_string` (obsolète)
  - `addslashes` (ne couvre que les guillemets et les apostrophes)
  - `mysqli_real_escape_string` (couvre-t-elle réellement tous les cas, y compris les encodages alternatifs comme `\u0022` ?)
  - la liste noire peut marcher, mais il y a mieux

- Utiliser les *prepared statements*, qui permettent de conserver la structure d'une requête
  - pour SQL, on trouve des fonctions `prepare()` et `bind_param()`
  - en *shell*, il faut éviter `system` et `popen` (qui appellent un nouvel interpréteur) et préférer appeler directement `exec*`

Architecture logicielle : gestion de privilèges

Chaque logiciel (ou partie du logiciel) doit tourner avec les privilèges minimaux

- un serveur web qui tourne en `root` est une hérésie
- il faut éviter qu'un programme privilégié fasse des traitements complexes sur des données venant de l'utilisateur
- parfois, cela nécessite de repenser profondément l'architecture
- Exemple : `postfix`

Architecture logicielle : isolation/cloisonnement des processus

En fonction des systèmes d'exploitation, il existe de nombreux mécanismes pour restreindre les capacités d'un processus [1]

1. Certains exemples sont spécifiques à Linux

Architecture logicielle : isolation/cloisonnement des processus

En fonction des systèmes d'exploitation, il existe de nombreux mécanismes pour restreindre les capacités d'un processus [1]

- `chroot` pour restreindre la visibilité sur le système de fichiers
- les (*net namespace*) pour limiter les capacités réseau
- divers mécanismes pour limiter les capacités d'un processus
  - `rlimit` (limitations sur les ressources)
  - `seccomp` (restriction des appels système)
  - SELinux, AppArmor ou RBAC GRsec

Ils peuvent participer à la sécurité du logiciel !

1. Certains exemples sont spécifiques à Linux

Quelques réponses concernant le web

Éléments de réponse

- lorsque c'est possible, ne pas utiliser directement les entrées utilisateur
- utiliser les *prepared statements* ou des outils préservant la structure
- *échapper* les chaînes de caractères avant utilisation
  - l'action à effectuer **dépend du contexte d'utilisation**, qui peut être multiple pour une même chaîne !
- utiliser pour cela un moteur de *templates* fourni par votre *framework*
- de manière générale, utiliser un *framework* éprouvé et à jour !

Quelques réponses concernant le web

Éléments de réponse

- lorsque c'est possible, ne pas utiliser directement les entrées utilisateur
- utiliser les *prepared statements* ou des outils préservant la structure
- *échapper* les chaînes de caractères avant utilisation
  - l'action à effectuer **dépend du contexte d'utilisation**, qui peut être multiple pour une même chaîne !
- utiliser pour cela un moteur de *templates* fourni par votre *framework*
- de manière générale, utiliser un *framework* éprouvé et à jour !

Défense en profondeur

- interdire ou retreindre les scripts *(Content Security Policies)*
- protéger les cookies (une cible de choix des XSS)

Connaissance du langage et du contexte

Connaissance du langage

- il existe de nombreux pièges dans les langages de programmation
- certains langages offrent des garanties de sûreté ou de sécurité
- il vaut mieux faire sobre et simple
- des constructions complexes seront difficiles à maintenir et à relire
- un exercice intéressant est de faire relire du code aux étudiants (idéalement le leur, après quelques mois en jachère !)

Connaissance du contexte et des bonnes pratiques

- Top 10 de l'OWASP
- Guides de bonnes pratiques pour un langage

De la bonne utilisation des outils

Les compilateurs modernes permettent de déceler de nombreuses erreurs de programmation

- en C, un projet étudiant ne passant pas `-Wall -Wextra -Werror` devrait avoir 0/20 ?
- dans tous les langages, on peut passer des *linter* pour vérifier des bonnes pratiques de base (`pylint` en Python)
- il existe des outils d'analyse statique pour aller plus loin (`mypy` en Python par exemple)

Il est important d'acquérir ces réflexes tôt, afin qu'ils ne soient pas perçus comme des freins au développement

Un peu de méthodologie

Afin de former les développeurs de demain, il semble important de leur donner le goût du test

- tests unitaires
- tests fonctionnels
- tests négatifs (il faut vérifier que ce qui doit échouer échoue en pratique)

On peut même rendre cela plus ludique

- intégration continue dans les forges logicielles
- suivi de la couverture de code
- introduction au *Test-Driven Development*

Plan

Ressources sur les langages

Quelques liens pour montrer les pièges dans les langages de programmation ou pour donner des conseils

- *Mind your languages* : `http://paperstreet.picty.org/yeye/tag/mind-your-languages.html`
- un livre sur des exemples surprenants en Java : `http://www.javapuzzlers.com/`
- le Top 10 de l'OWASP : `https://www.owasp.org/index.php/Top_10-2017_Top_10`
- écriture de scripts shell robustes : `https://www.davidpashley.com/articles/writing-robust-shell-scripts/`
- article dans Linux Magazine sur les avertissements C : `https://connect.ed-diamond.com/GNU-Linux-Magazine/GLMFHS-097/Les-options-pour-faire-du-compilateur-C-un-ami-qui-vous-v`

Ressources du module « Programmation Orientée Sécurité »

Cursus présenté à RESSI 2018 (`http://paperstreet.picty.org/yeye/2018/conf-ressi-Levillain18/`

- vulnérabilités classiques
- bonnes pratiques
- éléments sur le web
- méthodologie de développement (git, GitLab, TDD)
- TP (noté) sur les *parsers*
- TP (noté) sur le déverminage d'un projet en Python
- projets bibliographiques

Tout est disponible sous licence CC-BY sur demande

Questions/discussion sur la partie développement

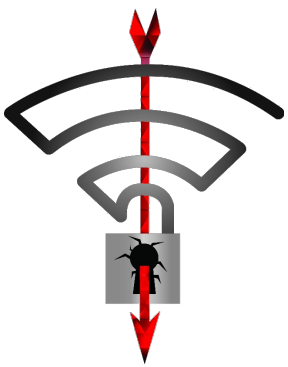
Plan

Plan

Réseaux sans-fil

WiFi

- WEP (*Wired Equivalent Privacy*) complètement cassé
  - WEP repose sur RC4, qui est de plus mal utilisé
- WPA (*WiFi Protected Access*)
  - diverses attaques sur WPA et WPA2
  - KRACK (*Key Reinstallation Attacks*, 2017)

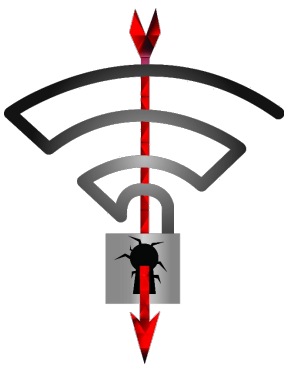


  - ambiguïté dans la spécification WPA2
  - que faire en cas de réception en double d'un certain message ?
  - l'impact varie en fonction des piles

Réseaux sans-fil

WiFi

- WEP (*Wired Equivalent Privacy*) complètement cassé
  - WEP repose sur RC4, qui est de plus mal utilisé
- WPA (*WiFi Protected Access*)
  - diverses attaques sur WPA et WPA2
  - KRACK (*Key Reinstallation Attacks*, 2017)



- ambiguïté dans la spécification WPA2
- que faire en cas de réception en double d'un certain message ?
- l'impact varie en fonction des piles

Là encore, la sécurité est loin d'être parfaite

- importance des mises à jour
- nécessité de la défense en profondeur

Couche IP

Exemples d'attaques au niveau IP

- *ARP spoofing*
  - usurpation d'une adresse sur un réseau local
- usurpation d'adresse IP source
  - utilisé pour mener des attaques en déni de service
  - par réflexion et par amplification

Couche IP

Exemples d'attaques au niveau IP

- *ARP spoofing*
  - usurpation d'une adresse sur un réseau local
- usurpation d'adresse IP source
  - utilisé pour mener des attaques en déni de service
  - par réflexion et par amplification

Mécanismes de sécurité

- BCP 38
  - filtrer les paquets au plus proche de la source
- MPLS
  - utile surtout pour la disponibilité
  - dispositif reposant sur la « bonne volonté » des infrastructures
- IPsec
  - établissement de tunnels chiffrés et intègres
  - efficace, mais mise en œuvre relativement complexe

Couche TCP

Vulnérabilités

- pas de protection en confidentialité ou en intégrité
- besoin de connaître/deviner les numéros de séquence pour s'insérer dans une connexion
  - le *3-way handshake garantit* que l'adresse source est joignable
  - difficile pour un attaquant en dehors du chemin

Couche TCP

Vulnérabilités

- pas de protection en confidentialité ou en intégrité
- besoin de connaître/deviner les numéros de séquence pour s'insérer dans une connexion
  - le *3-way handshake garantit* que l'adresse source est joignable
  - difficile pour un attaquant en dehors du chemin

Contre-mesure classique pour protéger les couches applicatives : SSL/TLS

Résolution de noms de domaine avec DNS (1/2)

Problèmes de sécurité liés à DNS

- intégrité des réponses (empoisonnement de cache)
- source de dénis de service distribués

Causes

- usurpation d'IP source facilitée par l'utilisation d'UDP
- requêtes DNS en grande partie prédictible

Résolution de noms de domaine avec DNS (2/2)

DNSSEC

- \+ protection en intégrité des réponses (sauf le dernier lien)
- − réponses plus longues (amplification des DDoS)

Résolution de noms de domaine avec DNS (2/2)

DNSSEC

- \+ protection en intégrité des réponses (sauf le dernier lien)
- − réponses plus longues (amplification des DDoS)

Limiter les réponses d'un serveur DNS

- \+ atténuation des effets d'un DDoS
- − certains empoisonnements de cache deviennent plus facile

Résolution de noms de domaine avec DNS (2/2)

DNSSEC

- \+ protection en intégrité des réponses (sauf le dernier lien)
- − réponses plus longues (amplification des DDoS)

Limiter les réponses d'un serveur DNS

- \+ atténuation des effets d'un DDoS
- − certains empoisonnements de cache deviennent plus facile

TCP (usurpation d'IP source plus difficile)

- \+ protection (légère) en intégrité
- \+ atténuation des effets d'un DDoS
- − nombreux *firewalls* ou serveurs mal configurés

Résolution de noms de domaine avec DNS (2/2)

DNSSEC

- \+ protection en intégrité des réponses (sauf le dernier lien)
- − réponses plus longues (amplification des DDoS)

Limiter les réponses d'un serveur DNS

- \+ atténuation des effets d'un DDoS
- − certains empoisonnements de cache deviennent plus facile

TCP (usurpation d'IP source plus difficile)

- \+ protection (légère) en intégrité
- \+ atténuation des effets d'un DDoS
- − nombreux *firewalls* ou serveurs mal configurés

DNS over HTTPS (DoH)

- \+ résoud les problèmes
- − atteintes possibles à la vie privée
- − service violant les modèles de sécurité classique en entreprise

Résolution de noms de domaine avec DNS (2/2)

DNSSEC

- \+ protection en intégrité des réponses (sauf le dernier lien)
- − réponses plus longues (amplification des DDoS)

Limiter les réponses d'un serveur DNS

- \+ atténuation des effets d'un DDoS
- − certains empoisonnements de cache deviennent plus facile

TCP (usurpation d'IP source plus difficile)

- \+ protection (légère) en intégrité
- \+ atténuation des effets d'un DDoS
- − nombreux *firewalls* ou serveurs mal configurés

DNS over HTTPS (DoH)

- \+ résoud les problèmes
- − atteintes possibles à la vie privée
- − service violant les modèles de sécurité classique en entreprise

Réparer le DNS est un problème d'ingénierie complexe !

Résolution de noms de domaine avec DNS (2/2)

DNSSEC

- \+ protection en intégrité des réponses (sauf le dernier lien)
- − réponses plus longues (amplification des DDoS)

Limiter les réponses d'un serveur DNS

- \+ atténuation des effets d'un DDoS
- − certains empoisonnements de cache deviennent plus facile

TCP (usurpation d'IP source plus difficile)

- \+ protection (légère) en intégrité
- \+ atténuation des effets d'un DDoS
- − nombreux *firewalls* ou serveurs mal configurés

DNS over HTTPS (DoH)

- \+ résoud les problèmes
- − atteintes possibles à la vie privée
- − service violant les modèles de sécurité classique en entreprise

Réparer le DNS est un problème d'ingénierie complexe !
Et il faut aussi penser aux ressources nécessaires à ces solutions.

Un peu de recul sur l'attaquant réseau

Un attaquant réseau ayant accès au canal de communication peut :

- écouter du trafic
- émettre de trafic en usurpant des champs source/identité
- modifier, insérer, détruire de messages

L'idée souvent répandue est que les messages réseau ne peuvent être échangés que par des acteurs légitimes.

C'est évidemment un mythe... Les échanges réseau n'ont rien de magique.

Un peu de recul sur les manipulations de réseau

Usurpation d'identité

- ARP, adresses MAC
- IP, adresses IP
- SMTP, émetteur
- ...

Empoisonnement / détournement

- BGP, préfixes IP annoncés
- ARP, correspondance MAC/IP
- DNS, correspondance IP/noms de domaine
- Google, quel serait l'impact d'un empoisonnement du cache de recherche ?
- ...

Enseignements

De manière générale, il ne faut pas compter sur les couches basses pour protéger efficacement la confidentialité et l'intégrité des données

Il ne faut pas faire confiance *a priori* à son correspondant : les paquets qui arrivent sur votre interface se ressemblent tous !

En général, les protocoles utilisés sont connus et documentés... ou il est possible de faire du *reverse engineering*

La sécurité peut être apportée par

- l'architecture, lorsque vous maîtrisez les locaux
- la cryptographie

Plan

Rôle d'un *firewall*

Un *firewall* a pour objectif de filtrer les paquets entre deux réseaux.

Positionnement

- entre l'extérieur et l'intérieur
- entre les différentes zones internes
  - DMZ (zone démilitarisée)
  - zones de serveurs internes
  - desserte (zones clients)
- sur les postes et serveurs (*firewall* local)

Face à un paquet, plusieurs verdicts possibles

- laisser passer (`ACCEPT`)
- rejeter avec une erreur (`REJECT`)
- ignorer silencieusement (`DROP`)

Avec ou sans état ?

Comment décider le verdict à appliquer ?

- en analysant les champs des différentes couches du paquet
- en consultant des informations stockées précédemment

Avec ou sans état ?

Comment décider le verdict à appliquer ?

- en analysant les champs des différentes couches du paquet
- en consultant des informations stockées précédemment

- *Firewall* simple sans état
  - lecture des en-têtes IP et TCP

Avec ou sans état ?

Comment décider le verdict à appliquer ?

- en analysant les champs des différentes couches du paquet
- en consultant des informations stockées précédemment

- *Firewall* simple sans état
  - lecture des en-têtes IP et TCP
- *Firewall* classique avec état
  - lecture des en-têtes IP et TCP
  - stockage des connexions observées
  - capacité de *suivre* une connexion

Avec ou sans état ?

Comment décider le verdict à appliquer ?

- en analysant les champs des différentes couches du paquet
- en consultant des informations stockées précédemment

- *Firewall* simple sans état
  - lecture des en-têtes IP et TCP
- *Firewall* classique avec état
  - lecture des en-têtes IP et TCP
  - stockage des connexions observées
  - capacité de *suivre* une connexion
- DPI (*Deep Packet Inspection*)
  - analyse des couches applicatives

Avec ou sans état ?

Comment décider le verdict à appliquer ?

- en analysant les champs des différentes couches du paquet
- en consultant des informations stockées précédemment

- *Firewall* simple sans état
  - lecture des en-têtes IP et TCP
- *Firewall* classique avec état
  - lecture des en-têtes IP et TCP
  - stockage des connexions observées
  - capacité de *suivre* une connexion
- DPI (*Deep Packet Inspection*)
  - analyse des couches applicatives

Plus l'analyse est complexe, plus le risque de déni de service est important (saturation de l'état, temps de traitement trop long)...

Avec ou sans état ?

Comment décider le verdict à appliquer ?

- en analysant les champs des différentes couches du paquet
- en consultant des informations stockées précédemment

- *Firewall* simple sans état
  - lecture des en-têtes IP et TCP
- *Firewall* classique avec état
  - lecture des en-têtes IP et TCP
  - stockage des connexions observées
  - capacité de *suivre* une connexion
- DPI (*Deep Packet Inspection*)
  - analyse des couches applicatives

Plus l'analyse est complexe, plus le risque de déni de service est important (saturation de l'état, temps de traitement trop long)...

et plus on doit se poser de questions sur le traitement des données sensibles (données personnelles, etc.)

Serveurs mandataires

Exemple classique : le *proxy* web

- les clients locaux se connectent au *proxy*
- le *proxy* relaie la requête et répond au client
- possibilité de mettre en cache les résultats

Autres exemples

- *reverse proxy* web
- relais mail SMTP

Serveurs mandataires

Exemple classique : le *proxy* web

- les clients locaux se connectent au *proxy*
- le *proxy* relaie la requête et répond au client
- possibilité de mettre en cache les résultats

Autres exemples

- *reverse proxy* web
- relais mail SMTP

Intérêt pour la sécurité de tels points de passages obligés

- détecter des attaques au niveau applicatif
- journaliser les interactions pour analyse d'un incident a posteriori

Qu'est-ce qu'une architecture ?

Un ensemble comprenant

- des machines : postes clients, serveurs
- des équipements réseaux : switches, routeurs
- des équipements de sécurité : firewalls, terminaisons VPNs, antivirus

Une architecture *sécurisée*

- vise à fournir un ensemble de services
- en prenant en compte certaines menaces
  - défiguration de sites web
  - compromission de machines
  - fuite de données...

Flux et politique de filtrage

Matrice des flux

Avant toute mise en place d'un pare-feu, il est nécessaire d'élaborer une matrice des flux qui décrit les échanges réseaux légitimes entre les différents équipements.

Cette matrice des flux sert ensuite à écrire la politique de filtrage

Dans les cas classiques :

1. tout trafic est interdit
2. le trafic des clients vers Internet est autorisé
3. le trafic d'Internet vers la DMZ est autorisé

Un mot sur les *appliances* et équipements divers

- Ce ne sont pas des boîtes magiques

Un mot sur les *appliances* et équipements divers

- Ce ne sont pas des boîtes magiques

- Quels sont les protocoles supportés ?
- Quels sont les flux légitimes ?

Un mot sur les *appliances* et équipements divers

- Ce ne sont pas des boîtes magiques
- Quels sont les protocoles supportés ?
- Quels sont les flux légitimes ?
- Ces questions semblent naturelles pour des serveurs, des passerelles ou des *firewalls*

Un mot sur les *appliances* et équipements divers

- Ce ne sont pas des boîtes magiques

- Quels sont les protocoles supportés ?
- Quels sont les flux légitimes ?

- Ces questions semblent naturelles pour des serveurs, des passerelles ou des *firewalls*

- Mais pour une imprimante réseau ?
- Ou une caméra IP ?
- Quelle doit être leur configuration réseau ?
- Est-ce normal que de tels périphériques émettent du *spam* ?

Configuration des équipements

- Restreindre la myriade de protocoles généralement supportés
- Modifier les mots de passe par défaut
- Supposer que les services sont vulnérables
- Restreindre les clients potentiels (quitte à filtrer ou faire passer les requêtes par un proxy)

- Pour certains équipements, seule l'architecture permet de sécuriser l'ensemble

Plan

Objectifs de la cryptologie

La **cryptologie** est la science du secret

- la **cryptographie** consiste à concevoir des mécanismes pour protéger l'information
- la **cryptanalyse** cherche à casser ces mécanismes

Propriétés de sécurité auxquelles contribue la crypto

- confidentialité
- intégrité
- authentification

Chiffrement symétrique

Première application de la crypto : garantir la confidentialité d'un échange

Chiffrement symétrique

Première application de la crypto : garantir la confidentialité d'un échange

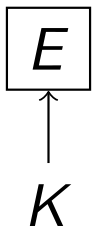
- une clé (symétrique) K

K

Chiffrement symétrique

Première application de la crypto : garantir la confidentialité d'un échange

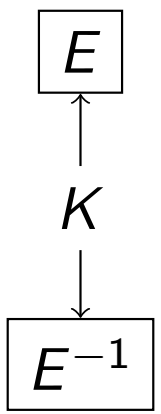
- une clé (symétrique) K
- un algorithme de chiffrement E paramétré par une clé



Chiffrement symétrique

Première application de la crypto : garantir la confidentialité d'un échange

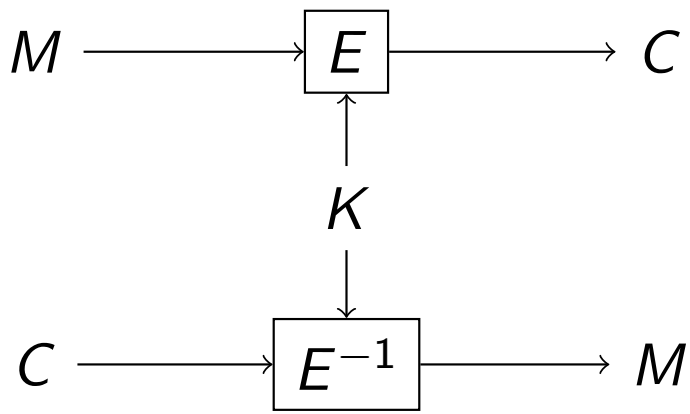
- une clé (symétrique) K
- un algorithme de chiffrement E paramétré par une clé
- un algorithme de déchiffrement E^{-1} paramétré par la même clé



Chiffrement symétrique

Première application de la crypto : garantir la confidentialité d'un échange

- une clé (symétrique) K
- un algorithme de chiffrement E paramétré par une clé
- un algorithme de déchiffrement E^{-1} paramétré par la même clé



Chiffrement symétrique

Première application de la crypto : garantir la confidentialité d'un échange

- on utilise une clé (symétrique) K
- un algorithme de chiffrement E paramétré par une clé
- un algorithme de déchiffrement E^{-1} paramétré par la même clé

Propriétés

- étant donné un message M, $E_K(M)$ ne doit rien révéler de M
- $E_K^{-1}(E_K(M)) = M$

Exemples d'algorithmes de chiffrement symétrique

- ~~RC4~~, ~~DES~~, ~~3DES~~
- AES (128, 192 ou 256 bits)
- Chacha20 (128 ou 256 bits)

Principe de Kerckhoffs

Principe classique en crypto

- le secret doit dépendre d'un paramètre de petite taille (la clé)
- et pas de l'algorithme !

Apport du chiffrement symétrique

- une fois l'algorithme choisi
- il *suffit* d'échanger de manière sécurisée la clé

Ce n'est évidemment que le début !

Il est généralement inutile ou contre-productif de protéger un flux en confidentialité mais pas en intégrité

- il existe des attaques crypto qui sont alors possibles (*padding oracle*
- il existe des attaques réseau sur des paquets chiffrés sans intégrité

Au-delà des aspects symétriques, on peut vouloir aborder des aspects asymétriques (signature, certificats, PKI, etc.)

L'essentiel, pour des populations de non-spécialistes en crypto, est

- comprendr les objets et leurs propriétés
- comprendre les hypothèses pour bien utiliser les primitives
- utiliser des constructions éprouvées et ne pas chercher à inventer

Plan

Quelques références

Une note de l'ANSSI intéressante sur le sujet :

- `https://www.ssi.gouv.fr/guide/definition-dune-architecture-de-passerelle-dinterconnexion-securisee/`

Les planches présentées ici sont tirées d'un cours d'introduction à la sécurité réseau

- planches et sources disponibles sur demande

Quelques ressources en vrac partageables

- analyses de trace avec Wireshark
- cours sur HTTP, SMTP, SSH
- TP sur SMTP

Questions/discussion sur la partie réseau

Plan

Système / réseau

- Matériel, OS, logiciels certifiés ANSSI
  - `https://www.ssi.gouv.fr/entreprise/produits-certifies/`
- Référentiel intéressant : socle interministériel de logiciels libres
  - `https://sill.etalab.gouv.fr/fr/software`
- Mises à jours systématiques/automatiques/monitorées et globales (BIOS/firmwares, OS, pilotes, logiciels, bibliothèques...)
- Administration sécurisée et durcissement
  - `https://www.ssi.gouv.fr/administration/guide/securiser-ladministration-des-systemes-dinformation/`
  - `http://www.ssi.gouv.fr/guide/recommandations-de-securite-relatives-a-un-systeme-gnulinux/`
  - `https://www.ssi.gouv.fr/guide/recommandations-de-securite-relatives-a-active-directory/`
- *Cloud*
  - `http://www.ssi.gouv.fr/entreprise/bonnes-pratiques/externalisation/`

Bonnes pratiques

- Guide d'hygiène (renforcer la sécurité de son SI en 42 mesures)
  - `https://www.ssi.gouv.fr/guide/guide-dhygiene-informatique/`
- Guide CPME : les bonnes pratiques de l'informatique
  - `https://www.ssi.gouv.fr/guide/guide-des-bonnes-pratiques-de-linformatique/`
- Trousseaux de mdp (éviter SaaS sur logiciel propriétaire et cloud avec données à l'étranger)
- Authentification avec plusieurs facteurs
- PRA/PCA
  - sauvegarde/archivage sur site externe/cloud
  - redondance

Compléments

- État de la menace
  - `http://www.cert.ssi.gouv.fr/`
- Droit
  - Loi Godfrain et cadre légal
  - RGPD / NIS
- En cas d'incident
  - `https://www.ssi.gouv.fr/en-cas-dincident/`
- Prestataires d'audit certifiés
  - `http://www.ssi.gouv.fr/entreprise/qualifications/prestataires-de-services-de-confiance-qualifies/prestataires-daudit-de-la-securite-des-systemes-dinformation-passi-quali`

Plan

Conclusion

- La sécurité n'est pas uniquement l'affaire des spécialistes
- CyberEdu a pour objectif d'injecter de la sécurité dans les formations du supérieur en informatique (et au-delà)
- Vous pouvez adhérer
  - à titre personnel
  - au nom de votre établissement
- Plus d'information sur `https://www.cyberedu.fr`

Quelques mots sur la labellisation (1/2)

Un des objectifs de l'association est d'encourager l'intégration de contenus de cybersécurité dans les formations du supérieur en informatique

- le label concerne les formations longues
- il référence les formations qui « tissent » des contenus de cybersécurité au long des cours
- il ne s'applique pas aux formations de spécialistes (pour ça, il faut regarder SecNum*edu* sur le site de l'ANSSI)

Quelques mots sur la labellisation (2/2)

Procédure de labellisation

- remplissage d'un formulaire
  - informations administratives
  - présentation des contenus de sécurité et de la démarche de tissage
- discussion possible avec des membres de l'association pour accompagner la labellisation
- passage du dossier en GT Labellisation
- en cas de validation, le label est octroyé pour 3 ans
- publication sur le site de l'association

Questions ?

Merci de votre attention

olivier.levillain@cyberedu.fr

https://www.cyberedu.fr