PROJET SCIENTIFIQUE COLLECTIF

Rapport de mi-parcours
INF07

10 janvier 2016

Pierrick Allègre
Gustavo Castro
Clément Durand
Felipe Garcia
Alexandre Harry
Francisco Ribeiro Eckhardt Serpa
Pierre-Alexandre Thomas
Guillaume Vizier
Tuteur : Thomas Clausen – Coordinateur : Franck Nielsen
Cadre : Capitaine Angélique Aubois

ÉCOLE POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

TABLE DES MATIÈRES

PRÉSENTATION SUCCINCTE DU SUJET

Nous avions initialement deux objectifs. Dans un premier temps, nous voulions mettre en évidence la possibilité d'obtenir des informations censées être sécurisées. Dans un deuxième temps, nous devions utiliser l'expérience acquise dans la phase « hacking » afin de créer, sûrement sous forme d'application, une méthode pour mieux protéger les données personnelles contenues par les smartphones. Aucun des deux objectifs n'a été totalement réalisé jusqu'à maintenant. Cela nous oblige à revoir nos exigences à la baisse par rapport au projet initial. Cela avait cependant été anticipé, lorsque nous nous étions rendu compte de la complexité du sujet et surtout de sa constante actualisation. Il semble plus raisonnable de concentrer nos efforts sur la partie mise en évidence des failles, qui représentait déjà le cœur du projet. Cette actualisation de nos objectifs nous permettra de mieux comprendre les détails du problème, plutôt que de se précipiter et essayer de corriger une faille mal comprise.

Objectif de la première partie d'année

1
LE POINT SUR NOS RÉSULTATS

1.1 Mise en place des moyens de communication

Nous avions décidé d'utiliser les moyens de communication suivants :

Git : Pour le développement collaboratif de nos programmes

Slack : Pour utiliser un moyen de communication instantanée, plus polyvalent qu'IRC. En effet, nous l'utilisons aussi pour surveiller l'état des dépôts `git`. De plus, cela nous permet d'éviter Facebook, qui en plus d'être peu pratique, est totalement imperméable au concept même de sécurité.

Trello : Qui permet de clarifier l'état actuel du projet en termes de tâches à accomplir.

FramaDate : Pour permettre une meilleure organisation, notamment trouver une plage horaire compatible avec les disponibilités de tous, incluant le tuteur et notre coordinatrice DFHM.

Nous avons constaté que des logiciels adaptés permettent une organisation claire, et que la difficulté venait plutôt d'un tout autre côté.

1.2 Compréhension du réseau

Nous avons commencé notre projet par une phase de recherche, notamment au sujet des différents protocoles existants. Nous avons décidé alors de nous concentrer sur les protocoles suivants :

DNS : Ce protocole donnant aux utilisateurs les adresses des sites qu'ils veulent consulter, il semble propice à une éventuelle exploitation par un tiers pour réaliser une attaque, et il était donc pertinent pour nous de l'étudier.

TCP : Ce protocole permettant un échange suivi entre deux entités du réseau, il est utilisé notamment comme support du protocole HTTP, servant à la transmission de pages web.

Notre utilisation de `dig` nous a permis de mieux comprendre l'architecture du réseau de l'École polytechnique, ainsi que le fonctionnement du protocole DNS.

1.3 DNS poisoning

Après cette phase de recherche sur les différents protocoles, et les réseaux en général, nous avons commencé à nous intéresser plus spécifiquement au DNS poisoning. Nous avons découpé le travail en deux parties : une partie observation, et une partie action. Nous avons donc écrit avec `scapy` (`python`) deux programmes :

- L'un qui analyse le trafic, détecte les requêtes DNS, et affiche les résultats, avec des informations sur le paquet (donc la requête).
- L'autre répond à chaque requête DNS par un paquet DNS fabriqué à la main.

Ces deux programmes sont à l'heure actuelle opérationnels, au sens où lorsqu'ils tournent et qu'un utilisateur du réseau sur lequel nous sommes envoie une requête pour consulter un site internet, notre programme simule une réponse du serveur DNS destinée à envoyer l'utilisateur sur une page web choisie par le programme. Cela permettrait par exemple à l'attaquant de rediriger l'utilisateur sur une fausse page d'identification destinée à récupérer ses identifiants.

Nous avons cependant rencontré un certain nombre de difficultés, notamment :

- Détecter un paquet DNS ne nous a pas posé de réel problème, mais il fallait détecter plus précisément les requêtes DNS, ainsi que récupérer les adresses IP, ports, etc. pour pouvoir créer une réponse adaptée. Les recherches que nous avions faites sur les différents protocoles nous ont permis d'identifier assez facilement les parties d'un paquet IP qui identifiaient les requêtes et réponses DNS, ainsi que celles qui contenaient leurs données.
- Nous réussissions à envoyer un paquet DNS, mais celui-ci semblait n'apparaître nulle part comme s'il n'était pas reçu par la cible. Une recherche plus poussée nous a permis de constater qu'il apparaissait en fait comme non-valide (*broken packet*) pour `wireshark` (qui était le seul de nos logiciels à le voir) : la checksum était fausse, et comme c'est l'élément qui assure l'ordinateur de l'intégrité du paquet, il était naturellement refusé par l'utilisateur. Nous avons donc modifié la façon dont nous créons le paquet DNS réponse : `scapy` calcule automatiquement les checksums, mais ne les recalcule pas en cas de modification du contenu du paquet, c'est pourquoi nous avons simplement créé nos paquets à partir de rien plutôt que de modifier des paquets copiés.
- La commande `dig` ne reconnaissait pas notre paquet réponse. Une théorie quant à la cause de ce problème fut l'absence d'autorité dans les données de nos premiers paquets réponse. Nous avons donc ajouté les autorités, serveurs de l'École polytechnique, dans la section "autorités" du paquet réponse, section a priori nécessaire pour que l'utilisateur accepte une réponse. Nous avons vérifié par ailleurs que le paquet était accepté par `wireshark` qui en effet ne signale plus de paquet brisé.
- Cela n'ayant pas résolu le problème, nous avons connecté nos ordinateurs sur un réseau Wi-Fi personnel (différent de celui de l'École polytechnique), et avons fait attention à interroger nos ordinateurs personnels, en passant par la bonne carte réseau. Nous avons eu alors des résultats positifs.

Le fait d'utiliser des réseaux personnels permet tout d'abord d'éviter de cibler ou écouter des ordinateurs extérieurs à notre groupe, ce qui serait illégal et ne serait pas non plus cohérent avec nos objectifs.

- La rapidité des serveurs DNS est un réel problème, auquel nous ne nous sommes pas encore attelés. En effet, pour réussir l'attaque, il faut répondre avant les vrais serveurs DNS : la première réponse est la réponse enregistrée. Il se trouve que nos premiers ordinateurs de test étaient extrêmement lents, ce qui nous a permis de constater ce problème. En utilisant des ordinateurs plus performants nous nous sommes rapprochés des résultats espérés, suffisamment pour avoir un programme fonctionnel. En revanche, dans nos phases de test, nous interrogeons un serveur DNS qui « n'existe pas » ce qui permet de ne pas avoir ce problème de temps de réponse et de se concentrer sur d'autres éléments.

1.4 Interruption d'une communication TCP établie

Pour assurer l'efficacité de l'attaque précédente, il faut faire en sorte qu'il y ait des requêtes DNS. Pour cela, il faut interrompre les connexions TCP existantes. Nous avons écrit un programme, toujours avec `Scapy`, qui envoie un paquet demandant la fin de la communication au demandeur (paquet FIN).

Dans le cas général, une connexion TCP commence par un three-way handshake (il existe aussi l'ouverture simultanée, mais elle est beaucoup moins fréquente) : le client envoie un paquet SYN (synchronize) au serveur ; le serveur répond avec un paquet SYN+ACK (acknowledgement), signalant qu'il a bien reçu la demande de connexion et qu'il est prêt à louvrir de son côté ; enfin, le client renvoie un paquet ACK et la connexion est finalement établie. Les paquets portent aussi des numéros de séquence, qui sont importants pour éviter toute confusion puisque la connexion d'un même client peut se fermer et rouvrir en succession rapide ou alors être interrompue avec perte de mémoire (il faut la rétablir à partir de zéro).

Après l'établissement, client et serveur échangent des données jusqu'à ce que l'un des deux demande de mettre un terme à la connexion, lorsqu'ils n'ont plus de données à envoyer. D'abord, nous avions l'idée d'imposer cette opération prématurément en envoyant un paquet FIN. Notre programme analyse le trafic, filtre les paquets TCP et, pour les paquets SYN trouvés, renvoie un paquet FIN fabriqué à la main.

De même que pour les paquets DNS, la détection des paquets TCP-SYN, avec les connaissances de sa structure, et la fabrication des paquets réponse, non pas par modification des paquets reçus, mais par création de nouveaux paquets avec `Scapy` pour éviter des problèmes comme celui de la somme de contrôle (checksum), ne nous ont pas posé de grandes difficultés. Cependant, ce programme est trop lent pour réellement interrompre la communication : la vraie réponse arrive toujours avant la nôtre, qui est invariablement ignorée. Nous envisageons donc deux possibilités de travail :

- Réécrire ce programme dans un langage plus bas-niveau, afin d'améliorer le temps de

réaction de notre programme, voire le faire plus vite que les serveurs : nous avons choisi le langage C.

- Anticiper les réponses suivantes : s'il nous est impossible de prendre de vitesse la première réponse, nous allons donc essayer de prendre la place de la deuxième, troisième ou énième réponse. Cela nous force à analyser plusieurs échanges de paquets TCP, pour comprendre et être en mesure de prédire les incrémentations futures. En effet, à chaque échange de paquets, les numéros de séquence sont incrémentés dans les paquets retournés. Si nous sommes capables de prédire ses valeurs après n échanges, nous avons résolu notre problème : même si notre réponse est plus lente, nous pourrons la minuter afin qu'elle s'ajuste à la bonne position.

C'est sur ces points que nous allons travailler dans les semaines à venir.

2
ACTUALISATION DES OBJECTIFS

- Accélérer le programme pour le DNS poisoning
- Accélérer le programme TCP et anticiper sur les réponses suivantes
- Comment bloquer les attaques ci-dessus ?

2.1 Accélérer le programme pour le DNS poisoning

Comme cela a été mentionné dans la première partie, la rapidité de notre réponse lors du DNS poisoning est un problème. En effet, les serveurs DNS sont beaucoup plus performants que nos ordinateurs et renvoient la réponse avant que nous ayons pu envoyer la notre, la cible ne se contentant que de la première réponse, l'attaque n'aboutit pas.

Il y a plusieurs pistes pour palier à ce problème, la première est évidemment d'utiliser des machines plus puissantes, nous avons eu de bons résultats en explorant cette piste et nous allons continuer dans cette voie. Néanmoins, dans le cadre de ce projet nous souhaitions aussi mettre en évidence la facilité d'une attaque sur les communications, c'est pourquoi nous pensons qu'il y a d'autres moyens d'accélérer notre réponse, qui ne passe pas forcément par une amélioration de notre matériel, mais plutôt par une amélioration du code.

En effet, nous avons choisi d'utiliser le langage Python ainsi que la bibliothèque Scapy, qui est un langage de relativement haut-niveau. Nous pensons qu'en essayant avec un langage de plus bas niveau, nous pourrions gagner un temps suffisant pour permettre à presque n'importe quelle machine de pratiquer une attaque de type DNS Poisoning. La piste à explorer serait alors de réécrire notre langage en C pour profiter du temps de réaction que fournit un langage de bas niveau et ainsi renvoyer la réponse avant le serveur DNS.

Étant donné le retard que nous avons pris sur nos objectifs initiaux, et comme nous avons une version fonctionnelle de notre programme, il est probable que nous mettions de côté, du moins pour le moment, cette partie-là et avancions sur l'interruption de la connexion TCP qui n'est pas tout à fait fonctionnelle pour le moment. De plus nous allons commencer à avancer sur la deuxième partie du projet qui est la partie sécurisation, qui risque elle aussi de prendre un peu de temps.

En parallèle, nous allons aussi nous poser la question d'éventuels tests sur des machines utilisant Android pour savoir si nous concentrons la partie sécurisation sur cet OS, afin d'essayer de fournir un produit final qui protégerait ce système contre une attaque de type DNS Poisoning.

2.2 Accélérer le programme TCP

Comme nous avons décrit dans la Section 1.4, dans notre objectif initial d'interrompre une communication TCP déjà établie, nous avons constaté quelques difficultés liées à la vitesse de réponse de notre programme : notre réponse arrive après la réponse du serveur. Divers facteurs affectent ce problème : la vitesse de l'ordinateur, notre localisation par rapport à la cible et la rapidité de réponse de notre algorithme. Ces deux premiers facteurs liés au hardware sont difficiles à contrôler pour nous et jusqu'à maintenant nous sommes plus concentrés sur l'amélioration de notre programme en terme de rapidité d'exécution.

Pour les raisons décrites ci-dessus et les problèmes que nous avons rencontrés, nous avons décidé d'actualiser notre objectif initial d'interrompre une connexion TCP déjà établie et au lieu de ça notre objectif final est maintenant de concevoir une protection contre une attaque du type DNS poisoning, ou au moins d'empêcher que la connexion TCP soit interrompue fallacieusement. Ce nouvel objectif est bien en adéquation avec notre objectif final de sécuriser un portable Android (mieux protéger les données personnelles contenues à l'intérieur du smartphone).

Ce nouveau but vise à sécuriser les échanges de données pour faire en sorte que la conversation ne soit pas interrompue et alors empêcher qu'un tiers récupère les données échangées tout au long de la communication par une attaque Denial of Service qui interromprait la connexion TCP.

Malgré ce changement d'objectif, nous allons quand même nous concentrer aussi en perfectionner notre programme d'interruption de code TCP en l'écrivant dans un langage plus bas-niveau comme C pour ainsi essayer d'anticiper les réponses à venir.

2.3 Comment bloquer les attaques ci-dessus ?

Notre but principal, dans cette deuxième moitié d'année, est de contrer les attaques précédentes. Nous commencerons à travailler sur ce sujet, bien qu'il faille que les attaques sus-mentionnées arrivent à leur fin : modifier la connexion entre les deux systèmes.

Concernant l'attaque TCP, une façon simple de protéger l'utilisateur des attaques est mettre tout le réseau (celui que les deux utilisateurs sont en train d'utiliser) dans un mode non-promiscuous. Le souci avec cette solution est que, pour la mettre en œuvre, il faut être (ou demander à) l'administrateur du réseau pour changer le mode pour le non-promiscuous. Ce qui n'est pas toujours le cas : on n'a pas nécessairement l'accès d'administrateur dans le réseau Wi-Fi qu'on utilise.

Une solution possible est de vérifier s'il existe ou non un type de mode non-promiscuous pour la carte réseau dans chaque ordinateur ou smartphone. Il s'agit d'une solution assez simple,

car lorsque vous êtes en mode non-promiscuous, personne sauf l'émetteur et le récepteur d'un paquet peuvent le voir sur la carte réseau de leur ordinateur. Nous allons chercher d'autres options, qui seront plus complexe : lorsque nous sommes victime de l'attaque TCP, la réponse que l'émetteur initial reçoit est une réponse fausse, qui paraît provenir du récepteur. Il n'est pas trivial de voir que la personne qui a envoyé le message n'est pas le récepteur.

Deuxièmement, pour l'attaque DNS Poisoning, nous pouvons utiliser la même idée : comme cette attaque repose aussi sur une interférence dans la transmission de paquets entre deux utilisateurs, nous pouvons utiliser les mêmes idées (sécuriser soit le réseau avec le mode non-promiscuous, soit la carte réseau de chaque appareil).

Pour le DNS Poisoning, la solution la plus connue s'appelle DNSSEC, mais cette solution doit être mise en place par l'administrateur du domaine. Nous revenons donc à la même problématique que pour contrer l'attaque TCP.

Il nous faut travailler sur ce sujet pour décider quelle solution est la meilleure. Nous souhaitons volontairement nous passer de cryptographie pour cette phase du projet.

CONCLUSION

Nous avons partiellement atteint nos objectifs : nos outils fonctionnent, mais ne sont pour l'instant pas assez efficaces. Cette partie vient donc s'ajouter aux objectifs de la deuxième partie, ce qui les rend trop ambitieux et nous force à les revoir à la baisse, bien que nous soyons nombreux dans le groupe.

Notre programme pour le deuxième semestre reste ambitieux, et nous espérons avoir davantage de sessions de travail les lundis après-midi, seules heures où nous sommes théoriquement tous disponibles pour travailler sur le Projet Scientifique Collectif tous ensemble.→***Nous espérons que notre tuteur ne sera pas tenu de dispenser des cours tous les lundis, afin que nous puissions profiter de son expertise.***

Ce deuxième semestre présentera cependant des particularités d'organisation, puisque notre coordinatrice DFHM est actuellement en stage, et le restera une bonne partie du temps qu'il nous reste. La communication se fera donc uniquement par mail. De plus, le groupe sera scindé en deux : une partie optimisera les algorithmes conçus pendant cette première phase, pendant que l'autre partie du groupe se chargera de trouver un moyen de contrer les outils fonctionnant à l'heure actuelle.

FIN