Security Now! #446 - 03-11-14

iOS Security

Link Tracking Warning!
This document was first authored in Google Docs, then Downloaded as a PDF. So, Google has thoughtfully (ha!) added "tracking" redirections to all of the links here. (I have no idea why, but that's Google.) If that bothers you, simply copy the text of the link into your browser's URL field.

This week on Security Now!

- Snowden's Live SXSW appearance
- Satoshi's true identity?
- Native Americans jumping on virtual currency.
- iOS's update to v7.1
- TrueCrypt Audit Update
- SQRL coming in 34 languages
- The deepest look yet into Apple's iOS security

Security News:

Last Week Follow-Up / Foxit Confirmation:

- Lobby Canada @lobbycanada
  @SGgrc It's true about Foxit. They've fallen in with the Conduit.com scumbags. Now I gotta uninstall from everywhere.
- @gh_m3
  @SGgrc I've abandoned Foxit years ago, it's long annoyed me. Sumatra is a great super lightweight alternative bit.ly/MOGVOw
  Robert @Really_Evil_Rob
- @SGgrc Upgraded Foxit. Installer added a Cloud component program without my knowledge. Uninstalled and switching to Sumatra.

Snowden at SXSW:

- https://www.youtube.com/watch?v=PEDj4N2teWw
- http://bit.ly/sn-snowden
- Strongest point: Crypto math works, and ONLY End-to-End Encryption is TNO.
- The 4th amendment, against illegal search and seizure, that he was sworn to uphold. Illegal search and seizure doesn't mean: seize everything but don't search through it. (The NSA was back on "The Good Wife" Sunday -- quite humorous!!)
- Draws exactly the same conclusions we have on this podcast:
  - This will result in a sea change within the industry.
  - The technical folks are furious... and have the power to chance the status quo.
  - There's nothing wrong with old school targeted surveillance -- and Snowden said: if the NSA wants to know anything about you, they can. But mass collection is flatly unconstitutional.

Who is Satoshi?

- 64-year-old physicist on the cover of Newsweek Magazine
- "Dorian" Nakamoto, living in Temple City, CA, used to be "Satoshi Nakamoto"
- According to Newsweek reporter Leah McGrath Goodman, when Dorian Nakamoto was confronted at his home before publication and asked about Bitcoin, he responded, "I am no longer involved in that and cannot discuss it. It's been turned over to other people."

  He told the AP that he was misunderstood.

  "It sounded like I was involved before with Bitcoin and looked like I'm not involved now," it quoted him as saying. "That's not what I meant. I want to clarify that."

  Newsweek magazine defended Goodman's reporting.

  "Ms. Goodman's research was conducted under the same high editorial and ethical standards that have guided Newsweek for more than 80 years," the magazine said on its website today. "Newsweek stands strongly behind Ms. Goodman and her article."

- "We will know Satoshi by his digital signatures," Garzik said in an e-mail, adding that the founder could either sign a message using his unique encryption key or make use of the Bitcoins that the creator is known to have kept.

  Analysts who looked at the Bitcoin ledger have concluded that the creator of the system owns about 1 million coins, worth over $600 million at current prices, said Jered Kenna, a San Francisco Bitcoin investor. If the owner pledged never to sell or trade them, it would help add stability to what has been a volatile market, Kenna said.

(The Verge) "Lakota" Native American indian tribe are adopting the "MazaCoin"

- MazaCoin is now the OFFICIAL CURRENCY of the seven bands that make up the Lakota Nation.
- Federal laws granting Native Americans special legal status, provide an argument for a currency totally independent of the US dollar. Native American sovereignty is legally defined over a patchwork of treaties, laws, and precedent.
- A spokesman for the Lakota said:"We're on sovereign soil so we have the right to have Bitcoin, Litecoin, MazaCoin."
- South Dakota legal counsel for the Lakota, Chase Iron Eyes, believes the federal government will push back if MazaCoin succeeds. He said:"There hasn't been a tribal nation that has declared its own currency and has mandated that that currency is used within its borders. But it's because of this pervasive, ever-present asserted dominion of the United States. They'll try to shut us down, try to cite us with law violations."
- ... a disruptive innovation, indeed.
- Native American tribes adopt Bitcoin-like currency, prepare to battle US government
- http://www.theverge.com/2014/3/5/5469510/native-americans-assert-their-independence-through-cryptocurrency-mazacoin

iOS v7.1:

- TouchID fixed? Too soon to tell.
    - Early reports from Tweeters to me have indicated that it seems both more resilient and a lot faster.
- Many UI Tweaks:
    - Phone App gets round buttons.
    - Add a contact directly from the Phone app
    - Users can now switch between light and dark keyboards from the Accessibility settings. (??)
    - More obvious caps lock
    - Built-in Icon tweaks (the green ones are flatter and stronger)
    - Changed the power-down slider
    - Siri has push-to-talk using the home button.
    - ... and she sounds more natural
    - Easier to disable the parallax effect (where wallpaper is set)
    - Calendar lists returned to the bottom of the calendar.
- CarPlay
- My own observation:
    - Fonts appear to be stronger.
    - Pinch returns to wallpaper… then fades in.

Truecrypt Audit:

- First phase audit is COMPLETE and results have been shared with the TrueCrypt development team.

Team CYMRU: SOHO Pharming

- https://www.team-cymru.com/ReadingRoom/Whitepapers/2013/TeamCymruSOHOPharming.pdf
- "Growing Exploitation of Small Office Routers Creating Serious Risks"

Miscellany:

"Influx" by Daniel Suarez.

SQRL

Translation Project -- now 34 languages and 80 volunteer translators.

- http://translate.grc.com // (Soon: grc.com/translate)

ALL user-interface design work finished... starting to write the code.

- http://bit.ly/sqrlui

SpinRite:

Jeff Harmon @harmon_jeff (10:41pm · 6 Mar 14 · Twitter for iPhone)
Thank goodness for #SpinRite and @SGgrc ! Repaired hard drive enough to pull off 350GB of photo/video to a new drive.

iOS Security / The White paper

http://images.apple.com/iphone/business/docs/iOS_Security_Feb14.pdf

The overall landscape:

- A YEAR OLD report (Feb 2013) by the soon-to-be-renamed McAfee, owned by Intel:
  - Mobile Malware: 792 samples in 2011 ...to... 36,699 in 2012.
  - 97% of those samples were designed to attack the Android platform.
  - http://www.mcafee.com/us/security-awareness/articles/mobile-malware-growth-continuing-2013.aspx

- A year later... F-Secure:
  http://www.f-secure.com/static/doc/labs_global/Research/Threat_Report_H2_2013.pdf
  - "97 percent of the mobile threats in 2013 were directed at the Android platform, which racked up 804 new families and variants," F-Secure said in its report (pdf). "The other three percent (23) were directed at Symbian. No other platforms had any threats. In contrast, 2012 saw 238 new Android threats."
  - <quote>
    "For mobile platforms, the continued dominance of the Android operating system makes it almost the exclusive target for mobile threats we've seen this period."

    "Though the relatively low number of vulnerabilities found in Android makes the operating system itself difficult to attack, this security is largely circumvented by the relative ease with which malware authors can provide their 'products' and dupe users into installing it on their own devices, with the necessary permissions to straightforwardly use the device (and the user's data) for the attacker's own benefit."

Apple & iOS: Carefully curated, closed, eco-system.

- Android:
  - Less expensive,
  - Many more hardware platforms
  - Much more freedom.
  - Much more open feeling.

- In order to pull off a closed eco-system -- in the face of massive assault -- it's necessary to build a very carefully designed security infrastructure.
- That's what iOS is.

Overall:

The design is absolutely user-Centric.

- Almost all of the imposition, that's inherent in crypto, has been carefully hidden.
- Things may sometimes seem a bit "odd"... but that slight oddness is masking a truly AMAZINGLY complex software control system.

The architecture evidences a total respect for the user's privacy and security.

- For example: The device's unique ID (UID) and a device group ID (GID) are AES 256-bit keys fused into the application processor during manufacturing. No software or firmware can read them directly; they can see only the results of encryption or decryption operations performed using them. The UID is unique to each device and is not recorded by Apple or any of its suppliers.

Apple NEVER receives anything they don't have reason to, and if they can arrange to be unable to decrypt it, that's what they do.

BUT!!!!!!! ------> To deliver ***some*** communication services, they DO keep dangerous information:

- iMessage & Facetime are "secure" but **NOT** guaranteed private, because Apple holds everyone's public keys.

Hardware enhanced security:

- Every iOS device has a dedicated AES 256 crypto engine built into the DMA path between the flash storage and main system memory, making file encryption highly efficient. Along with the AES engine, SHA-1 is implemented in hardware, further reducing cryptographic operation overhead.

Secure Boot Chain:

- ***Boot ROM*** contains Apple Root CA public key used to check the signature of anything loaded.

Software Update Security

- Device connects to Apple, sends an inventory, a nonce, and the device's UniqueID (ECID).
- Apple builds the update package, adds the ECID to it, and signs the result.
- The bundled ECID personalizes the update for the requesting device.
- ECID cannot be changed without breaking Apple's signature.
- This prevents "downgrade attacks" using older software with known problems to compromise.

Secure Enclave

- Separate on-chip co-processor.
- Has its own independent secure boot.
- Hardware true random number generator.
- Has its own UID (unique ID):
    - Inaccessible to other parts of the system.
    - NOT known to Apple.
    - "When the device starts up, an ephemeral key is created, tangled with its UID, and used to encrypt the Secure Enclave's portion of the device's memory space."
    - ("Tangling" is never elucidated, but from every usage in the document it appears to be a keyed hash, probably an HMAC, operation.)
- Extremely limited communications.
    - Interrupt-driven mailbox using shared data buffers.

Example of the Secure Enclave's usage:

The Secure Enclave is responsible for processing fingerprint data from the Touch ID sensor, determining if there is a match against registered fingerprints, and then enabling access or purchase on behalf of the user. Communication between the A7 and the Touch ID sensor takes place over a serial peripheral interface bus. The A7 forwards the data to the Secure Enclave but cannot read it. It's encrypted and authenticated with a session key that is negotiated using the device's shared key that is built into the Touch ID sensor and the Secure Enclave. The session key exchange uses AES key wrapping with both sides providing a random key that establishes the session key and uses AES-CCM [mode] transport encryption. (Counter w/CBC-MAC - authenticated encryption)

TouchID innards:

The 88-by-88-pixel, 500-ppi raster scan is temporarily stored in encrypted memory within the Secure Enclave while being vectorized for analysis, and then it's discarded after. The analysis utilizes subdermal ridge flow angle mapping, which is a lossy process that discards minutia data that would be required to reconstruct the user's actual fingerprint. The resulting map of nodes never leaves iPhone 5s, is stored without any identity information in an encrypted format that can only be read by the Secure Enclave, and is never sent to Apple or backed up to iCloud or iTunes.

Locking and Unlocking:

On devices with an A7 processor, the Secure Enclave holds the cryptographic class keys for Data Protection. When a device locks, the keys for Data Protection class Complete are discarded, and files and keychain items in that class are inaccessible until the user unlocks the device by entering their passcode.

On iPhone 5s with Touch ID turned on, the keys are not discarded when the device locks; instead, they're wrapped with a key that is given to the Touch ID subsystem. When a user

attempts to unlock the device, if Touch ID recognizes the user's fingerprint, it provides the key for unwrapping the Data Protection keys and the device is unlocked. This process provides additional protection by requiring the Data Protection and Touch ID subsystems to cooperate in order to unlock the device.

The decrypted class keys are only held in memory, so they're lost if the device is rebooted. Additionally, as previously described, the Secure Enclave will discard the keys after 48 hours or 5 failed Touch ID recognition attempts.

Hardware-enforced protection:

- The UID allows data to be cryptographically tied to a particular device. For example, the key hierarchy protecting the file system includes the UID, so if the memory chips are physically moved from one device to another, the files are inaccessible. The UID is not related to any other identifier on the device.
- Apart from the UID and GID, all other cryptographic keys are created by the system's random number generator (RNG) using an algorithm based on CTR_DRBG. System entropy is gathered from interrupt timing during boot, and additionally from internal sensors once the device has booted.

Wear-Leveling Bypass:

Securely erasing saved keys is just as important as generating them. It's especially challenging to do so on flash storage, where wear-leveling might mean multiple copies of data need to be erased. To address this issue, iOS devices include a feature dedicated to secure data erasure called Effaceable Storage. This feature accesses the underlying storage technology (for example, NAND) to directly address and erase a small number of blocks at a very low level.

File System Protection

Every time a file on the data partition is created, the data protection sub-system creates a new 256-bit key (the "per-file" key) and gives it to the hardware AES engine, which uses the key to encrypt the file as it is written to flash memory using AES CBC mode. The initialization vector (IV) is the output of a linear feedback shift register (LFSR) calculated with the block offset into the file, encrypted with the SHA-1 hash of the per-file key.

The per-file key is wrapped with ONE of several class keys, depending on the circumstances under which the file should be accessible. Like all other wrappings, this is performed using NIST AES key wrapping, per RFC 3394. The wrapped per-file key is stored in the file's metadata.

When a file is opened, its metadata is decrypted with the file system key, revealing the wrapped per-file key and a notation on which class protects it. The per-file key is unwrapped with the class key, then supplied to the hardware AES engine, which decrypts the file as it is read from flash memory.

The metadata of all files in the file system is encrypted with a random key, which is created

when iOS is first installed or when the device is wiped by a user. The file system key is stored in Effaceable Storage. Since it's stored on the device, this key is not used to maintain the confidentiality of data; instead, it's designed to be quickly erased on demand (by the user, with the "Erase all content and settings" option, or by a user or administrator issuing a remote wipe command from a mobile device management server, Exchange ActiveSync, or iCloud). Erasing the key in this manner renders all files cryptographically inaccessible.

Passcodes

By setting up a device passcode, the user automatically enables Data Protection. iOS supports four-digit and arbitrary-length alphanumeric passcodes. In addition to unlocking the device, a passcode provides the entropy for encryption keys, which are not stored on the device. This means an attacker in possession of a device can't get access to data in certain protection classes without the passcode.

The passcode is "tangled" with the device's UID, so brute-force attempts must be per-formed on the device under attack. A large iteration count is used to make each attempt slower. The iteration count is calibrated so that one attempt takes approximately 80 milliseconds. This means it would take more than 5½ years to try all combinations of a six-character alphanumeric passcode with lowercase letters and numbers.

The stronger the user passcode is, the stronger the encryption key becomes. Touch ID on iPhone 5s can be used to enhance this equation by enabling the user to establish a much stronger passcode than would otherwise be practical. This increases the effective amount of entropy protecting the encryption keys used for Data Protection without adversely affecting the user experience of unlocking an iOS device multiple times throughout the day.

Data Protection Classes

- When a new file is created on an iOS device, it's assigned a class by the app that creates it. Each class uses different policies to determine when the data is accessible.

- **Complete Protection**
  The class key is protected with a key derived from the user passcode and the device UID. When the device is unlocked the decrypted class key is discarded, rendering all data in this class inaccessible until the user enters the passcode again or unlocks the device using Touch ID.

  The Mail app implements Complete Protection for messages and attachments. App launch images and location data are also stored with Complete Protection.

- **Protected Unless Open**
  Some files may need to be written while the device is locked. For example, a mail attachment downloading in the background. This is achieved using asymmetric elliptic curve cryptography (ECDH over Curve25519). Along with the usual per-file key, Data Protection generates a file public/private key pair. A shared secret is computed using the file's private key and the Protected Unless Open class public key, whose corresponding

private key is protected with the user's passcode and the device UID. The per-file key Passcode is wrapped with the hash of this shared secret and stored in the file's metadata along with the file's public key; the corresponding private key is then wiped from memory.

As soon as the file is closed, the per-file key is also wiped from memory. To open the file again, the shared secret is re-created using the Protected Unless Open class's private key and the file's ephemeral public key; its hash is used to unwrap the per-file key, which is then used to decrypt the file.

- **Protected Until First User Authentication**
  This class behaves in the same way as Complete Protection, except that the decrypted class key is not removed from memory when the device is locked. The protection in this class has similar properties to desktop full-disk encryption, and protects data from attacks that involve a reboot. This is the default class for all third-party app data not otherwise assigned to a Data Protection class.

- **No Protection**
  This class key is protected only with the UID, and is kept in Effaceable Storage. Since all the keys needed to decrypt files in this class are stored on the device, the encryption only affords the benefit of fast remote wipe. If a file is not assigned a Data Protection class, it is still stored in encrypted form (as is all data on an iOS device).

App Security

- Only Apps written by known and recognized developers can obtain a code signing certificate whose signatures iOS will verify and allow to load and execute.
- Businesses can write in-house apps that do NOT need to go through Apple.
  - iOS Developer Enterprise Program (iDEP)
  - Obtain a "provisioning profile" to permit apps to run on devices it authorizes.
  - (Essentially allows a business to authorize its own apps.)

Process Security

- Apps are inherently "sandboxed" without any access to other apps resources, unless deliberately arranged by each end.
- Installed in a randomly named file system directory.
- ASLR is enabled for all Xcode-produced code.
- iOS uses ARM's "Execute Never" (XN) feature which restricts where code can execute.

Accessories:

When an accessory communicates with an iOS device using a Lightning connector cable, or via Wi-Fi or Bluetooth, the device asks the accessory to prove it has been authorized by Apple by responding with an Apple-provided certificate, which is verified by the device. The device then sends a challenge, which the accessory must answer with a signed response. This process is entirely handled by a custom integrated circuit that Apple provides to approved accessory manufacturers and is transparent to the accessory itself.

Accessories can request access to different transport methods and functionality; for example, access to digital audio streams over the Lightning cable, or Siri hands-free mode over Bluetooth. The IC ensures that only approved devices are granted full access to the device. If an accessory does not provide authentication, its access is limited to analog audio and a small subset of serial (UART) audio playback controls.

AirDrop

iOS devices that support AirDrop use Bluetooth Low-Energy (BTLE) and Apple-created peer-to-peer Wi-Fi technology to send files and information to nearby devices.

When a user enables AirDrop, a 2048-bit RSA identity is stored on the device. Additionally, an AirDrop identity hash is created based on the email addresses and phone numbers associated with the user's Apple ID.

When a user chooses AirDrop as the method for sharing an item, the device emits an AirDrop signal over BTLE. Other devices that are awake, in close proximity, and have AirDrop turned on detect the signal and respond with a shortened version of their owner's identity hash.

By default, AirDrop is set to share with Contacts Only. Users can also choose if they want to be able to use AirDrop to share with Everyone or turn off the feature entirely.

In Contacts Only mode, the received identity hashes are compared with hashes of people in the initiator's Contacts. If a match is found, the sending device creates a peer-to-peer Wi-Fi network and advertises an AirDrop connection using Bonjour. Using this connection, the receiving devices send their full identity hashes to the initiator. If the full hash still matches Contacts, the recipient's first name and photo (if present in Contacts) are displayed in the AirDrop sharing sheet.

When using AirDrop, the sending user selects who they want to share with. The sending device initiates an encrypted (TLS) connection with the receiving device, which exchanges their iCloud identity certificates. The identity in the certificates is verified against each user's Contacts. Then the receiving user is asked to accept the incoming transfer from the identified person or device. If multiple recipients have been selected, this process is repeated for each destination.

In the Everyone mode, the same process is used but if a match in Contacts is not found, the receiving devices are shown in the AirDrop sending sheet with a silhouette and with the device's name, as defined in Settings > General > About > Name.

The Wi-Fi radio is used to communicate directly between devices without using any Internet connection or Wi-Fi Access Point.

iMessage -- A bit of misdirection here:
<quote> Apple iMessage is a messaging service for iOS devices and Mac computers. iMessage supports text and attachments such as photos, contacts, and locations. Messages appear on all of a user's registered devices so that a conversation can be continued from any of the user's devices. iMessage makes extensive use of the Apple Push Notification Service (APNs). Apple does not log messages or attachments, and their contents are protected by end-to-end encryption so no one but the sender and receiver can access them. Apple cannot decrypt the data.

<quote> When a user turns on iMessage, the device generates two pairs of keys for use with the service: an RSA 1280-bit key for encryption and an ECDSA 256-bit key for signing. For each key pair, the private keys are saved in the device's keychain and the public keys are sent to Apple's directory service (IDS), where they are associated with the user's phone number or email address, along with the device's APNs address.

How iMessage sends and receives messages
Users start a new iMessage conversation by entering an address or name. If they enter a phone number or email address, the device contacts the IDS to retrieve the public keys and APNs addresses for all of the devices associated with the addressee. If the user enters a name, the device first utilizes the user's Contacts to gather the phone numbers and email addresses associated with that name, then gets the public keys and APNs addresses from the IDS.

The user's outgoing message is individually encrypted using AES-128 in CTR mode for each of the recipient's devices, signed using the sender's private key, and then dispatched to the APNs for delivery. Metadata, such as the timestamp and APNs routing information, is not encrypted. Communication with APNs is encrypted using TLS.

Siri
In order to facilitate Siri's features, some of the user's information from the device is sent to the server. This includes information about the music library (song titles, artists, and playlists), the names of Reminders lists, and names and relationships that are defined in Contacts. All communication with the server is over HTTPS.

When a Siri session is initiated, the user's first and last name (from Contacts), along with a rough geographic location, is sent to the server. This is so Siri can respond with the name or answer questions that only need an approximate location, such as those about the weather. If a more precise location is necessary, perhaps to determine the location of nearby movie theaters for example, the server asks the device to provide a more exact location. This is an example of how, by default, information is sent to the server only when it's strictly necessary in order to process the user's request. In any event, session information is discarded after 10 minutes of inactivity.

The recording of the user's spoken words is sent to Apple's voice recognition server. If the task involves dictation only, the recognized text is sent back to the device. Otherwise, Siri analyzes the text and, if necessary, combines it with information from the profile associated with the device. For example, if the request is "send a message to my mom," the relationships and names that were uploaded from Contacts are utilized. The command for the identified action is then sent back to the device to be carried out.

iCloud

Appears to be ALMOST completely solid.

Keychain syncing:
When a user enables iCloud Keychain for the first time, the device establishes a circle of trust (which will exist among devices owned by the individual) and creates a syncing identity for itself. A syncing identity consists of a private key and a public key. The public key of the syncing identity is put in the circle, and the circle is signed twice: first by the private key of the syncing identity, then again with an asymmetric elliptical key (using P256) derived from the user's iCloud account password. Also stored with the circle are the parameters (random salt and iterations) used to create the key that is based on the user's iCloud password.

The signed syncing circle is placed in the user's iCloud key value storage area. It cannot be read without knowing the user's iCloud password, and cannot be modified without having the private key of the syncing identity of its member.

But… this is the ONLY reference to Apple use of the NIST P-256 elliptic curve… and both Dan Bernstein and Bruce Schneier now declare ANY use of the NSA/NIST curves unsafe.

http://safecurves.cr.yp.to/index.html
http://www.hyperelliptic.org/tanja/vortraege/20130531.pdf

Jerry Solinas at NSA