# Déverrouillage d'Android en simulant un clavier/souris

Antoine Cervoise et Julien Reitzel
antoine.cervoise@gmail.com
julien@reitzel.info

**Résumé.** Depuis Android Honeycomb (3.0), il est possible de brancher un périphérique USB sur un terminal Android, à condition que l'équipement le supporte au niveau hardware. Ceci introduit un nouveau vecteur d'attaque sur l'écran de verrouillage : l'attaque par force brute ou par dictionnaire de ces codes en simulant un clavier ou une souris avec un Arduino. Le code du projet est disponible sur GitHub : https://github.com/cervoise/Hardware-Bruteforce-Framework-2.

## 1 Introduction

### 1.1 Verrouillage téléphones et tablettes Android

Les téléphones et tablettes Android peuvent être verrouillés de différentes manières. Les plus communes sont le code pin, le mot de passe et le schéma de verrouillage. Cependant les possibilités varient en fonction de la version du système d'exploitation de Google et de la surcouche constructeur. On trouve, entre autres : Empreintes digitales, Reconnaissance faciale, Reconnaissance vocale et faciale, Knock Code (spécifique à LG)...

### 1.2 Attaque de téléphones et tablettes Android

Différentes attaques existent pour déverrouiller un téléphone Android. Côté logiciel il y a, par exemple, l'utilisation du *mode debug*, du *ClockworkMod recovery*. Côté matériel l'utilisation du JTAG pour récupérer la RAM ou bien la réalisation d'une attaque *Cold Boot* avec *Frost*. Un robot simulant les doigts d'un humain est aussi capable d'attaquer les codes PIN (Black Hat Arsenal USA 2013). Enfin la divination peut être utilisée notamment en analysant les traces de doigts sur le téléphone afin de réduire le nombre de codes possibles et de tester manuellement les possibilités restantes.

### 1.3 Utilisation de l'OTG

Pour notre attaque, nous allons exploiter la possibilité d'utiliser un clavier ou une souris externe sur Android. La couche matérielle supportant cela est l'USB On-The-Go (OTG). Il s'agit d'une extension de la norme USB permettant d'utiliser un Smartphone comme un maître et non plus comme un esclave. Cela permet notamment l'utilisation de clé USB ou de clavier/souris sur les terminaux. Cette norme nécessite que le système d'exploitation supporte le mode hôte (sous Android c'est le cas depuis la version Honeycomb dédiée aux tablettes). Enfin, le matériel doit gérer l'OTG. Bien que ceci tende à se généraliser, certains modèles récents ne supportent pas cette norme.

## 2 Hardware-Bruteforce-Framework-2

### 2.1 Fonctionnement

Le fonctionnement du framework est simple, celui-ci simule un clavier ou une souris afin de réaliser une attaque par dictionnaire ou bien par force brute.

### 2.2 Montage hardware

Pour cela, un Arduino ou équivalent est utilisé. Il est nécessaire que le modèle choisi supporte l'émulation de clavier/souris HID (Teensy, Arduino Leonardo, Arduino Micro). Cet Arduino est piloté en I2C par un Raspberry Pi. Le montage est donc assez simple, il y a trois câbles à brancher entre l'Arduino et le Raspberry Pi.

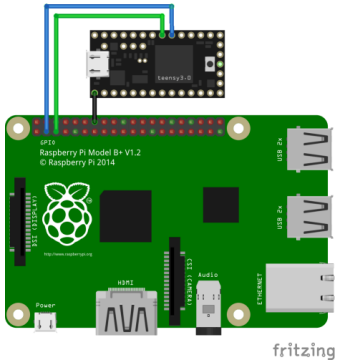


**Fig. 1.** Montage avec une Teensy 3.0

*Note :* Ce montage est la deuxième version du framework. La première version du framework avait un fonctionnement uniquement basé sur l'Arduino et des *shields* permettant la lecture des dictionnaires sur carte SD et l'affichage d'information sur un écran LCD.

### 2.3 Utilisation du framework

L'utilisation du framework se fait via l'appel d'un script Python. Un fichier modèle est préparé pour la cible à attaquer. Ce fichier décrit les actions à réaliser dans une syntaxe simple que l'on pourrait comparer à celle utilisée par l'USB Rubber Ducky. Voici des exemples de mots clés simples utilisés : password, enter, delay 100 (en ms). Des mots clés plus particuliers existent, notamment un qui consiste à préciser qu'il est nécessaire d'appeler un autre fichier modèle toutes les N tentatives. Ceci sera utilisé par la suite afin de prendre en compte le blocage d'Android toutes les cinq tentatives.

Le framework embarque différents fichiers modèles testés sur différents types de cibles (BIOS/UEFI, Android, etc.).

## 3 Attaque d'Android avec le framework

### 3.1 Mot de passe

La phase d'approche consiste à brancher un clavier sur le téléphone et à tester la saisie successive de mots de passe :

- entrer le mot de passe ;
- appuyer sur entrée ;
- attendre la vérification du mot de passe ;
- recommencer.

Lors de la cinquième tentative, le téléphone est bloqué 30 secondes. Il est donc nécessaire de préciser un comportement à suivre afin d'attendre les 30 secondes lorsque ce cas survient. Le fichier modèle utilisé pour l'attaque est donc le suivant.

```
password
delay 300
enter
delay 3500
screenshot
wait wait-generic.txt 5
```

**Listing 1.** Attaque de mot de passe

Un second fichier modèle est indiqué via la commande *wait* afin de préciser le comportement à tenir tous les cinq essais.

```
enter           enter           enter           delay 6000
delay 5000      delay 5000      delay 5000
enter           enter           enter
```

**Listing 2.** Fichier modèle d'attente

La touche entrée est pressée toutes les 5 secondes afin d'éviter la mise en veille du téléphone. Le même fichier sera utilisé pour l'attaque de code PIN et le schéma de verrouillage.

### 3.2 Code pin

Si un dictionnaire est utilisé pour l'attaque du code PIN, le principe est le même que dans le cas précédent à ceci près que le temps d'attente pour la vérification du code PIN peut être moins important que pour le mot de passe. Dans le cas où l'on souhaite réaliser du bruteforce, il est aussi nécessaire de prendre en compte le point précédent. Ensuite il suffit de remplacer le mot clé *password* par *bruteforce numeric X-Y*. X étant la taille minimale de code PIN à attaquer par force brute, et Y la taille maximale. Notons que *bruteforce numeric X* est aussi fonctionnel.

```
bruteforce numeric 4-6
delay 300
enter
delay 3500
screenshot
wait wait-generic.txt 5
```

**Listing 3.** Attaque de code Pin

### 3.3 Schéma de verrouillage

La première étape consiste à analyser le comportement de l'équipement lorsque l'on branche une souris : où celle-ci est positionnée initialement ; et à quelle distance elle se trouve du point en haut à gauche. Ensuite, il est nécessaire de déterminer l'écart entre deux points du schéma. Cette écart sera appelé delta.

Dans le fichier modèle on précise donc le premier mouvement à réaliser pour rejoindre le point en haut à gauche, la valeur de delta, puis nous retrouvons nos éléments déjà vus précédemment :

```
initMouse -240 160
delta 240
pattern
delay 200
screenshot
wait wait-generic.txt 5
```

**Listing 4.** Attaque de schéma de verrouillage

### 3.4 Cas spéciaux

Sur la Samsung Galaxy Notes 10.1 (Android 4.1.2) lorsque cinq mauvaises tentatives de schéma de verrouillage sont réalisées, un code PIN de secours est utilisable. Celui-ci est attaquable sans limitation de temps toutes les 5 tentatives.

## 4 Détection de réussite de l'attaque

### 4.1 Problématiques

Il existe plusieurs cas simples pour détecter une attaque réussie : soit assister à l'attaque ou soit filmer l'attaque puis regarder la vidéo. Une autre méthode consiste à réaliser une photo de l'écran à chaque tentative et comparer les images à l'issue de l'attaque. Pour la comparaison, nous utilisons la fonctionnalité *compare* de *ImageMagick*. La comparaison de deux images retourne un score entre 0 (deux images complètement différentes) et 1 (deux images parfaitement identiques).

Cette méthode entraîne deux problématiques. La première est que la durée de l'attaque est augmentée par le temps de capture de chaque image. À titre d'exemple, la caméra officielle pour Raspberry Pi met entre 5 et 7 secondes pour capturer une image. Une webcam USB (2MP) met entre 600 et 900 ms. La seconde problématique est celle de la luminosité. En effet, deux photos d'un équipement dans un même état avec une luminosité qui diffère ne seront pas identiques. Il est donc nécessaire d'enfermer l'équipement dans une boîte à l'abri de la lumière et de sortir les équipements d'attaque afin d'éviter d'ajouter du bruit du fait de leds qui clignotent.

### 4.2 Fonctionnement

L'ajout de captures d'images au sein du framework est assez simple. Il se fait par l'ajout d'une webcam en USB (supportée par le Raspberry Pi ou équivalent) et par l'utilisation du mot clé *screenshot* dans le fichier modèle. Le fichier de l'image capturée utilise le mot de passe tenté.

Avant de lancer l'attaque, il est nécessaire de réaliser des captures témoins des différents états possibles de la cible en mode verrouillé, par exemple, l'écran de verrouillage et l'écran avec le message signalant qu'il est nécessaire d'attendre trente secondes.

Une fois l'attaque terminée, un script Python permet de comparer chaque capture avec les fichiers témoins. Un fichier CSV est retourné par la commande. L'analyse peut se faire en comparant les valeurs ou bien en réalisant des graphiques. Quelques points d'attention : la valeur -1 correspond au cas où l'image n'est pas disponible (erreur lors de la capture). Il est important de couper le téléphone du réseau afin d'éviter des nuisances liées à la réception de SMS, notifications Facebook, etc.

## 5 Limitation des attaques sur Android

Dans le cas d'attaques par force brute de code PIN, les cibles testées (Samsung Galaxy S4 et S6) ont eu des comportements anormaux allant jusqu'au plantage du téléphone. Il n'a pas encore été possible d'identifier le problème. Cependant, ce type d'attaque peut s'apparenter à du *fuzzing* et il est envisageable d'utiliser le framework afin de réaliser du *fuzzing* sur Android.

## 6 Amélioration du framework

Les sujets d'amélioration en cours d'analyse côté matériel sont le support d'autres Pi (Orange Pi, Banana Pi) et l'utilisation du port OTG d'un Raspberry Pi Zero ou d'un Orange Pi afin de se passer de l'Arduino. Cela réduirait le coût de l'ensemble et pourrait également réduire le temps d'attaque en évitant les communications I2C. Cependant, l'utilisation d'une carte moins puissante pourrait engendrer un temps de capture d'image plus long. Les améliorations logicielles sont à réaliser sur la détection : comparaison des images à la volée depuis une autre machine, détection du succès de l'attaque.