Security Now! #447 - 03-18-14

iOS Security, Pt.2

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!

- More on the coming "XP Armageddon"
- Fun at Vancouver's 14th annual CanSecWest conference.
    - PwnToOwn
    - A disturbing revelation that the so-called "Early Random" pseudo-random number generator newly devised for iOS v7 is a little more Pseudo than we would like!
- Cloud Storage costs plummet.
- A 10-point plan to thwart the NSA.
- SQRL progress report
- Part 2 of our deep dive into the iOS Security white paper.

Security News:

Dvorak on The end of Windows XP

- Killing Windows XP Wastes Billions
- http://www.pcmag.com/article2/0,2817,2454844,00.asp
- John notes:
    - Approximately 500 Million (half a billion) XP users.
    - 29 percent of the computers in the world.
    - Many do not want to upgrade to anything new. "They are happy campers."
- John writes: "Upgrading the Microsoft OS is a needless exercise in agony. I'd still be using an XP machine for my podcasting efforts if the machine itself had not crapped out. What's the point of changing? For prettier icons?" <un-quote>
- John's right: Given the incredible anxiety that's being created by the end of the drip, drip, drip IV update feed... there's money to be made. "I'd rather fight than switch."

Five steps for XP usage past April 8th:

- Run as a "Standard User"
- Remove Flash and Java.
- Use Chrome or Firefox, never IE. (XP can't run the later more secure IE's.)
    - Remember: updates are NOT stopping for Flash, Java, Firefox, Chrome,.etc.... ONLY for the core XP OS.
    - MOST of the vulnerability updates are now in IE and Office, not in the underlying OS.
- Use "LibreOffice" instead of MS Office.
- Behave yourself... Don't do dangerous things.
    - (Embedded systems don't surf the web...)

"95% of ATMs could face hacking threat"

- http://archive.azcentral.com/business/news/articles/20140311atms-facing-tech-deadline.html
- <quote> Banks and other businesses have less than a month to get their plans in place before the computer operating system powering about 95 percent of the country's ATMs becomes vulnerable to hackers and computer viruses.
- ***Not true.***
- Ken Colburn, president of Data Doctors: "If you're still running your equipment on Windows XP, you're open to a lot of new threats. All of the devices out there still running Windows XP have to get patched up and changed fairly quickly, or they are going to be exposed to hacks."
- Colburn said, "There's still a large number of people out there that just don't realize how big a security threat this is," he said. "And after April 8 ... these hackers can come knocking and you're going to be defenseless."
- Scott Kinka, chief technology officer of Evolve IP, wrote last Tuesday, in an article for ATM Marketplace titled "One Month From Today: XP Armageddon." /he wrote/ "Surprisingly, only 15 percent of financial institutions are expected to react before the April 8 cutoff, according to a recent ATM industry association survey."

CanSecWest 2014 security conference in Vancouver --> Pwn2Own

- The 14th annual CanSecWest conference.
- March 12-14 2014 at the Sheraton Wall Centre hotel in downtown Vancouver, British Columbia.
- All major browsers fall during second day of Pwn2Own hacking contest
    - http://www.pcworld.com/article/2108560/all-major-browsers-fall-during-second-day-at-pwn2own-hacking-contest.html
    - Zero-day exploits against Chrome, IE, Safari, Forefox, Flash Player and Adobe Reader.
    - Three different successful attacks against FF on the first day, another on the second day.
    - A French team from Vupen hacked Google Chrome, exploiting a use-after-free vulnerability that affects both the WebKit and Blink rendering engines. The researchers then bypassed Chrome's sandbox protection to execute arbitrary code on the underlying system.
    - Another anonymous researcher presented a Chrome remote code execution exploit Thursday, but the contest judges declared it only a partial win because some details of the hack were similar to those of an exploit presented earlier at Google's own hacking contest that runs in parallel to Pwn2Own.
    - Another researcher with the Chinese Keen Team, combined a heap overflow vulnerability with a sandbox bypass to achieve remote code execution in Safari. He and a fellow researcher then demonstrated a remote code execution exploit for Adobe Flash Player.
    - The prizes won during the second and final day of the competition put the total contest payout to a record $850,000, not including charitable donations or the value of the test laptops won by the researchers after their successful hacks.

- Firefox Was the Most Attacked & Exploited Browser At Pwn2own 2014
    - "Though IE, Chrome and Safari were all attacked and all were exploited, no single web browser was exploited at this year's Pwn2own hacking challenge as Mozilla Firefox. A fully patched version of Firefox was exploited four different times by attackers, each revealing new zero-day vulnerabilities in the open-source web browser. When asked why Mozilla was attacked so much this year, Sid Stamm, senior engineering manager of security and privacy said, 'Pwn2Own offers very large financial incentives to researchers to expose vulnerabilities, and that may have contributed in part to the researchers' decision to wait until now to share their work and help protect Firefox users.' The Pwn2own event paid researchers $50,000 for each Firefox vulnerability. Mozilla now pays researcher only $3,000 per vulnerability."

Weak Random Number Generator Threatens iOS 7 Kernel Exploit Mitigations

- http://blog.azimuthsecurity.com/2014/03/attacking-ios-7-earlyrandom-prng.html?m=1
    - http://mista.nu/research/early_random-paper.pdf
    - http://mista.nu/research/early_random-slides.pdf
- http://threatpost.com/weak-random-number-generator-threatens-ios-7-kernel-exploit-mitigations/104757
- "Early Random PRNG"
- Kernel Vulnerability Exploitation Mitigations:
    - Logical-to-Physical map randomization
        - Randomizes the location of physical memory within much greater 64-bit logical memory space.
    - Stack check guard
        - Function prologue places a cookie on the stack.
        - Function epilogue verifies the stack cookie.
        - Stack cookie created at boot time.
    - Zone cookies and factor
    - Kernel map randomization (ASLR - Address Space Layout Randomization)
    - Pointer Obfuscation
    - Also used to seed the Yarrow CSPRNG.
- iOS v6 focused upon security improvements to prevent jailbreaking
- These guys concluded their previous iOS v6 examination presentation saying:
    - "iOS 6 mitigations significantly raise the bar"
    - Many of the old tricks don’t work
    - A variety of bugs likely to be (reliably) unexploitable now
- Really really horrible PRNG design.
    - Simple Linear Congruential PRNG
    - Seeded from iBoot by sampling the low bit of the processor clock, with delay, until 32 bits have been obtained.
    - Low bits are discarded.
    - To make up for small output, FOUR states are combined.
- The BIG problem:
    - The OS LEAKS this weak randomness all over the place.
    - The PRNG state can be tracked backward to the seed and forward forever…
    - “Scrambling bits” by shifting and XORing is weak obfuscation. NOT SECURE!!

Google changes the game with cloud storage pricing

- http://readwrite.com/2014/03/17/google-drive-pricing-plans-drop-cloud-rivals-breakdown
- http://bit.ly/sn-google
- 15gb - Free
- $2/mo. (previously $5) - 100gb
- $10/mo. (previously $50) - 1,000gb (1tb)
    - Seagate Desktop HDD 4 TB SATA 6Gb/s NCQ 64MB Cache 3.5-Inch Internal Bare Drive
- $155, so < $40/tb.
- $100/mo. - 10tb+
- SN will be revisiting the topic of TNO Cloud Storage solutions with an updated Cloud Storage Roundup.

Wired: A 10-Point Plan to Keep the NSA Out of Our Data

- Kim Zetter: http://www.wired.com/threatlevel/2014/03/wishlist/
- 1) End-to-end encryption.
- 2) Bake user-friendly encryption into products from the get-go.
- 3) Make all web sites SSL/TLS.
- 4) Enable HTTP Strict Transport Security.
- 5) Encrypt data-center links.
- 6) Use perfect forward secrecy.
- 7) Secure software downloads.
- 8) Reduce storage/logging time.
- 9) Replace Flash with HTML5.
- 10) Fund a global account to support community audits of open source code.

iOS v7.1:

- TouchID fixed? Reports generally positive.
    - I cannot tell since I'm too in love with my Type Keyboard.

SQRL

Translation Project:

- https://crowdin.net/project/sqrl
- Last week: 34 languages and 80 volunteer translators.
- This week: 48 languages and 213 volunteer translators.
- http://translate.grc.com // (Soon: grc.com/translate)

Working on the code foundation:

- First internationalized code I've ever written,
- Proceeding carefully and converting much to multibyte -- unicode.

A logo for SQRL:

- Purchased an unrestricted use license.
- Lower resolutions are blurry.

SpinRite:

Christian Alexandrov
Location: Sofia City, Bulgaria
Subject: Short Spinrite Testimonial
Date: 16 Mar 2014 02:04:48
:
After my HDD failed on me more that year ago, I used spinrite to bring it back to the world of living. After that, I started running spinrite regularly, once every month on level 4. Now my HDD is stronger than ever, and spinrite helped my drive to locate and put it's weak sectors out of use, leaving only strong sectors in use. My hard drive is stronger and healthier than it was when it was new. Running spinrite regularly keeps my drive in good shape. No hdd problems ever since.

---

iOS Security / The White paper Part 2

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

Last Week Recap:

The overall landscape:

- 97% of mobile malware is on Android, NOT because the OS is vulnerable, but because the Androis ecosystem is, by design, open.

- Quoting from the most recent McAfee Mobile Malware Report:
  Android Malware: Backdoors, Exploits & Spyware
  - The Android platform continues to make up the bulk of malware targets, representing 97% of total mobile malware. McAfee Labs researchers are tracking a range of mobile malware targeting these devices, including backdoors that enable attackers to gain control of a smartphone, new mobile exploits, and spyware.

- **Mobile Backdoors**
  Attackers love it when users install malicious apps that let the bad guys gain complete control of victims' phones, so it's no wonder that mobile backdoors remain popular with attackers. Here is how a few of these apps operate:
  - **Android/FakeLookout.A** is a mobile backdoor that pretends to be an update to antivirus software. In reality it hands control of a phone to an attacker. It's designed to steal and upload text messages and other files to the attacker's server.
  - **Android/GinMaster.A** is a mobile backdoor that uses a root exploit to gain further access to a user's phone. It posts a number of pieces of identifying information to the attacker's server and accepts commands from the attacker.

    - **Android/Citmo.A** is SMS-forwarding malware that sends mobile Transaction Authorization Numbers (mTANs), the secret codes sent via text message to a smartphone to verify that a user is logging in online. When a user has inadvertently downloaded Android/Citmo.A, the attackers will log in and wait for the bank to send an mTAN. Your infected phone will immediately forward the mTAN to the attackers, allowing them to log in to your system and potentially get access to financial accounts.

- There is **TREMENDOUS** penetration pressure on the iOS world, but malware has a very difficult time because, by design, the Apple ecosystem is closed.

The challenge:

- CLOSING a large, powerful and flexible digital ecosystem, such as iOS, is incredibly difficult. To do so takes:
    - Dedicated hardware support to protect the system during boot, before the software is able to protect itself.
    - An obsessive focus on and concern for security… because the bad guys are obsessively focused upon finding and exploiting ANY weakness.

Overall:

- The design is absolutely user-Centric
    - Almost all of the imposition, that's inherent in crypto, carefully hidden.
- The architecture evidences a total respect for the user's privacy and security.
    - Apple doesn’t receive ANY piece of information that isn’t truly necessary for the delivery of the service. NOTHING gratuitous.
    - If they can arrange to be unable to decrypt it, that's what they do.

Last week:

- Secure Boot Chain (everything digitally signed).
- Software update security (custom per-device updates).
- Secure Enclave
    - Security co-processor with strictly limited communications.
    - Holds all of the crypto master keys -- unique per device -- and unable to expose them.

Continuing This Week:

Hardware-enforced protection:

- The UID (which NEVER leaves the device and which is unknown to Apple) allows iOS data to be cryptographically tied to a particular device so that even IF the keys were divulged, ONLY that device can use them. For example, the key hierarchy protecting the file system is wrapped (encrypted) by the UID, so if the memory chips are physically moved from one device to another, the files are inaccessible.
- Apart from the UID (unique device) and GID (group), 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.

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, the Secure Enclave will discard the keys after 48 hours or 5 failed Touch ID recognition attempts.

NOTES:

- After a reboot, users MUST re-enter their full passcode… not because Apple wants to force that, but because there's no choice.
- The quality of the user's passcode DIRECTLY AFFECTS the true security of the device's locked state.
- For MAXIMUM long-term security:
    - Yes a good strong passcode.
        - (Complex, numeric only, passcodes only show the 10-key pad)
    - Then reboot or power down the device, which flushes RAM.

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 generates a new, random, 256-bit key (called 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) for the CBC block cipher mode 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. (This is done to random access into the file.)

The per-file key is wrapped with ONE of several **class keys**, depending on the circumstances under which the file should be accessible. The wrapped, per-file key, is stored in the file's metadata.

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

The file system key:

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.

(This is why the entropy of the user's passcode is important.)

The passcode is "tangled" with the device's Unique ID (UID), so brute-force attempts must be performed **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 locked 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 UniqueID (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.

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