How to avoid this Apple Siri https cracking scenario?

Asked 13 years, 1 month ago Modified 12 years, 2 months ago

Viewed 2k times Part of Mobile Development Collective

2

After reading the post about "cracking Siri", I understand that the HTTPS traffic from the iPhone to the Siri Https server were "cracked" (decrypted) by creating:

- a custom DNS server
- a fake HTTPS server (pretending to be 'guzzoni.apple.com')
- a custom SSL certification authority

and by modifying the client (iPhone) DNS and SSL Certification Authority settings, they were able to fake the complete "environment" and decrypt the traffic.

But how could Apple (or anyone else) avoid this type of "crack"/hack?

MD ios https

Share

Improve this question

Follow

edited Nov 16, 2011 at 8:03

skaffman

403k ● 96 ● 824 ● 774

asked Nov 16, 2011 at 8:01

Timothée HENRY

14.6k ● 22 ● 98 ● 138

4 Answers

Sorted by: Highest score (default)

1

HTTPS is designed to be secure given that you trust installed certification authorities. Since anyone can willfully install a rogue/alternative certification authority on just about any device, it is a mistake from any developer to trust HTTPS to protect the data against the machine it's coming from.

Share Improve this answer

Follow

answered Nov 16, 2011 at 8:07

zneak

138k ● 46 ● 265 ● 335

Thanks, though I have two comments on your answer: 1. The concern is not that the developer trusts the data received, it is that the data has been decrypted by a hacker. 2. You did not say how to avoid this type of crack/hack.
– Timothée HENRY Nov 16, 2011 at 8:58

@tucson, I wrote this answer in hope to sweep the assumption that HTTPS should do what you are currently asking for. HTTPS provides two guarantees: parties know they deal with the person they think they're dealing with, and the communication is protected against eavesdropping. HTTPS was not designed to protect the data against the person it's coming from. – zneak Nov 16, 2011 at 16:20

@tucson, the right way to secure your protocol is to rely on authentication rather than on secrecy. Apple does it too. In order to use Siri, developers need a device identifier, which

makes the service unusable to people outside the Apple environment with a decent certainty. In the end, it doesn't really matters that people figured out the communications between iPhones and Apple's servers if they can't use it. It also didn't reveal anything to worry about. It is also arguable that HTTPS serves very well the purposes it was designed to: speech is encrypted between devices and Apple. – zneak Nov 16, 2011 at 16:25

1

After some reading, it seems the only way to avoid this type of hack (which I understand is the famous Man in the Middle attack) is to do a correct authentication. Well explained here:

> Public-key algorithms can guarantee the secrecy of a message, but they do not necessarily guarantee secure communications because they do not verify the identities of the communicating parties. To establish secure communications, it is important to verify that the public key used to encrypt a message does in fact belong to the target recipient. Otherwise, a third party can potentially eavesdrop on the communication and intercept public key requests, substituting its own public key for a legitimate key (the man-in-the-middle attack).
>
> In order to avoid such an attack, it is necessary to verify the owner of the public key, a process called authentication. Authentication can be accomplished through a certificate authority

> (CA), which is a third party that is trusted by both of the communicating parties.
>
> The CA issues public key certificates that contain an entity's name, public key, and certain other security credentials. Such credentials typically include the CA name, the CA signature, and the certificate effective dates (From Date, To Date).

So I would guess the only way to avoid such hack is to only let the client (here iPhone) use pre-determined CAs.

Share Improve this answer

Follow

answered Nov 16, 2011 at 10:56

Timothée HENRY

14.6k • 22 • 98 • 138

1 Sorry, this is not a solution. You don't need a man in the middle attack to intercept the https communication. Assuming that the iPhone is yours you can intercept the communication just hooking into the iOS functions. There is not solution for this kind of attack if you assume that one end point (i.e: the phone) is yours and you can reverse engineer it. – sw. Nov 17, 2011 at 0:39

0

I would analyze the situation starting with two facts:

1. From your iPhone's perspective, your iPhone is trusted.
2. From your iPhone's perspective, the server is untrusted.

As a first step on top of HTTPS, I would challenge the server to prove it's identity. The challenge is something like "sign my UUID, this timestamp, and this nonce." Using public key cryptography with a published key, you can verify that you're talking to the server you know and trust.

That is vulnerable to a man in the middle for observation, but the challenge is sufficiently complex that a replay attack won't work. By the time you know how to answer the challenge, you'll never see that challenge again.

I think observing Siri's protocol is a non-issue if it can't be replay-attacked, plus we've all seen it now anyhow, but the man in the middle can be blocked by using a trusted certificate authority. If *guzzoni.apple.com* isn't what it should be, the smell test fails. Some Netflix clients check with VeriSign that they're talking to the real Netflix.

That is an analysis from your iPhone's perspective, which trusts itself implicitly. Obviously, your phone is the untrusted party and Apple is the trusted party. Once iOS is jailbroken, all bets are off. There are two absolute solutions, and neither of them are technological.

1. Backport Siri to the iPhone 3GS, iPhone 4, iPad, and iPad 2. Then, nobody will need fake Siri.
2. Don't worry about jail-breakers, and just let them have their fun. Seems to be working so far.

(As a side note, there has been a long-standing replay attack vulnerability in the iOS firmware verification

process. The verification server would give you a signature based on a hardware-based token and the firmware version, which Cydia could replay at a later date. It's been corrected in iOS 5, several years laters, which leads me to believe Apple consciously chose #2.)

Share Improve this answer Follow

answered Apr 15, 2012 at 7:36

wjl

7,352 ● 2 ● 32 ● 49

0

You have three problems here. One is the problem with SSL/TLS and Public CA hierarchy. Second is the problem of insecure DNS. Third is Apple Security. We can fix two of the three with technical controls.

Nearly all the problems with SSL/TLS, DNS, Public CAs can be fixed with Public Key (or Certificate) Pinning. Pinning is "white listing" expected public keys or certificates in the application. Public Key Pinning is equivalent to `StrictHostKeyChecking` in SSH. It will solve Problem (1) and Problem (2). You can see an example of how to pin public keys in Cocoa/CocoaToauch it at Graham Lee's On SSL Pinning for Cocoa [Touch].

There are a few more problems with Infrastructure (Carriers, Handset OEMs, Interception Proxies, Untrustworthy CAs, Broken CAs, Untrustworthy Browsers, DNS, PKI Revocation, etc), and most are discussed at OWASP Mobile's mailing list: Mobile, SSL/TLS, and Certificate or Public Key Pinning. The good

news is that pinning will close most holes created by the other folks, too.

You are not going to be able to fix Problem (3). When ZonD80 did the equivalent for In-App purchases, Apple sent lawyers to perform take downs rather than placing technical controls in StoreKit to ensure it can't happen again. Here, the technical control is pinning. Its another Apple Security failure. (See Hacker Bypasses Apple's iOS In-App Purchases for details).

Two pieces of reading you might be interested in (these problems have been well known for years): PKI is broken and The Internet is Broken.

edited Oct 21, 2012 at 21:36

answered Oct 21, 2012 at 21:08

jww
102k ● 100 ● 437 ● 936