White paper

FOCUS 11 - Come ti violo l'Apple iPad

Di Gabriel Acevedo e Michael Price
McAfee® Labs™

Indice dei contenuti

Nell'ottobre dello scorso anno si è tenuta la conferenza FOCUS 2011 di McAfee. McAfee Labs ha offerto ai clienti varie discussioni su malware e altre minacce oltre che su numerosi altri argomenti legati alla sicurezza. Uno degli eventi più apprezzati è stato l'intervento Hacking Exposed, che ha attratto oltre 2.000 persone. La maggior parte dei partecipanti ha spento i propri portatili, telefoni, Apple iPad e altri dispositivi quando Stuart McClure, Chief Technology Officer di McAfee, ha annunciato che il suo gruppo avrebbe eseguito degli atti di pirateria informatica dal vivo durante la sessione. Abbiamo osservato un gran fermento su Twitter e altri social network alla ricerca della violazione di cui tutti stavano parlando alla conferenza. Alcuni scettici hanno affermato che non sarebbe stata una dimostrazione dal vivo, ma poco prima dell'inizio la gente ha confermato che il falso hotspot Wi-Fi di FOCUS era attivo.

Abbiamo puntato una telecamera sullo schermo dell'iPad dimostrativo che stava per essere violato in modo che i partecipanti potessero seguire. Mentre McClure caricava Gmail tramite una connessione SSL (Secure Sockets Layer), gli astanti potevano vedere lo schema dell'URL HTTPS e l'icona del lucchetto su Safari. Dietro le quinte, si stava caricando un exploit che installava un server secure shell (SSH). Pochi secondi dopo, il pubblico ha applaudito in segno di approvazione nel momento in cui lo schermo dell'iPad è comparso sul nostro client di controllo remoto VNC. L'iPad era stato compromesso. Abbiamo ottenuto accesso root tramite il terminale e accesso grafico attraverso il client VNC che ci ha permesso di vedere cosa stava facendo McClure e di interagire con il suo dispositivo. In questa situazione, un tipico utente di iPad non si sarebbe accorto che avevamo ottenuto un tale controllo del sistema. La vittima non stava visitando un sito web pericoloso, ma stava semplicemente controllando la posta elettronica tramite una connessione protetta. Come è stato possibile effettuare tale violazione?

In questo report, risponderemo a tale domanda e spiegheremo la nostra ricerca, i problemi che ci siamo trovati ad affrontare e gli strumenti che abbiamo creato.

(Spieghiamo le tecniche che abbiamo utilizzato solo a scopo educativo e divulgativo sui dati e i principi di sicurezza implicati in tali tecniche. McAfee non utilizza e non approva l'uso di tali tecniche per violare leggi in vigore o superare i limiti del comportamento etico).

Motivazione

Nel luglio 2011, gli SpiderLabs di TrustWave hanno scoperto una vulnerabilità in un errore di validazione delle limitazioni di base nella catena di validazione dei certificati SSL di iOS[1]. I loro ricercatori hanno scoperto che era possibile generare un certificato fasullo utilizzandone un altro che, sebbene valido, non era pensato per generare certificati per altri siti web. Gli SpiderLabs, comunque, non hanno fornito molti dettagli. Quanto complicato è stato riprodurre questa vulnerabilità? Come può essere usata questa vulnerabilità dagli aggressori, oltre che per intercettare dati che dovrebbero restare privati?

È passato del tempo da quando furono rivelate le vulnerabilità sfruttate da jailbreakme.com (note collettivamente come JailbreakMe, o JBME). Finora, il solo uso pubblico che abbiamo visto è stato su jailbreakme.com. A molta gente piacerebbe fare il jailbreaking di un dispositivo Apple. Comunque, ciò di cui alcuni non si rendono conto è che la tecnica JBME sfrutta le vulnerabilità per prendere il controllo del dispositivo e, come nel caso di altre vulnerabilità, tali punti deboli potrebbero essere usati per scopi illeciti.

Cosa succederebbe se combinassimo insieme queste due vulnerabilità? È possibile usarle in un attacco sofisticato, silenzioso ed efficace? Si noti che la vulnerabilità SSL non era necessaria per compiere l'intrusione, ma volevamo provare che anche quando la vittima pensa di essere al sicuro, è ancora possibile compromettere il dispositivo. La nostra idea era quindi di sfruttare il sistema mentre la vittima stava visitando il sito web della banca, controllando la posta oppure facendo un pagamento online, tutto tramite una connessione SSL.

Entrambe queste vulnerabilità sono state corrette nelle recenti versioni del sistema operativo iOS di Apple. Comunque, molti utenti non hanno ancora effettuato l'upgrade per vari motivi, dalla semplice ignoranza al fatto che volevano fare il jailbreak dei propri dispositivi e installare applicazioni non disponibili nell'App Store della Apple. Quanti di questi utenti sono ancora vulnerabili?

Per adesso abbiamo posto un sacco di domande. Nelle sezioni seguenti forniremo le risposte.

Ricerca

Vulnerabilità dell'implementazione SSL del sistema operativo iOS.

Nel luglio 2011 Paul Kehrer e Gregor Kopf hanno rivelato una vulnerabilità da errore di validazione dell'input (CVE-2011-0228) in Apple iOS. Il sistema operativo mobile non era in grado di convalidare la catena di certificati X.509, consentendo a un aggressore di intercettare o modificare i dati protetti dal SSL/TLS ed eseguire attacchi di tipo "man in the middle" (MITM). La vulnerabilità interessava le versioni 4.3.4 e precedenti di iOS per dispositivi di rete GSM e le versioni 4.2.9 e precedenti per i dispositivi di rete CDMA. I ricercatori hanno avvisato la Apple, che ha quindi rilasciato gli aggiornamenti di sicurezza HT4824 e HT4825 per i dispositivi CDMA[2,3]. In seguito, durante DEFCON 19, Kehrer ha rivelato ulteriori dettagli, compreso il fatto che iOS ignorava l'estensione limitazioni di base di X.509 v3[4].

I certificati emessi per le entità di root e intermedie devono includere l'estensione limitazioni di base con il campo CA impostato su TRUE[5]. Per esempio:

```
estensioni X509v3:
    1.3.6.1.4.1.311.20.2:
        ...C.A
    Utilizzo Chiave X509v3:
        firma digitale, firma certificata, firma CRL
    Vincoli di base X509v3: critico
        CA:TRUE
```

Comunque, per i certificati utente il parametro CA dovrebbe essere impostato su FALSE. Di seguito sono riportati alcuni esempi:

```
estensioni X509v3:
    Vincoli di base X509v3:
        CA:FALSE
```

Il difetto di queste versioni di iOS è che consente a un aggressore di usare un certificato non CA legittimo (CA:FALSE) per firmare i successivi certificati per un dominio qualsiasi e di farlo accettare ai dispositivi senza avviso, proprio come un normale certificato X.509 firmato da un'entità valida. Per esempio, la seguente catena di certificati funzionerà sulla versione 4.3.4 di iOS, ma non nelle versioni con la patch (come la versione 4.3.5):

```
Certificato di root
    Certificato intermedio (CA:TRUE)
        Certificato utente finale (CA:FALSE)
            Certificato di dominio arbitrario—Attendibile
```

L'estensione limitazioni di base ha anche un parametro opzionale: pathlen. Questo parametro indica il massimo numero di certificati che possono comparire sotto di esso nella catena. Quindi, considerando l'esempio sopra, se il certificato intermedio (Intermediate Certificate) ha il parametro pathlen impostato su zero, il dispositivo iOS non riterrà attendibili alcuni certificati di dominio arbitrari (Some Arbitrary Domain Certificate) generati dal certificato utente finale (End User Certificate). La seguente catena di certificati non funzionerà neanche su iOS 4.3.4.

```
Certificato di root
    Certificato intermedio (CA:TRUE,pathlen:0)
        Certificato utente finale (CA:FALSE)
            Certificato di dominio arbitrario—Non attendibile
```

Siamo giunti a questa conclusione mentre tentavamo di riprodurre questa vulnerabilità. L'avviso originale affermava che iOS non riconosce le limitazioni di base, ma noi abbiamo concluso che le versioni vulnerabili ignorerebbero solo il valore CA.

Pertanto, per generare un falso certificato X.509 e sfruttare la vulnerabilità CVE-2011-0228, il certificato che nella catena precede immediatamente quello che genera il certificato falso non deve includere il parametro pathlen, oppure deve contenere un valore sufficientemente grande.

La vulnerabilità "JailbreakMe"

JBME è un sito web che contiene una raccolta di jailbreak per diverse versioni di iOS. Progettati da comex, sfruttavano i difetti di Safari e del kernel. La prima versione fu rilasciata nel 2007 e funzionava per il firmware 1.1.1 di iPhone e Apple iPod Touch. Una seconda versione è stata rilasciata nell'agosto 2010 per iOS 4.0.1, mentre la più recente è del luglio 2011. E funziona per iOS 4.3.3. La Apple ha corretto le vulnerabilità usate da JBME in iOS versione 4.3.4.

La versione del luglio 2011 di JBME - chiamata anche Saffron - sfrutta una vulnerabilità (CVE-2011-0226) nel componente di analisi FreeType di Mobile Safari, tramite un file PDF appositamente realizzato. La routine nel PDF costruisce e carica nello stack un payload di programmazione orientato sul valore return, che sfrutta una vulnerabilità del kernel nell'interfaccia IOMobileFrameBuffer IOKit (CVE-2011-0227). La routine quindi usa un altro payload per modificare alcune funzioni del kernel, disabilitare la firma del codice e ottenere privilegi di root. Una volta completato l'exploit, un aggressore potrebbe installare applicazioni non firmate come Cydia e tutte le app fornite tramite quel sistema[6].

Migliaia di persone usano JBME per effettuare il jailbreak dei propri dispositivi, al fine di accedere ad applicazioni non disponibili nell'App Store di Apple o per modificare le impostazioni cui gli utenti normali non possono accedere. Ma ciò di cui la gente non si rende conto è che i loro dispositivi devono essere violati per far sì che il jailbreak abbia successo.

L'intrusione

Hardware

Ci bastava poco: abbiamo usato un portatile Apple MacBook Air e un dongle wireless. Il dongle ci è servito per sondare il terreno e trovare diversi punti dai quali lanciare il nostro attacco dimostrativo.

Il MacBook Air è molto comodo per conferenze, caffè o sale d'attesa degli aeroporti. Un Mac è inoltre necessario per ricostruire il PDF nocivo. Per il resto, non importa tanto quale computer desktop o portatile si usa, quanto il sistema operativo, che dev'essere di tipo Unix come Mac OS X, Linux o FreeBSD. Tali sistemi facilitano l'inoltro del traffico per mezzo di un firewall del kernel come ipfw o iptables. Sono necessari anche strumenti che sono installati di default su quei sistemi operativi.

Software e strumenti

Per eseguire la violazione abbiamo usato alcune comode applicazioni. Alcune erano dei semplici script, altre erano software più sofisticati come Apache HTTP Server, strumenti inclusi di default nel sistema operativo e alcune applicazioni personalizzate.

Per generare il PDF ci serviva star_ di comex, che contiene tutti i file per generare il PDF nocivo e richiede lo strumento xpwn di posixninja[7]. In quel progetto è reperibile anche l'exploit del kernel. È necessario un firmware di iPad decodificato oppure si può generare il proprio tramite xpwn (le chiavi di decodifica non sono sempre incluse)[8]. Usando gli strumenti e le istruzioni di comex abbiamo generato i seguenti file:

- PDF: prende il controllo iniziale del sistema
- freeze.tar.xz: pacchetto contenente i file e le app (come VNC) da installare
- install-3.dylib: libreria dinamica utilizzata durante l'exploit
- saffron-jailbreak.deb: pacchetto contenente una serie di codici binari per avviare il processo di installazione

star_ / pdf / mkpdf.py

Fork and edit this file

100644 | 21 lines (16 sloc) | 0.545 kb raw | blame | history

```
import zlib, sys, re, struct
u = open(sys.argv[1], 'rb').read()
z = zlib.compress(u, 9)

m = re.search('currentfile eexec[\r\n]*', u)
encstart = m.end()
assert u[encstart:encstart+2] == '\x80\x02'
encend = encstart + 6 + struct.unpack('<I', u[encstart+2:encstart+6])[0]
zeroes = u.find('0000000', encend)

fmt = {}
fmt['length'] = len(z)
fmt['length1'] = encstart
fmt['length2'] = zeroes - encstart
fmt['length3'] = len(u) - zeroes
fmt['stream'] = z

pdf = open('out.pdf.template', 'rb').read().format(**fmt)
open(sys.argv[2], 'wb').write(pdf)
```

Per fornire l'exploit e le applicazioni all'iPad abbiamo usato Apache HTTP Server, noto come componente Condivisione Web, incluso in Mac OS X. Il server web doveva fornire il contenuto tramite SSL, quindi era necessario creare allo scopo un certificato SSL. L'abbiamo fatto con lo strumento OpenSSL incluso di default in OS X.

Per verificare che l'exploit fosse richiesto dall'iPad ci siamo avvalsi di tcpdump (incluso di default in OS X). Abbiamo configurato il payload freeze.tar.xz per salvare un file in /var/mobile/Media/post-jailbreak, eseguito dall'exploit star_ (se presente). In quel file abbiamo incluso un comando ping verso il nostro sistema, per sapere quando l'SSH sarebbe stato in funzione. Con tcpdump abbiamo verificato se l'iPad stava inviando il ping al nostro portatile.

Ci serviva lo strumento ipfw di BSD per impostare alcune regole di firewall sul nostro portatile, in modo da reindirizzare il traffico dell'iPad a iSniff, il nostro strumento di MITM.

iSniff è stato scritto da hubert3 ed è stato ispirato dallo strumento sslsniff di Moxie[9]. Lo strumento di Hubert3 "fiuta" il traffico SSL inviando certificati falsi, generati istantaneamente, alla vittima. Abbiamo modificato il codice sorgente di iSniff per poter modificare i dati che la vittima stava inviando al server e viceversa (iSniff è stato progettato per l'esecuzione su Debian GNU/Linux, pertanto lo abbiamo modificato per funzionare su 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 )
```

Scenario

Abbiamo usato l'interfaccia wireless USB sul nostro MacBook per connetterci all'hotspot Wi-Fi di FOCUS e l'interfaccia wireless Airport per creare un hotspot con un nome SSID simile a quello della rete legittima. Essendo vicino a quello reale, il nome del nostro hotspot poteva indurre la vittima a connettersi alla nostra rete. Un semplice trucco è quello di prendere un normale nome SSID e modificarlo solo con uno spazio davanti. In questo modo l'imitazione del nome di hotspot va in cima all'elenco delle reti.

Gli aggressori lasceranno normalmente l'hotspot aperto (non richiederà alcuna password). In questo caso però abbiamo aggiunto una password così che i partecipanti alla nostra sessione non venissero attaccati involontariamente.

Attacco

Abbiamo connesso il portatile a Internet tramite il dongle wireless, avviato il server web e condiviso la connessione Internet tramite Airport per far credere alle vittime che fossimo un hotspot Wi-Fi gratuito.

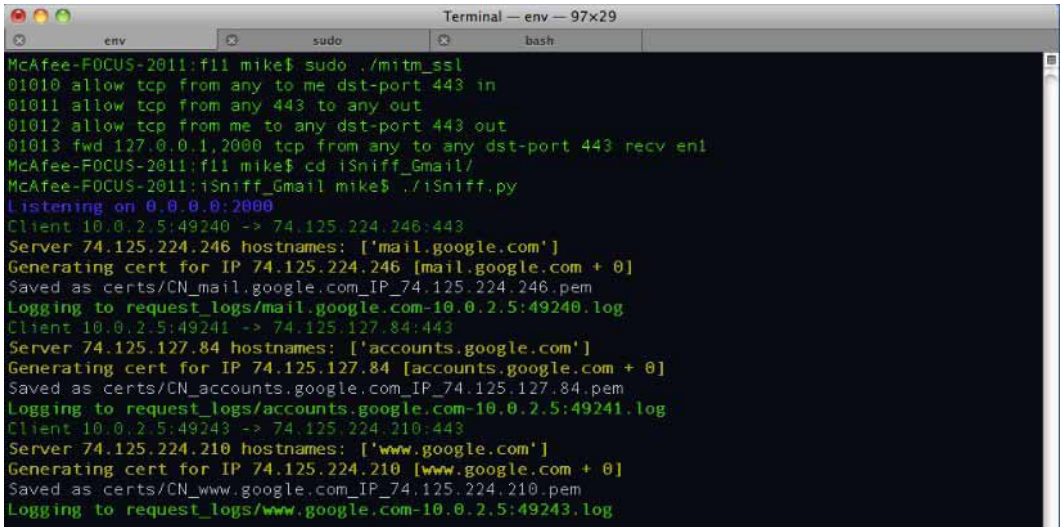
Abbiamo aperto qualche scheda nell'applicazione terminale del Mac. Nella prima abbiamo avviato tcpdump mettendolo in ascolto sull'interfaccia di rete condivisa. Nella seconda scheda abbiamo creato un paio di regole firewall tramite ipfw. Una regola si è rivelata particolarmente utile:

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

In questa regola, 1013 è il numero della regola, 127.0.0.1,2000 indica il punto di ascolto del proxy MITM, 443 indica che desideriamo intercettare le connessioni tramite la porta SSL standard, mentre en1 è l'interfaccia di rete tramite cui stiamo condividendo la connessione Internet. Dopodiché abbiamo avviato il server proxy MITM (iSniff), configurato per ascoltare la porta 2000.

Abbiamo osservato la nostra vittima visitare https://mail.google.com sull'iPad.

Il proxy intercetta la connessione della vittima, genera un certificato per il nome di dominio che la vittima sta cercando di raggiungere (mail.google.com), glielo invia per stabilire una connessione "protetta" con il dispositivo, infine impersona la vittima stessa per stabilire una connessione con la destinazione finale.



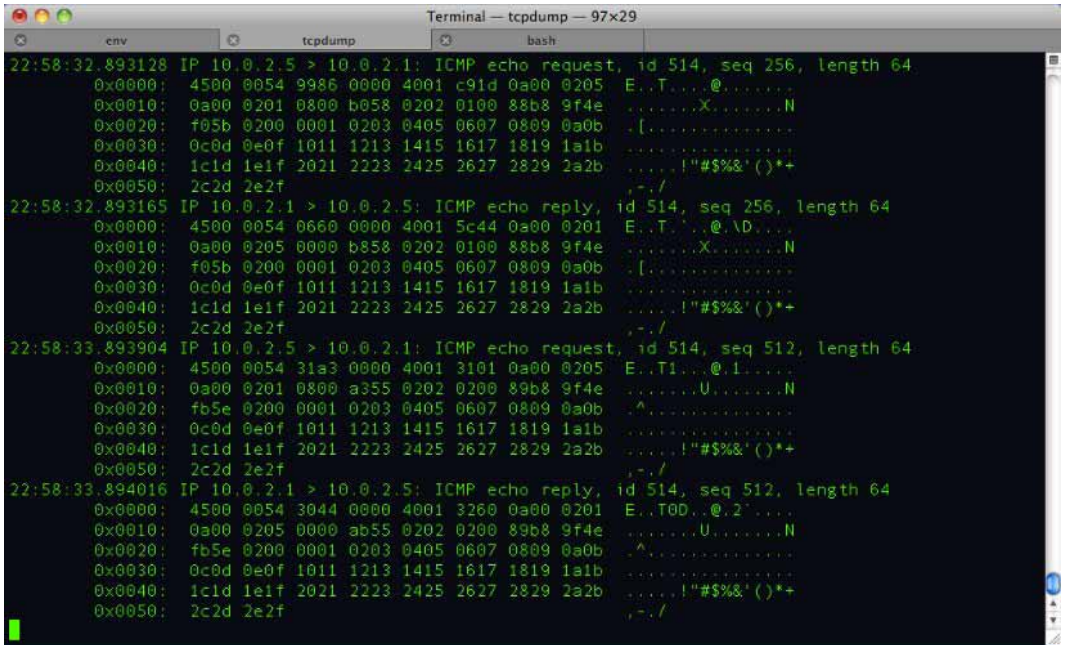
Dato che la versione di iOS vulnerabile non è in grado di verificare la catena di certificati SSL, ritiene attendibile il certificato generato e non fa scattare nessun allarme, né restituisce errori. In tale scenario osservavamo tutto il traffico in ingresso e in uscita, con anche la possibilità di modificarlo.

Lo strumento MITM modificava la richiesta HTTP della vittima chiedendo al server una pagina non compressa; questo ci rendeva facile elaborare e modificare la pagina web restituita inserendo un elemento HTML iframe appena prima del tag di chiusura </body> dell'HTML.

L'iframe conteneva una pagina web con un link per caricare il PDF nocivo e potevamo ridurne le dimensioni al punto da renderlo quasi impercettibile all'utente. Potevamo anche inserirlo proprio in fondo alla pagina. Nel nostro accesso illecito abbiamo però mantenuto una dimensione grande in modo che fosse visibile sullo schermo.

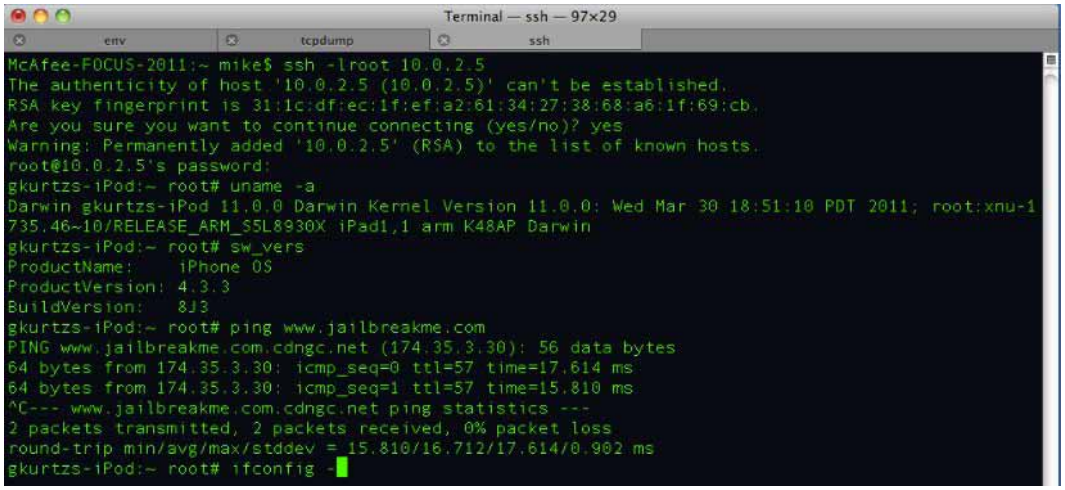
Il file PDF conteneva un font FreeType Type 1 realizzato appositamente per sfruttare il browser web Mobile Safari ed eseguire il codice speciale per scaricarvi il payload. L'iPad ha scaricato tutti questi file (il pacchetto .deb saffron, freeze.tar.xz, install.dylib) dal nostro server HTTP locale. Dato che abbiamo generato un certificato SSL per il dominio che puntava al nostro computer locale, la connessione rimaneva sotto SSL durante lo scaricamento del PDF, facendo visualizzare a Safari l'icona del lucchetto per tutto il tempo e facendo sentire sicura la vittima.

Nella violazione dell'iPad della vittima abbiamo scaricato un payload che installava il server SSH, VNC e qualche dipendenza dal nostro server HTTP. A questo punto JBME di solito installerebbe Cydia, ma dato che non volevamo rivelare alcun segno di attacco, abbiamo modificato il codice star_ di JBME per non fargli creare un'icona sul desktop, né installare Cydia. Abbiamo usato uno script di postinstallazione per rimuovere alcune delle nostre fingerprint, come l'icona VNC e le impostazioni dal menu Impostazioni di Sistema di iOS. Lo script di postinstallazione ha quindi inviato un comando di ping per farci sapere che il dispositivo era pronto ad accettare connessioni sulla porta 22. Abbiamo rilevato il ping con tcpdump in esecuzione sul terminale.

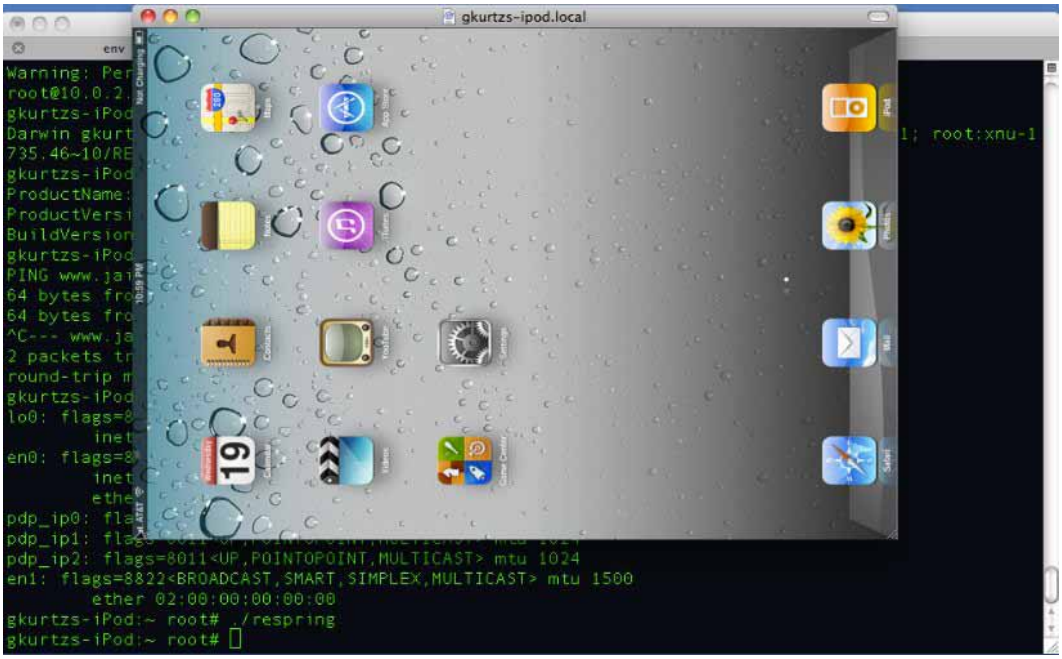


Risultati

Successivamente abbiamo usato SSH per raggiungere il dispositivo come root e poter fare quello che ci pareva: leggere messaggi privati, scaricare immagini private, prendere l'elenco dei contatti, installare ulteriori applicazioni (magari un keylogger) e altro leggere messaggi privati, scaricare immagini private, prendere l'elenco dei contatti, installare ulteriori applicazioni (magari un keylogger) e altro.



Abbiamo ricaricato l'interfaccia dell'iPad per far rispondere adeguatamente il server VNC alle nostre connessioni. Quindi abbiamo avviato il nostro client VNC preferito per connetterci al dispositivo. Potevamo vedere tutto quello che la vittima faceva sull'iPad, addirittura interagendo con essa.



Mentre operavamo sotto mentite spoglie, la vittima continuava a navigare in Gmail e a visitare siti web, ignara del fatto che eravamo penetrati nel sistema, compromettendo tutti i suoi dati privati.

Non appena la springboard dell'iPad è comparsa sullo schermo, il pubblico del FOCUS ha fatto partire un applauso. Alcuni si sono avvicinati per farci delle domande e, in qualche caso, per essere sicuri che non avessimo attaccato nessuno durante la conferenza. Ovviamente, non l'abbiamo fatto. Abbiamo solo dimostrato di cosa sono capaci le persone senza scrupoli.

Conclusioni

Apple iOS è più sicuro di molti altri sistemi operativi, ma non è impenetrabile. Ogni tanto vengono scoperte vulnerabilità in questo sistema operativo e nelle sue applicazioni. Nonostante gli sforzi di marketing della Apple per provare il contrario, non possiamo affidarci unicamente sul produttore per proteggere i nostri dispositivi mobili.

Nel momento in cui pensiamo di essere al sicuro, diventiamo vulnerabili. Per questo accesso illecito, non contava il fatto che la vittima stesse usando l'SSL. Tutto ciò che ci serviva era una vittima ignara o non preoccupata. Tendiamo a pensare che la sicurezza dei nostri sistemi - e dei nostri dati, soldi, identità online - possa poggiare unicamente su una password efficace oppure su un grosso e costoso firewall (o qualsiasi altra appliance) o un buon programma antimalware. In realtà la nostra protezione si basa su una combinazione di tali elementi. Ma la cosa più importante di cui abbiamo bisogno per proteggerci online è la consapevolezza. Dobbiamo essere consci del fatto che i malintenzionati sono online, che le vulnerabilità esistono e dobbiamo porvi rimedio o coprirle.

A volte una singola vulnerabilità sembra innocua. Se si guarda a ciascuna di esse separatamente, si può ritenere impossibile che qualcuno possa usare tale difetto per penetrare in un dispositivo. Se però si combinano insieme alcune vulnerabilità, queste possono costituire un serio rischio. Va inoltre tenuto presente che molte cose utili possono essere riprogrammate e volte a scopi illeciti. Strumenti buoni nelle mani sbagliate possono diventare armi che irrompono nei nostri sistemi.

Le vulnerabilità sono presenti nella maggior parte del software. Bisogna essere informati sugli avvisi dei produttori e installare le patch o comunque prendere delle contromisure per prevenire che i sistemi vengano messi a repentaglio. Possiamo anche trarre vantaggio dagli strumenti di sicurezza che ci avvisano del software vulnerabile nelle nostre reti.

Ringraziamenti

Grazie a Stuart McClure per il suo costante incoraggiamento e a Raul Collantes per il suo contributo in idee e opinioni.

Informazioni sugli autori

Gabriel Acevedo è ricercatore della sicurezza nei McAfee Labs di Santiago, Cile. Ricerca le vulnerabilità nuove ed esistenti nella piattaforme Microsoft Windows e Unix, nelle appliance di sicurezza e in altri sistemi. Acevedo fa parte del gruppo McAfee Vulnerability Management Content (precedentemente chiamato McAfee Foundstone®), un team globale responsabile di implementare verifiche nel software per rilevare la presenza di vulnerabilità nei sistemi informatici remoti. È inoltre a capo del Mobile Security Working Group, facente parte dei McAfee Labs, che esegue collaborazioni di ricerca in materia di sicurezza dei dispositivi mobili e incorporati.

Michael Price è l'ex responsabile senior operazioni dei McAfee Labs nell'ufficio di Santiago. Durante la sua esperienza in McAfee ha lavorato con entità esterne in Cile e America Latina, promuovendo l'eccellenza e l'innovazione tecniche. Price è ora capo architetto per iOS in Appthority, dove conduce ricerche su iOS e la sicurezza delle applicazioni.

[1] Kehrer, Paul. "Trustwave's SpiderLabs Security Advisory TWSL2011-007" (Avviso di sicurezza TWSL2011-007 SpiderLabs di Trustwave), luglio 2011. https://www.trustwave.com/spiderlabs/advisories/TWSL2011-007.txt

[2] Apple. "Informazioni sul contenuto di sicurezza dell'aggiornamento del software di iOS 4.3.5 per iPhone", luglio 2011. http://support.apple.com/kb/HT4824

[3] Apple. "Informazioni sul contenuto di sicurezza dell'aggiornamento del software di iOS 4.2.10 per iPhone", luglio 2011. http://support.apple.com/kb/HT4824?viewlocale=it_IT

[4] Percoco, Nicholas e Paul Kehrer. "Getting SSLizzard" (Ottenere SSLizzard), agosto 2011. http://defcon.org/html/defcon-19/dc-19-speakers.html#Percoco

[5] Per maggiori informazioni su certificati e autorità di certificazione, consultare http://mcaf.ee/2mjdv

[6] Bédrune, Jean-Baptiste. "Analysis of the jailbreakme v3 font exploit" (Analisi dell'exploit jailbreakme v3), luglio 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 Srl
via Fantoli, 7
20138 Milano
Italia
(+39) 02 554171
www.mcafee.com/it

41724wp_ipad-hack_0212_fnl_ETMG