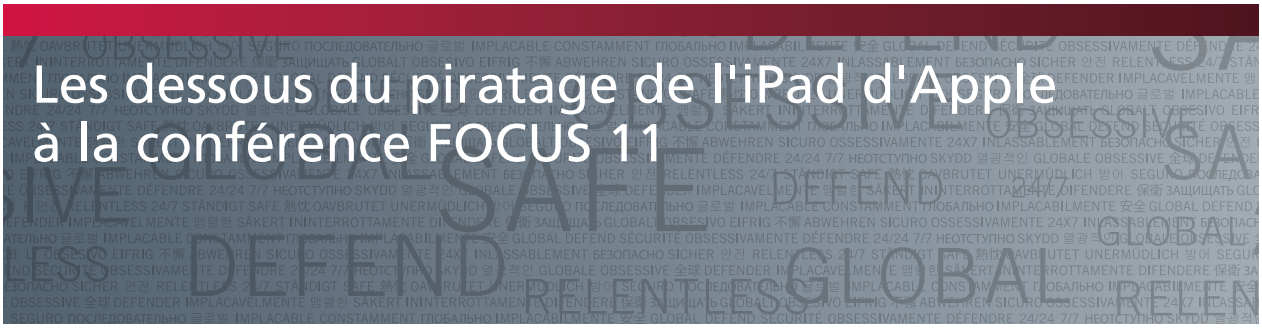
Livre blanc

Les dessous du piratage de l'iPad d'Apple à la conférence FOCUS 11



Gabriel Acevedo et Michael Price
McAfee® Labs™

Sommaire

Lors de la conférence FOCUS 11 organisée par McAfee en octobre dernier, les clients ont eu l'occasion d'assister à plusieurs discussions sur les logiciels malveillants (*malware*) et d'autres menaces ainsi qu'à des présentations abordant de nombreux thèmes de la sécurité. L'exposé *Hacking Exposed* (Halte aux hackers) a remporté un succès remarquable, rassemblant plus de 2 000 personnes. La plupart des participants ont éteint leur ordinateur portable, téléphone, iPad ou autre terminal lorsque le Directeur des technologies de McAfee, Stuart McClure, a annoncé que son équipe allait effectuer un piratage en direct pendant la session. Ce piratage, qui avait fait beaucoup parler de lui sur Twitter et d'autres réseaux sociaux, était attendu avec impatience par tous les visiteurs présents à la conférence. Quelques sceptiques ont prétendu qu'il ne s'agirait pas d'une démonstration en direct mais dès les premiers instants, des participants ont confirmé que le faux point d'accès Wi-Fi de FOCUS était effectivement opérationnel.

Une caméra de télévision filmait l'écran de l'iPad cible de l'attaque afin que tous puissent suivre l'opération en direct. Pendant que Stuart McClure chargeait Gmail via une connexion SSL (Secure Socket Layer), l'URI HTTPS et l'icône du verrou étaient visibles dans Safari. Dans le même temps, un exploit était discrètement chargé et installait un serveur SSH (Secure Shell). Quelques secondes plus tard, la foule applaudissait l'apparition de l'écran de l'iPad sur notre client de commande à distance VNC. L'iPad était compromis. Nous disposions d'un accès de niveau superutilisateur (*root*) via le terminal ainsi que d'un accès graphique via VNC qui nous permettait de suivre en direct les manipulations de Stuart McClure et d'interagir avec son terminal. Un utilisateur normal confronté à une telle situation n'aurait aucun moyen de savoir que son iPad était totalement sous notre contrôle. En effet, la victime ne consultait pas un site web malveillant mais vérifiait simplement ses e-mails via une connexion sécurisée. Comment un tel piratage a-t-il été possible ?

Ce rapport va répondre à cette question mais aussi expliquer nos recherches, les problèmes rencontrés et les outils utilisés.

Il convient de noter que nous présentons les techniques utilisées dans le seul but d'expliquer et de faire prendre conscience aux utilisateurs des faits et de l'impact de ces techniques sur la sécurité. McAfee ne recourt jamais à ces techniques pas plus qu'il ne sanctionne leur utilisation pour enfreindre les lois en vigueur ou l'éthique.

Motivation

En juillet 2011, l'équipe SpiderLabs[1] de TrustWave révélait une vulnérabilité liée à une erreur de validation d'une contrainte de base dans la chaîne de validation des certificats SSL d'iOS. Les chercheurs avaient découvert qu'il était possible de générer un faux certificat à l'aide d'un autre certificat qui, bien que valide, ne devait normalement pas être utilisé afin de générer des certificats pour d'autres sites web. Toutefois, SpiderLabs n'a donné que peu de détails à ce sujet. Etait-il difficile de reproduire la vulnérabilité ? Comment les pirates pouvaient-ils l'exploiter autrement que pour intercepter des données censées rester confidentielles ?

Cela fait quelque temps maintenant que les vulnérabilités exploitées par jailbreakme.com (appelées collectivement JailbreakMe ou JBME) sont connues du public. Pour autant que l'on sache, leur utilisation restait jusqu'ici limitée au site jailbreakme.com puisque bon nombre d'utilisateurs cherchent effectivement à débrider leur terminal Apple. Cela dit, beaucoup ne se rendent sans doute pas compte que la technique JBME exploite des failles afin de prendre le contrôle du terminal, et qu'à l'instar de nombreuses autres vulnérabilités, il est possible d'utiliser celles-ci à des fins malveillantes.

Que se passerait-il si nous venions à combiner ces deux vulnérabilités ? Serait-il possible de les utiliser dans le cadre d'une attaque silencieuse, efficace et sophistiquée ? Notez que l'exploitation de la vulnérabilité SSL n'était pas indispensable pour effectuer le piratage, mais nous souhaitions prouver que même lorsque son propriétaire estime être en sécurité, il est toujours possible de compromettre l'équipement. Notre intention était donc d'exploiter le système pendant que la victime consultait un site bancaire, vérifiait sa messagerie ou effectuait un paiement en ligne, toujours via une connexion SSL.

Ces deux vulnérabilités ont été corrigées dans les versions récentes d'Apple iOS. Toutefois, de nombreux utilisateurs n'ont pas effectué de mise à niveau pour de multiples raisons, parfois par simple ignorance ou parce qu'ils souhaitaient conserver un terminal débridé afin de télécharger des applications qui n'étaient pas disponibles sur le site App Store d'Apple. Parmi ces utilisateurs, combien restent vulnérables ?

Comme vous le voyez, les questions sont nombreuses. Il est temps à présent de tenter d'y répondre.

Recherche

Vulnérabilité de l'implémentation SSL du système iOS

En juillet 2011, Paul Kehrer et Gregor Kopf ont révélé l'existence d'une vulnérabilité liée à une erreur de validation des saisies (CVE-2011-0228) dans le système d'exploitation Apple iOS. Le système d'exploitation mobile ne parvenait pas à valider la chaîne de certificats X.509, ce qui permettait à un pirate de capturer ou de modifier les données protégées par SSL/TLS et de lancer des attaques de l'intercepteur (*man-in-the-middle*). La vulnérabilité concernait les versions 4.3.4 et antérieures d'iOS pour les équipements réseau GSM et les versions 4.2.9 et antérieures pour les équipements réseau CDMA. Les chercheurs ont averti Apple, qui a distribué les mises à jour de sécurité HT4824 et HT4825 pour les équipements CDMA[2,3]. Plus tard, lors de la conférence DEFCON 19, Paul Kehrer a révélé des détails supplémentaires, dont le fait qu'iOS ignorait l'extension des contraintes de base X.509 v3[4].

Les certificats émis pour les entités racine et intermédiaires doivent inclure l'extension Basic Constraints avec un champ CA affecté de la valeur TRUE[5], comme le montre l'exemple ci-dessous :

```
Extensions X509v3 :
    1.3.6.1.4.1.311.20.2:
        ...C.A
    Utilisation de la clé X509v3 :
        Signature numérique, Signature du certificat, Signature de la liste
de révocation de certificats
    Contraintes de base X509v3 : critique
        CA:TRUE
```

En revanche, pour les certificats utilisateur, le paramètre CA doit avoir la valeur FALSE, comme illustré ci-après :

```
Extensions X509v3 :
    Contraintes de base X509v3 :
        CA:FALSE
```

La faille présente dans ces versions d'iOS permet à un auteur d'attaque d'utiliser un certificat légitime sans CA (CA:FALSE) afin de signer d'autres certificats pour un domaine quelconque et de les faire accepter par les équipements sans avertissement, comme n'importe quel certificat X.509 normal signé par une entité valide. Par exemple, la chaîne de certificats suivante fonctionne sur iOS version 4.3.4 mais pas avec les versions corrigées (p. ex. la version 4.3.5) :

```
Certificat racine
    Certificat intermédiaire (CA:TRUE)
        Certificat d'utilisateur final (CA:FALSE)
            Certificat de domaine arbitraire — Fiable
```

L'extension des contraintes de base comporte également un paramètre facultatif de longueur de chemin. Ce paramètre indique le nombre maximal de certificats qui peuvent apparaître sous le certificat concerné dans la chaîne. Dès lors, en se référant à l'exemple ci-dessus, si le certificat intermédiaire possède un paramètre de longueur de chemin dont la valeur est zéro, l'équipement iOS n'acceptera pas un certificat de domaine arbitraire généré par le certificat d'utilisateur final. La chaîne de certificats suivante ne fonctionnera pas, même sur iOS 4.3.4.

```
Certificat racine
    Certificat intermédiaire (CA:TRUE,pathlen:0)
        Certificat d'utilisateur final (CA:FALSE)
            Certificat de domaine arbitraire — Fiable
```

Nous sommes arrivés à cette conclusion lorsque nous avons tenté de reproduire cette vulnérabilité. L'avis de sécurité initial indiquait qu'iOS ne reconnaissait aucune des contraintes de base mais nous avons déterminé que les versions vulnérables ignoraient uniquement la valeur du paramètre CA.

Par conséquent, pour générer un faux certificat X.509 capable d'exploiter la vulnérabilité CVE-2011-0228, le certificat de la chaîne précédant directement le certificat qui génère le faux certificat ne doit pas inclure le paramètre pathlen ou alors il doit spécifier une valeur suffisamment élevée pour celui-ci.

La vulnérabilité « JailbreakMe »

JBME est un site web qui contient une série de débrideurs (ou *jailbreaks*) pour différentes versions d'iOS. Ceux-ci ont été conçus par comex et exploitent des failles de Safari et du noyau. La première version a été distribuée en 2007 et fonctionnait pour le micrologiciel 1.1.1. de l'iPod touch et l'iPhone d'Apple. Une deuxième version a été publiée en août 2010 pour iOS 4.0.1 et la dernière en juillet 2011 pour iOS 4.3.3. Apple a corrigé les vulnérabilités exploitées par JBME dans la version iOS 4.3.4.

La version JBME de juillet 2011, appelée aussi Saffron, exploite une vulnérabilité (CVE-2011-0226) du composant d'analyse FreeType du navigateur Mobile Safari à l'aide d'un fichier PDF spécialement conçu à cette fin. La routine contenue dans le fichier PDF crée et charge une charge active ROP (Return-Oriented Programming, programmation orientée retour) dans la pile et exploite une vulnérabilité du noyau dans l'interface IOKit IOMobileFrameBuffer (CVE-2011-0227). La routine utilise ensuite une autre charge active pour modifier certaines fonctions du noyau afin de désactiver la mise en œuvre de la signature du code et d'obtenir des privilèges de superutilisateur (*root*). Une fois l'exploitation terminée, un pirate peut installer des applications non signées telles que Cydia et toutes les applications fournies via ce système[6].

Des milliers de personnes utilisent JBME pour débrider leurs terminaux, accéder à des applications non disponibles dans la boutique App Store d'Apple ou modifier des paramètres inaccessibles aux utilisateurs normaux. Malheureusement, elles ne se rendent pas compte qu'un tel débridage n'est possible que grâce à l'exploitation de leur terminal.

Le piratage

Matériel

Le piratage n'a nécessité qu'un équipement limité : un ordinateur portable Apple MacBook Air et une clé matérielle (*dongle*) sans fil. Nous avons utilisé ce dongle pour localiser différents points d'accès à partir desquels lancer notre démonstration d'attaque.

Le MacBook Air est un terminal très pratique pour les conférences, les cafés ou les salons d'aéroport. La régénération du fichier PDF malveillant nécessite un ordinateur Mac. Pour le reste, n'importe quel ordinateur portable ou de bureau peut faire l'affaire, mais le système d'exploitation doit être de type Unix, par exemple Mac OS X, Linux ou FreeBSD. Ceux-ci permettent de transférer plus facilement le trafic à l'aide d'un pare-feu du noyau tel qu'ipfw ou iptables. Nous avons également employé plusieurs outils installés par défaut sur ces systèmes d'exploitation.

Logiciels et outils

Nous avons eu recours à plusieurs applications très pratiques pour réaliser le piratage. Certaines d'entre elles étaient de simples scripts mais d'autres étaient plus sophistiquées, notamment Apache HTTP Server, plusieurs outils inclus par défaut dans le système d'exploitation et quelques applications personnalisées.

Pour générer le fichier PDF, nous avons eu besoin de l'outil star_ de comex. Il contient tous les fichiers nécessaires à la génération du fichier PDF malveillant, qui nécessite l'outil xpwn de posixninja[7]. Ce projet comprend également l'exploit du noyau. L'attaque nécessite un micrologiciel iPad déchiffré personnalisé ou un micrologiciel que vous créez vous-même à l'aide de l'outil xpwn (les clés de déchiffrement ne sont pas toujours incluses)[8]. A l'aide d'instructions et des outils de comex, nous avons généré les fichiers suivants :

- Fichier PDF — Pour prendre le contrôle initial du système.
- freeze.tar.xz — Package contenant les fichiers et les applications (telles que VNC) à installer.
- install-3.dylib — Bibliothèque dynamique utilisée pendant l'exploitation.
- saffron-jailbreak.deb — Package contenant une série de fichiers binaires pour amorcer automatiquement le processus d'installation.

Pour transmettre l'exploit et les applications à l'iPad, nous avons utilisé Apache HTTP Server, le composant de partage web, inclus dans Mac OS X. Dans la mesure où le serveur web devait transmettre le contenu via SSL, nous avons dû créer un certificat SSL pour celui-ci. Pour ce faire, nous avons eu recours à l'outil OpenSSL inclus par défaut dans OS X.

L'outil tcpdump (inclus par défaut dans OS X) nous a permis de vérifier que l'exploit avait bien été demandé par l'iPad. Nous avons configuré la charge active freeze.tar.xz afin qu'elle dépose un fichier dans le répertoire /var/mobile/Media/post-jailbreak, qui est exécuté par l'exploit star_ (s'il est présent). Nous avons ajouté au fichier une commande ping vers notre système afin qu'il nous avertisse du démarrage de SSH. Nous avons utilisé tcpdump pour vérifier si l'iPad envoyait une requête ping à notre ordinateur portable.

L'outil ipfw de BSD nous a permis de configurer plusieurs règles de pare-feu sur l'ordinateur portable afin de rediriger le trafic de l'iPad vers notre outil d'attaque de l'intercepteur, iSniff.

iSniff a été écrit par hubert3 et s'inspirait de l'outil sslsniff de Moxie[9]. L'outil d'hubert3 « renifle » le trafic SSL en envoyant à la victime de faux certificats générés à la volée. Nous avons modifié le code source d'iSniff afin de pouvoir manipuler les données que la victime envoyait au serveur et inversement. (Comme iSniff a été conçu pour s'exécuter sur Debian GNU/Linux, nous l'avons modifié pour qu'il fonctionne sur OS X.)

```
iSniff.py › run
            import re
            the_match = re.search('Content\-Length: ([0-9]+)', data)
            if the_match != None:
                chunked = False
                to_replace = the_match.group(0)
                current_size = the_match.group(1)
                data = data.replace(to_replace, "Content-Length: " + str(int(current_size) +
                    dynamic_size))


            if linenotfound:
                # TODO: Some pages have ugly HTML and will use BODY instead of body.
                if data.find("</body>") > 0:
                    linenotfound = False

                    if chunked:
                        static_size = 18 # Comment, body and html closing
                        total_size_hex = hex(static_size + dynamic_size)[2:]
                        chunk_info = "\r\n" + total_size_hex + "\r\n"
                        data = data.replace("</body>", "<!-- nn1234567" + chunk_info + " -->" +
                            MITM_Insertion + "</body>")
                    else:
                        chunk_info = ""
                        data = data.replace("</body>", MITM_Insertion + "</body>")

        if self.logfile:
            self.logfile.write(data)
            self.logfile.flush()

        self.sink.send( data )
    except:
        break

PipeThread.pipes.remove( self )
```

Scénario

Nous avons utilisé l'interface sans fil USB de notre MacBook pour nous connecter au point d'accès Wi-Fi de FOCUS et l'interface sans fil intégrée AirPort pour créer un point d'accès avec un nom SSID similaire au nom du réseau légitime. La similarité du nom de notre point d'accès avait pour but d'inciter la victime à se connecter à notre réseau. L'astuce consiste à prendre un nom SSID légitime et à ajouter un espace devant celui-ci pour en créer un autre. De cette façon, le nom du faux point d'accès figure en première place dans la liste des réseaux.

Un point d'accès créé par des pirates reste généralement accessible (sans mot de passe). Toutefois, dans ce cas-ci, nous avons ajouté un mot de passe afin que le public assistant à notre démonstration ne soit pas accidentellement piraté.

Attaque

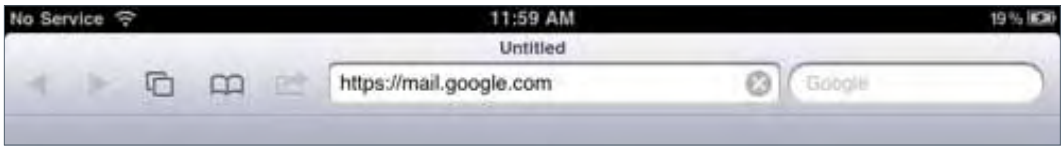
Nous avons connecté l'ordinateur portable à Internet à l'aide du dongle sans fil, démarré le serveur web et partagé la connexion Internet via AirPort afin de laisser croire à la victime que nous hébergions un point d'accès Wi-Fi gratuit.

Nous avons ouvert deux onglets dans l'application Terminal du Mac. Dans le premier, nous avons démarré tcpdump afin qu'il écoute l'interface réseau partagée. Dans le second, nous avons créé quelques règles de pare-feu à l'aide de l'outil ipfw, dont une particulièrement utile :

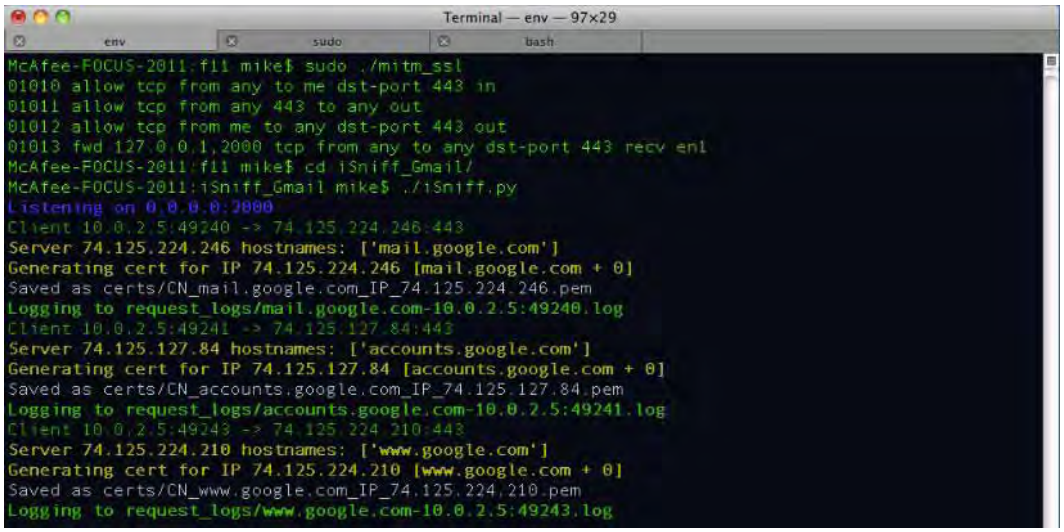
```
sudo ipfw add 1013 fwd 127.0.0.1,2000 tcp from any to any 443 recv en1
```

Dans cette règle, 1013 représente le numéro de la règle, 127.0.0.1,2000 l'adresse qu'écoute le serveur proxy utilisé pour l'attaque de l'intercepteur, 443 indique que nous souhaitons intercepter les connexions via le port SSL standard, et en1 constitue l'interface réseau par l'intermédiaire de laquelle nous partageons la connexion Internet. Ensuite, nous avons démarré le serveur proxy en question (iSniff). Il était configuré pour écouter le port 2000.

Nous avons regardé notre victime accéder au site https://mail.google.com sur l'iPad.



Le proxy a intercepté la connexion de la victime, généré un certificat pour le nom du domaine auquel cette dernière tentait d'accéder (mail.google.com), lui a ensuite envoyé ce certificat pour établir une connexion « sécurisée » avec le terminal et, pour terminer, usurpé son identité pour établir une connexion avec la destination finale.



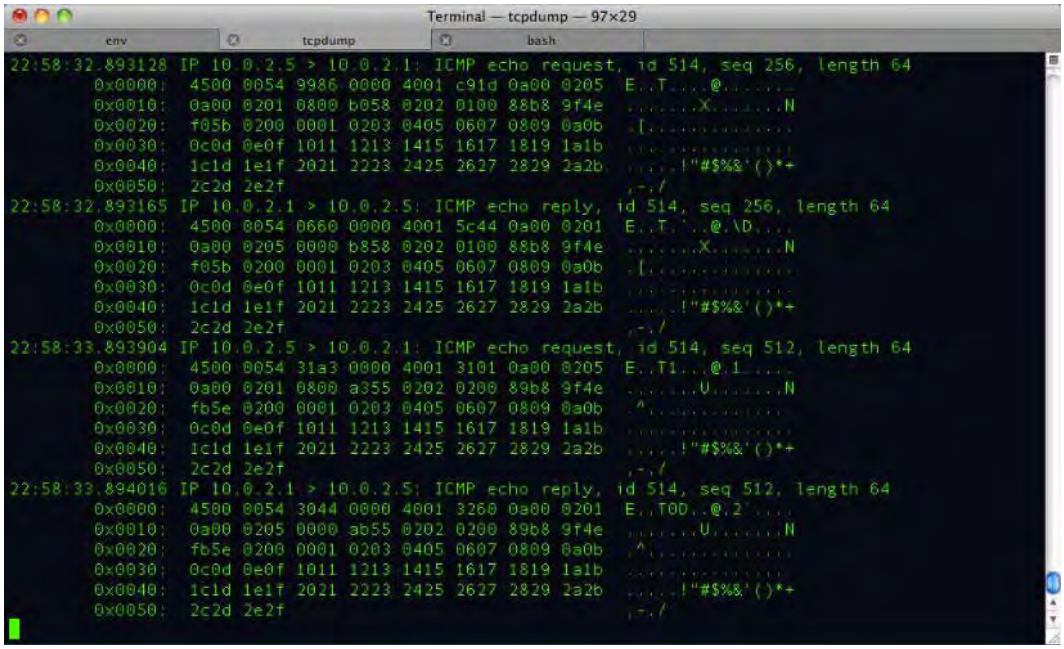
Comme la version d'iOS vulnérable ne pouvait pas vérifier la chaîne de certificats SSL, elle a fait confiance au certificat que nous avions généré sans déclencher d'alarme ni renvoyer d'erreur. Dans ce scénario, nous avons pu voir tout le trafic entrant et sortant mais également le modifier.

L'outil d'attaque de l'intercepteur a modifié la requête HTTP de la victime et demandé au serveur une page non compressée, ce qui nous a permis de traiter et de manipuler plus facilement la page web retournée afin d'y insérer un élément HTML iframe juste avant la balise de fermeture </body> du code HTML.

L'iframe contenait une page web avec un lien destiné à charger un fichier PDF malveillant. Nous aurions pu réduire la taille de l'iframe afin de le rendre presque invisible à l'utilisateur. Nous aurions pu également l'insérer tout en bas de la page. Toutefois, pour les besoins de la démonstration, nous avons conservé un iframe de grande taille afin que le public puisse le voir à l'écran.

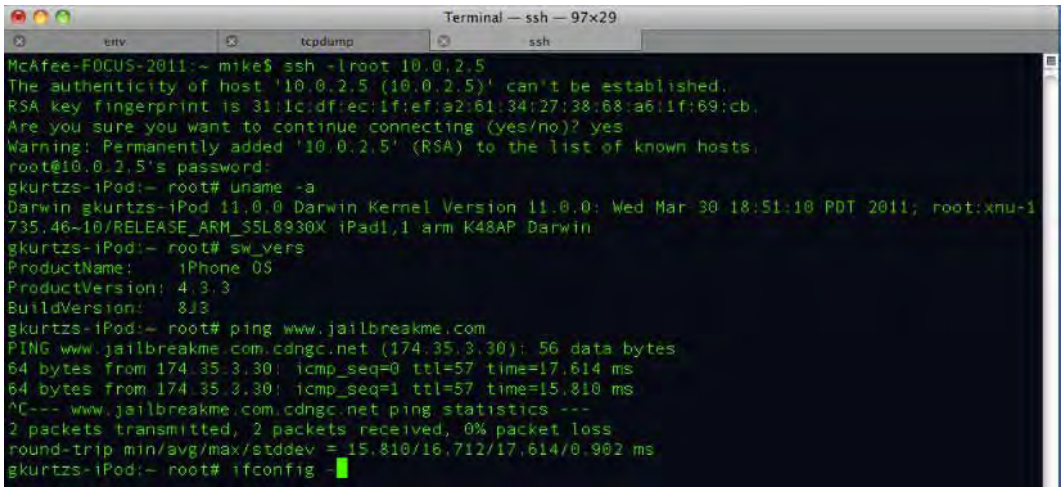
Le fichier PDF contenait une police FreeType Type 1 spécialement conçue pour exploiter le navigateur web Mobile Safari et exécuter un code spécial afin de télécharger une charge active sur le terminal. L'iPad a téléchargé tous ces fichiers (package .deb saffron, freeze.tar.xz, install.dylib) depuis notre serveur HTTP local. Comme nous avons généré un certificat SSL pour le domaine pointant vers notre ordinateur local, la connexion restait sécurisée via SSL pendant le téléchargement du fichier PDF, ce qui affichait une icône de verrou dans Safari et inspirait confiance à l'utilisateur.

Lors de la compromission de l'iPad de la victime, nous avons téléchargé une charge active qui installait le serveur SSH, VNC ainsi que quelques dépendances de notre serveur HTTP. C'est à ce stade que JBME aurait dû installer Cydia mais, comme nous ne voulions révéler aucun signe d'une attaque, nous avons modifié le code star_ de JBME pour ne pas créer d'icône sur le Bureau ni installer Cydia. Nous avons utilisé le script postinstall pour effacer certaines de nos empreintes, notamment l'icône VNC et des paramètres du menu Réglages d'iOS. Ce script a ensuite émis une commande ping pour nous avertir que le terminal était prêt et acceptait les connexions sur le port 22. Nous avons détecté la commande ping grâce à l'outil tcpdump exécuté dans Terminal.

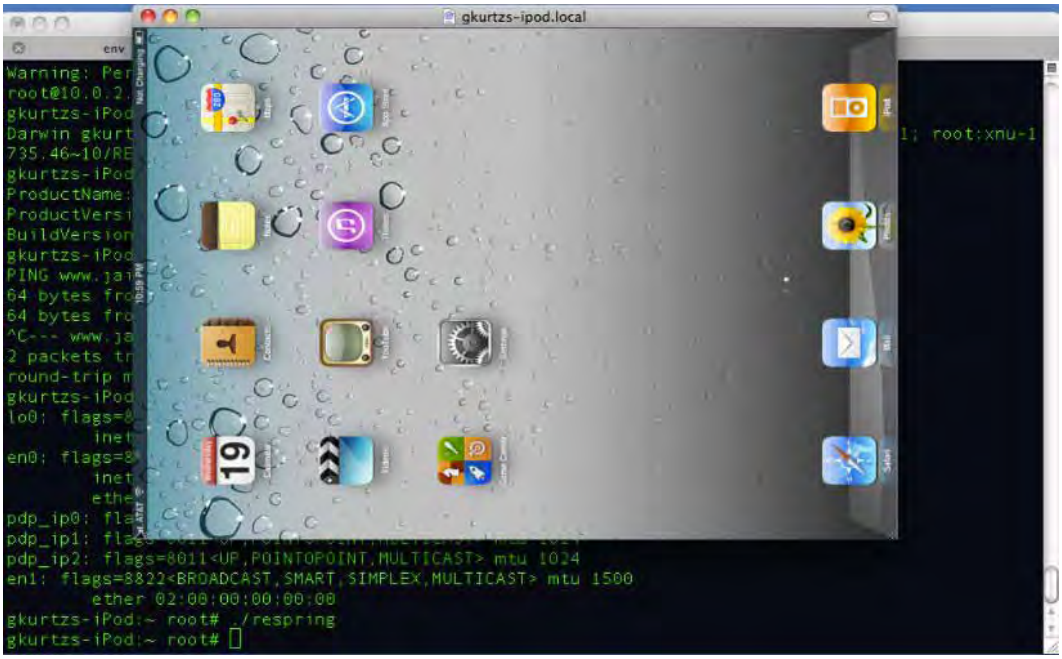


Résultats

Ensuite, nous avons utilisé SSH pour accéder au terminal en tant que superutilisateur (*root*) et le manipuler à volonté : lire des message privés, télécharger des photos personnelles, capturer la liste de contacts, installer d'autres applications (tel un enregistreur de frappes) et bien plus encore.



Nous avons rechargé l'interface de l'iPad pour que le serveur VNC réponde correctement à nos connexions. Ensuite, nous avons démarré notre client VNC préféré et l'avons connecté au terminal. Nous avons pu voir tout ce que faisait la victime sur l'iPad et même interagir avec le terminal.



Alors que nous opérions à l'insu total de la victime, celle-ci continuait à consulter Gmail et d'autres sites web, sans soupçonner un seul instant que nous avions compromis son système et toutes ses données personnelles.

Dès que le « springboard » de l'iPad est apparu à l'écran, le public de FOCUS s'est mis à applaudir. Plusieurs participants nous ont posé des questions, parfois simplement pour s'assurer que personne n'avait été piraté pendant la conférence. La réponse était évidemment non. Nous nous sommes contentés de montrer ce dont des personnes malintentionnées étaient capables.

Conclusions

Apple iOS est plus sûr que bien d'autres systèmes d'exploitation mais il n'est pas impénétrable. Quelques vulnérabilités sont parfois détectées au sein de ce système d'exploitation et de ses applications. En dépit de dénégations d'Apple à ce sujet, il est clair qu'il ne faut pas se fier uniquement au fournisseur pour la sécurisation de nos terminaux mobiles.

Dès le moment où nous nous croyons en sécurité, nous devenons vulnérables. Pour ce piratage, le fait que l'utilisateur ciblé utilise SSL ou non n'avait aucune importance. En revanche, c'est de son ignorance ou de son insouciance qu'il nous fallait pouvoir profiter. Nous avons tendance à croire que la sécurité de nos systèmes, et celle de nos données, de notre argent et de notre identité en ligne, ne nécessite qu'un mot de passe fort, un pare-feu coûteux (ou une autre appliance) ou encore un bon programme antimalware. En réalité, elle dépend de la combinaison de tous ces éléments. Toutefois, pour être correctement protégé lors de nos interactions en ligne, le plus important reste l'information et la connaissance. Nous devons être conscients de la réalité, à savoir que des pirates rôdent en ligne, que des vulnérabilités existent et qu'il est impératif de les corriger ou de se prémunir contre leur exploitation.

Parfois, une seule vulnérabilité semble inoffensive. Prises isolément, il paraît impossible qu'elles puissent compromettre un terminal. Toutefois, lorsqu'elles sont combinées, elles peuvent poser un risque majeur. N'oubliez jamais qu'il est toujours possible de détourner des outils pratiques et inoffensifs pour les utiliser à des fins malveillantes. De tels outils dans de mauvaises mains peuvent se transformer en armes redoutables capables de compromettre n'importe quel système.

La plupart des logiciels présentent des vulnérabilités. Nous devons nous informer continuellement des avis de sécurité publiés par les fournisseurs et installer les patchs ou les contre-mesures nécessaires pour éviter toute compromission. Nous pouvons également tirer parti des outils de sécurité qui nous avertissent de la présence de logiciels vulnérables au sein de nos réseaux.

Remerciements

Nous tenons à remercier Stuart McClure pour ses encouragements constants et Raul Collantes pour ses idées et l'aide précieuse apportée lors de la révision.

Les auteurs

Gabriel Acevedo est chercheur en sécurité auprès du bureau de McAfee Labs à Santiago (Chili). Il mène des recherches sur les vulnérabilités nouvelles et existantes des plates-formes Microsoft Windows et Unix, des appliances de sécurité et d'autres systèmes. Gabriel Acevedo fait partie de l'équipe McAfee Vulnerability Management Content (anciennement McAfee Foundstone®), une équipe internationale chargée d'implémenter des contrôles logiciels afin de détecter la présence de vulnérabilités sur des systèmes informatiques distants. Il dirige également le groupe de travail Mobile Security Working Group, qui fait partie de McAfee Labs, et mène des recherches sur la sécurité collaborative pour les terminaux mobiles et embarqués.

Michael Price est l'ancien Directeur des opérations du bureau de McAfee Labs à Santiago. Lorsqu'il travaillait pour McAfee, il a collaboré avec des entités externes au Chili et en Amérique latine dans le but d'encourager l'excellence et l'innovation techniques. Michael Price est aujourd'hui responsable de l'architecture iOS chez Appthority, où il est chargé des recherches sur la sécurité des applications et d'iOS.

[1] Paul Kehrer. *Trustwave's SpiderLabs Security Advisory TWSL2011-007* (Avis de sécurité de Trustwave SpiderLabs — TWSL2011-007), juillet 2011. https://www.trustwave.com/spiderlabs/advisories/TWSL2011-007.txt

[2] Apple. *A propos du contenu de sécurité de la mise à jour logicielle iOS 4.3.5 pour l'iPhone*, juillet 2011. http://support.apple.com/kb/HT4824?viewlocale=fr_FR

[3] Apple. *About the security content of iOS 4.2.10 Software Update for iPhone* (A propos du contenu de sécurité de la mise à jour logicielle iOS 4.2.10 pour l'iPhone), juillet 2011. http://support.apple.com/kb/HT4825

[4] Nicholas Percoco et Paul Kehrer. *Getting SSLizzard* (A se procurer absolument : l'outil SSLizzard), août 2011. http://defcon.org/html/defcon-19/dc-19-speakers.html#Percoco

[5] Pour en savoir plus sur les autorités de certification et les certificats, consultez la page http://mcaf.ee/2mjdv

[6] Jean-Baptiste Bédrune. *Analysis of the jailbreakme v3 font exploit* (Analyse de l'exploit lié à une police dans jailbreakme version 3), juillet 2011. http://esec-lab.sogeti.com/post/Analysis-of-the-jailbreakme-v3-font-exploit

[7] https://github.com/comex/star_

[8] https://github.com/posixninja/xpwn

[9] https://github.com/hubert3/iSniff

McAfee S.A.S.
Tour Franklin, La Défense 8
92042 Paris La Défense Cedex
France
+33 1 47 62 56 00 (standard)
www.mcafee.com/fr

41724wp_ipad-hack_0212_fnl_ETMG