Apples goto fail

Thomas Ambichl
FH Wiener Neustadt
Wiener Neustadt
111683@fhwn.ac.at

Abstrakt- Anfang des Jahres 2014 gab Apple ein Sicherheitsupdate heraus, in Form von iOS 7.0.6, welche eine Sicherheitslücke schließen soll. Dabei handelte es sich um einen Tippfehler, wodurch Malware über vermeidlich sichere Verbindungen verschickt werden konnten.

I. Einleitung

Um zu Beginn wird im nachfolgenden Absatz die Situation kurz erklärt, um einen groben Überblick über die damaligen Verhältnisse zu gewähren.
Im Februar 2014 veröffentlichte der Konzern Apple, welcher vor allem für seine Smartphone-Reihe iPhone bekannt ist, iOS 7.0.6. Dabei handelte es sich um ein Sicherheitsupdate für diverse Hardwares von Apple. Die offiziellen Patchnotes besagen, dass es Angreifern in privilegierter Position in einem Netzwerk möglich war online Daten einzulesen und zu modifizieren. Dies lag daran das Verbindungen von der eigenen Software nicht Validiert werden konnten. Die Aussage lässt vermuten, dass eine sogenannte „Man-in-the-Middle‘‘ Attacke möglich war, obwohl den Nutzer die Verbindung vom Betriebssystem als Sicher eingestuft wurde. Mittlerweile sei dieses Problem von Apple auf sämtlichen Geräten behoben worden.

II. Begriffserklärung

Bevor der Programfehler an sich unter die Lupe genommen wird, werden einige Begriffe, welche zum Verständnis des Problems wichtig sind, kurz erklärt.

A. Transport Layer Security

TLS, auch bekannt als Secure Sockets Layer (SSL), ist ein Protokoll, welches zur verschlüsselten Kommunikation verwendet wird. Es soll vor allem die Privathaltung von Daten, die zwischen zwei Computern ausgetauscht werden, gewährleisten. Bei einem Verbindungsaufbau schickt der Server hierfür zunächst ein Zertifikat an den Client. Dabei handelt es sich um ein autorisiertes Dokument, mit dem sich zum Beispiel Webseiten im Netz ausweisen können. Wenn das Zertifikat sich als korrekt erweist, benutzt der Client den „Public Key‘‘, eine vom Server zur Verfügung gestellte Zahlenfolge, um eine willkürliche Zahl, genannt „Season Key‘‘, verschlüsselt an den Server zu senden. Die Kommunikation verläuft nun über den Season Key und kann von außen nicht gelesen werden. Dieser Vorgang wird TLS-Handshacke genannt.

B. Man-in-the-Middle Attacke

Die MitM Attacke ist eine Methode, um unbemerkt Malware auf Geräten von Opfern zu installieren. Zu diesem Zwecke benutzt der Täter ein von ihm gesteuertes WLAN-Netzwerk. Wenn ein User im Netzwerk eine bestimmte Website aufruft, kann der Täter statt auf die originale Seite, den Nutzer auf eine von ihm erstellte gefälschte Variante leiten. Wenn dies der Fall ist kann der Täter, neben den von der Seite gewünschten Content, auch Malware an den User schicken. Im Normalfall kann dieses Vorgehen verhindert werden, in dem https, welches TLS verwendet, und nicht http genutzt wird, da eine gefälschte Website nicht das gewünschte Zertifikat vorlegen kann. Selbst wenn das Zertifikat einer anderen Seite herangezogen wird, kann dennoch nicht bewiesen werden, dass der Server über den „Private Key‘‘, welcher zum entschlüsseln des Public Key benötigt wird, besitzt.

III. Apples TLS-Bug

Im nachfolgenden Kapitel wird nun der Bug, der zur Sicherheitslücke führt, erklärt und wie dieser genutzt werden kann. Abschließend wird kurz gezeigt wie man als Nutzer solche Risiken vermeiden kann.

A. Der Bug

In der Version 55741 vom „SecureTransport‘‘, Apples TLS Libary, befindet sich das File „SSLProcessServerKeyExchange()‘‘ welches in Fig. 1 zu sehen ist. Dieses überprüft, ob der Private Key sich mit dem Zertifikat validieren lässt. Dafür hat es zwei Methoden, wovon die erste für TLS 1.2 benutzt wird. Sollte es sich um eine ältere Version handeln wird die Methode „SSLVerifySignedServerKeyExchange()‘‘ verwendet, zu sehen in Fig. 2. Man erkennt, dass zunächst die Prüfsumme errechnet wird und anschließend die Methode „sslRawVerify()‘‘ aufgerufen wird. Wie der Name vermuten lässt, wir hier der Key nun geprüft. Sollte alles stimmen, bekommt die Variable „err‘‘ den Wert null, was für „in Ordnung‘‘ steht. Sollte ein anderer Wert herauskommen, zündet eines der if-Statements und es wird „goto fail‘‘ aufgerufen, was nichts anderes bedeutet als, dass die Verbindung ist gescheitert. Nun steht aber unter einem if-Statement zweimal goto fail. Aufgrund dieser Schreibweise in C wird das zweite goto fail immer dann auslösen, wenn das erste nicht aufgerufen wird. In Folge dessen, überspringt das Programm sslRawVerify() und verlässt die Methode. Zusammengefast wird der der Prozess als erfolgreich gewertet, obwohl die eigentliche Überprüfung nie stattfand.

```
SSLProcessServerKeyExchange()
  -> SSLDecodeSignedServerKeyExchange()
     -> SSLDecodeXXKeyParams()
     IF TLS 1.2 -> SSLVerifySignedServerKeyExchangeTls12()
     OTHERWISE  -> SSLVerifySignedServerKeyExchange()
```

Fig. 1: Das File SSLProcessServerKeyExchange.

```
. . .
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;  /* MISTAKE! THIS LINE SHOULD NOT BE HERE */
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;

err = sslRawVerify(...);
. . .
```

Fig. 2: Die Methode zur Validierung.

B. Wie der Bug missbraucht wird

Das MitM Verfahren kann, dank des Bugs, selbst mit einer Verbindung über https erfolgreich sein. Dafür wird wieder ein eigenes WLAN-Netzwerk benötigt. Nun kann der Täter über den Server vom Nutzer verlangen Forward Secrecy, ein Verschlüsselungs-Algorithmus, zu benutzen. Entscheidend ist, dass der Täter eine ältere Version von TLS anfordert, wie etwa V 1.1. Wenn ein potenzielles Opfer jetzt eine bestimmte Website aufruft, kann der Täter ein Zertifikat vorweisen, ohne den passenden Private Key zu besitzen, da diese Validierung ja übersprungen wird. Das Opfer hat unter diesen Umständen keine Möglichkeit, den Schwindel zu bemerken. Der Bug funktioniert bekanntlich mit dem Apple Browser Safari und der Mail Applikation

C. Was unternommen werden kann

Zum einem empfiehlt es sich immer die neuste Version eines Betriebssystems zu verwenden, da solche Bugs in der Regel schnell behoben werden, falls sie bemerkt werden. Um rauszufinden welche Version eine Applikation verwendet, kann „otool‘‘, das cmd von Apple, und der Befehl -L benutzt werden. Für experimentierfreudige Programmierer gibt es auch viele inoffizielle Patches.

Des Weiteren sollte man sich nicht in fremde Netzwerke einloggen, selbst wenn sie passwortgeschützt sind. Auch bekannte WLANs können ein Problem werden, da die „Automatisch verbinden‘‘-Funktion dazu führen kann, dass man sich ausversehen mit einen gleichnamigen Netzwerk verbindet. Dagegen hilf ein VPN. Mit diesem werden unsichere Netzwerke nur zur Bildung eines verschlüsselten Pfades ins Internet verwendet und die ausgetauschten Daten bleiben unersichtlich.

Außerdem können Browser von Drittanbietern, wie Firefox oder Chrome, verwendet werden, welche eigene TLS Libaries verwenden. Es gibt aber auch Applikationen, welche den https Traffic einlesen und bei Vorkommnissen den Nutzer benachrichtigen. Bekannte Produkte sind Sophos Web Appliance und Sophos UTM.

IV. Fazit

Kurz gesagt kam es aufgrund eines Tipp- oder Kopierfehler zu einer Sicherheitslücke, welche relativ einfach zu nutzen ist. Für Nutzer bedeutet dies unter anderem, dass man sich auf Software, selbst solche von großen Entwicklern, nicht immer verlassen kann und man sich eine „zweite Meinung‘‘ von Drittanbietern mit deren Produkten einholen soll.

Für Programmierer kann dieses Beispiel als Mahnmal gedeutet werden. Heißt Code sollte immer mehrmals durchgelesen werden, um solcherlei Fehler zu umgehen. Außerdem verdeutlicht es den Wert von Tests. Das MitM Verfahren ist nämlich in der Branche bekannt. Mit einem gezielten Test wäre dieser Bug eventuell noch vor der Veröffentlichung bemerkt worden.

[1] Folien WS-SS

[2] https://nakedsecurity.sophos.com/2014/02/24/anatomy-of-a-goto-fail-apples-ssl-bug-explained-plus-an-unofficial-patch/