INFO-F201 – Systèmes d'exploitation 1
Projet 2
Le super serveur et les attaques DoS

Année académique 2012–2013

Énoncé

Dans ce projet, nous vous demandons d'implémenter une architecture client-serveur qui mette en évidence une des problématiques majeures de la sécurité du web : *les attaques par déni de service (DoS)* [1].
Le but principal de ces attaques est de rendre les services proposés par les serveurs inaccessibles, ou plus particulièrement de diminuer leur fiabilité.

Les DoS sont des attaques dont on ne connaît pas une solution générique, en effet, selon le type de l'attaque et son niveau, des solutions précises (et parfois coûteuses) doivent être mises en place.

Le but du projet est d'implémenter une simulation d'une attaque DoS à l'aide de l'architecture client-serveur. Les deux composantes principales de votre conception peuvent être décrites comme suit :

Le serveur : exécute une boucle infinie pour accepter toutes les demandes de connexion venant des clients. Une fois lancé, le fils généré par l'application serveur génère un nombre aléatoire (entre `1` et `3`) et l'envoie au client. Selon la réponse du client (`0` ou `1` cf. infra), on distingue deux cas de figure. Si la réponse vaut `1`, le serveur renvoie un message ‘‘`J'ai confiance en toi :-)`’’. Sinon, si la réponse vaut `0`, le serveur génère un autre nombre et le renvoie. Si par contre, après `5` essaies, le client répond toujours par `0`, le serveur renvoie un message ‘‘`Tu es un attaquant !!`’’ et ferme la connexion.

Le client : quant à lui, génère `5` fils itérativement [2], chacun de ces fils a un comportement déterminé par une valeur aléatoire générée par le père. En effet, le père génère aléatoirement un nombre `0` ou `1`, sur lequel il se base pour attribuer un comportement à son fils. On distingue donc deux comportements différents pour les processus fils.

1. client gentil (*niceClient* si le nombre choisit par le père vaut `0`) : c'est le client qui tente de communiquer en toute honnêteté avec le serveur. Il génère un nombre aléatoire propre à lui (entre `1` et `3`), compare le nombre reçu par le serveur, et renvoie au serveur la valeur du test.
2. client byzantin (*byzantineClient* si le nombre choisit par le père vaut `1`) : c'est le client implémentant la procédure d'attaque et qui fera tout pour mettre en cause la fiabilité du serveur. Ce client malicieux répond `0` à toute réponse du serveur, il n'a donc pas besoin de générer un nombre vu qu'aucune comparaison n'est envisageable.

1. **D**enial **o**f **S**ervice attacks
2. La génération ne se fait pas d'une façon parallèle. Un processus fils est généré à la fois.

Toutefois, nous avons laissé volontairement quelques points flous concernant l'architecture des programmes client-serveur afin de vous laisser une certaine liberté de réflexion.

1 Modalités de remises et de réalisations

Les programmes doivent être écrits en C. Les appels systèmes devront être uniquement ceux vus durant les séances de travaux pratiques. Vous ne serez donc pas autorisés à utiliser des API's ou d'autres framework supplémentaires.
Nous vous demandons de nous fournir un listing complet de votre implantation. Ce dernier devra être accompagné de deux diagrammes d'activité (un par entité) décrivant votre architecture, vos choix concernant les points laissés flous et un justificatif de ces derniers.

Certains critères d'évaluation de vos projets seront la clarté, la lisibilité, la propreté d'implémentation et l'efficacité bien entendu. Commentez donc **intelligemment** votre code et implémentez ce qu'on vous demande de façon claire et précise.

Votre projet sera remis en version papier **et** électronique.

Consignes pour la remise du projet

À respecter scrupuleusement !

1. Votre projet doit indiquer **votre nom** et **votre numéro de groupe** (par exemple en commentaires au début de chaque fichier).
2. Votre projet doit être **dactylographié**. Les projets écrits à la main ne seront **pas corrigés** (0/10).
3. Votre code doit être **commenté**.
4. Si votre code ne s'exécute pas, votre projet ne sera **pas corrigé** (0/10).
5. Tout cas de triche sera sanctionné d'un 0/10
6. Votre projet sera remis en deux versions : **papier et électronique**.
7. Vous devez respecter les modalités de remise suivante :
   - Date de remise : **le lundi 17 décembre 2012**
   - Lieu : **au Secrétariat « étudiants » du Département d'Informatique, local 2N8.104**
   - Heure : **avant 15h30**
   - La version électronique est à envoyer à l'adresse suivante : `myoussef@ulb.ac.be`, en mettant comme titre : `info-f-201-2012-Projet2`. **Attention !** les mails sont traités automatiquement, tout non-respect de cette règle signifie que votre projet n'ai pas reçu.

   Après 15h30, les projets seront considérés comme **en retard**, et vous aurez **0 points** sur votre note finale de projet.