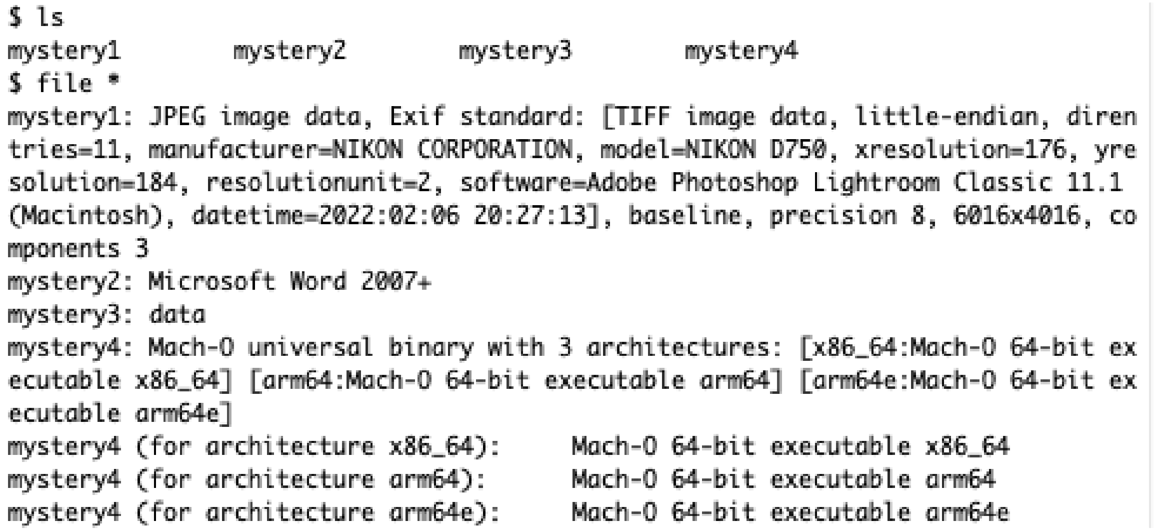
Mobile Device Security

The ‘file’ Command



```
$ ls
mystery1        mystery2        mystery3        mystery4
$ file *
mystery1: JPEG image data, Exif standard: [TIFF image data, little-endian, diren
tries=11, manufacturer=NIKON CORPORATION, model=NIKON D750, xresolution=176, yre
solution=184, resolutionunit=2, software=Adobe Photoshop Lightroom Classic 11.1
(Macintosh), datetime=2022:02:06 20:27:13], baseline, precision 8, 6016x4016, co
mponents 3
mystery2: Microsoft Word 2007+
mystery3: data
mystery4: Mach-O universal binary with 3 architectures: [x86_64:Mach-O 64-bit ex
ecutable x86_64] [arm64:Mach-O 64-bit executable arm64] [arm64e:Mach-O 64-bit ex
ecutable arm64e]
mystery4 (for architecture x86_64):    Mach-O 64-bit executable x86_64
mystery4 (for architecture arm64):     Mach-O 64-bit executable arm64
mystery4 (for architecture arm64e):    Mach-O 64-bit executable arm64e
```

The World Turned Upside Down

- The iPhone debuted in June, 2007
- In other words, it is less than 15 years old—but has changed the world
- The security implications are, umm, challenging

Phone Security

- Phones are generally very secure
- iPhones used to be much better than Android, but Android is catching up
- iPhone probably still has an edge in encryption technology and security features—but as usual, bugs in the OS' security perimeters are more important
- Phones are probably more secure than desktop OSs, though Windows 10 and 11 are excellent

The Usual Questions

- What are we trying to protect?
- Against whom?
- Both answers vary a lot, because our threat model depends on the applications involved
- Also: the answers can be very different for iPhones and Android

Attackers

- Phones cannot resist an attack by an intelligence agency
- Most are pretty good against ordinary attackers
- Cryptographic flaws rarely matter in the real world, unless you're being targeted by intelligence agencies

Players

- Phone software developers
- Phone hardware developers
- Mobile phone carriers
- App developers
- Server site developers

Apple iPhone Players

- Apple controls the software
- Apple controls the hardware
- Carriers have little or no role
- Apple controls its app store, and users can’t (easily) install apps from other sources
- Servers are independent of Apple, but are constrained by what Apple will permit apps to do

The Android Players

- Google develops the base software
- Many parties—Samsung, Google, HTC, many more—manufacture handsets and add their own software
- Carriers often have a say (and add their own code); software updates often go through carriers
- Many app stores, often with different policies; installing outside apps is relatively easy

Security Consequences: iPhone

- iPhones patches come from one party, Apple; installing updates is generally easy
- iPhone patch uptake rate is generally pretty high (though Apple has had quality control problems of late)
- Phones are supported for about five years
- (Phone battery lifetime is 2–3 years)

Security Consequences: Android

- Patches from Google have to be filtered through the manufactuer and the carrier
- Consequence: it takes a long time to get patches out
- Consequence: not all vendors release patches promptly
- Shorter support lifetime
- Consequence: many unpatched phones
- Consequence: Google has moved more of the code to its app store, to permit more rapid update of applications, independent of the base OS
- Consequence: Google is now selling many more phones of its own
- But—there are other app stores

More Consequences

- Apple could push out security features faster—it didn't rely on others adopting them or making fast enough hardware
- Android had to support a wide range of hardware capabilities; this hurt the standard deployment of, e.g., encryption
- But—iOS is a monoculture; one hole affects many products
- A flaw in a Samsung Android phone may not affect an HTC phone
- Apple's tight controls have limited the development of memory inspection tools—good for hindering ordinary hackers, but not a barrier to intelligence agencies, and third parties can't help as much with analysis

Structural Issues

- Many of the same issues as IoT
- But—relatively short ownership; phones generally replaced every few years
- Users want new features and more storage, batteries age, newer operating systems often too slow on older hardware
- The major vendors are pretty stable
- This limits some of the economic impact

Attack Scenarios

- Hackers trying to take over a phone (why?)
- Spies trying to take over a phone (why?)
- App writers trying to build secure code, especially client/server
- Users trying to get at private app data
- Thieves trying to get at private app data
- Law enforcement trying to break through encryption?

Phone Security Features

- PIN or biometric unlock
- Android supports movement pattern unlock
- Lock-out and remote wipe
- Sandbox
- Flash memory encryption
- Secure coprocessor

Note well: these defenses address different threats

Unlocking the Phone

- Generally secure enough
- Phone biometric unlock is extremely convenient
- It’s weak against some threat models—children, abusive partners—but strong against others
- Short PINs seem weak, but if accompanied by a lockout feature (and remote wipe) can be strong enough

What Do Locked Phones Protect?

- Primarily: defense against thieves, and against phones left lying around
- Protects owner against: compromise of private information (photos?) and misuse of sensitive apps
- Some apps give users the option of trusting the phone unlock; others insist on their own authentication
- Newer: owner lock—prevents resale of stolen phone
- (Robbers' countermove: "unlock this phone and reset the security or else!")

PINs and Encryption

- Modern phones encrypt flash memory
- Where does the key come from?
- Crucial question: what are the attacker's powers?

Attacker Powers

- Try typing the PIN or pattern manually
- Smudge attacks
- Hardware attacks on the flash chips

Manual PIN/Pattern Entry

- Slow
- Many folks pick bad PINs, so there's some likelihood of success
- But: phones implement delays and retry limits
- Without knowledge of the target; the odds of a successful attack are low if the phone owner is reasonably careful

Smudge Attacks

- When you touch your phone, your fingers leave a smudge on the glass
- It's pretty easy for a camera and suitable software to find these smudges
- Some experiments show a 92% success rate—and smudges are remarkably persistent
- (Could a telephoto lens pick up smudge patterns at a distance? Could a surveillance camera pick up folks entering their PINs?)

Hardware Attacks

- If you pop out the flash chips, you’re not constrained by software limits like retry limits
- The same is likely true if you can load new firmware into the phone
- Serious attackers, e.g., law enforcement with confiscated phones, can try such things

Flash Mirroring

- Pop out the flash memory
- Copy it to other flash chips (after analyzing the wiring and communications protocols)
- Try a number of PINs
- When the retry limit is reached, restore the flash from backup
- (Done experimentally for the iPhone 5C at the University of Cambridge)

Newer iPhones

- Enter the Secure Enclave
- At manufacture time, the phone’s Secure Enclave generates a random 256-bit AES key known as the UID
- On power-up, the PIN is combined—via a deliberately slow algorithm—with the UID
- This is used to decrypt the “key bag”, a set of random keys used to encrypt flash
- To wipe the phone, just destroy the key bag
- Only the key bag has to be reencrypted if you change your PIN
- Popping out the flash for hardware guessing doesn’t help—the UID is necessary to decrypt flash

The Secure Enclave

- Separate co-processor
- Can do encryption, random number generation, etc.
- Handles PINs and biometrics
- Enforces retry limits
- No way to get at the UID—can only use it for encryption and decryption, and can see the results, not the key
- (Newer Macs have similar abilities via the T2 chip)
- PIN entry required to update phone firmware

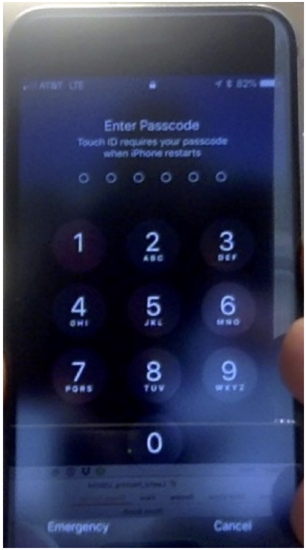
But. . .

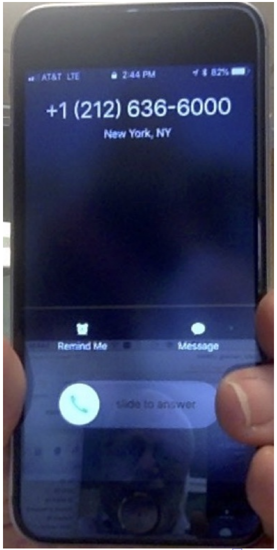
- Not everything is encrypted
- More precisely, not everything is encrypted with a key derived from the UID and PIN
- iOS has different classes of storage, encrypted with different keys; only the most sensitive data is encrypted with a UID+PIN-protected key
- Why? To permit some phone function at power-up

The iPhone Attack Surface

- Because not all of flash is encrypted with a UID-derived key, many functions of the phone are available after power-on but before PIN entry
- This means that the attack surface is relatively high
- But you still can’t get at the UID

Receiving a Call at Powerup





Hacking iPhones

- Multiple companies have serious expertise in hacking iPhones
- They sell to law enforcement and intelligence agencies—and sometimes governments in nasty countries
- Why? Law enforcement claims they need to unlock seized phones to solve crimes
- Phone encryption is a major point of tension between Apple and law enforcement

Phone Sandbox

- Both iPhones and Android phones require apps to be signed by an app store
- (On Android, it’s easy to turn this off or to switch stores)
- This gives Apple and Google (some) control over apps and what APIs they use
- Considerably more restrictive than classical Unix permissions
- Semantic settings, e.g., “Contacts”, “Make a call”, “Location”
- Strong separation between apps

App Protection

- Apps often want protection from the user of the phone
- Example: books or movies with digital rights management (DRM) restrictions
- Example: prevent disclosure of, e.g., passwords or keys
- Example: prevent user from doing sensitive things, e.g., a child buying apps or doing in-game purchases

App Protection

- Fundamental protection: sandbox—can't load an app to extract another app's data
- Apps can ask for extra authentication or invoke platform authentication
- Apps should—but don't always—use the platform's secure storage

Client/Server Apps

- Many apps are front ends for a server
- More, and friendlier, functionality than a web interface
- (The original iPhone had *no* app store—Steve Jobs felt that web apps were sufficient. . .)
- Many challenges

Mobile App Crypto

- Setting up a point-to-point encrypted circuit should be easy—we've had SSL or TLS for >25 years
- It turns out it's not easy to do it properly
- In one study, "87% of Android apps and 80% of iOS apps analyzed by Veracode were found to have cryptographic issues." (Smith, 2015)
- Basic checks were often missing

Sample Failures

- Didn’t verify the name in a certificate
- Didn’t verify the CA
- Didn’t verify the signature
- Didn’t check for expiration
- Didn’t check for revocation
- “We were able to capture credentials for American Express, Diners Club, Paypal, bank accounts, Facebook, Twitter, Google, Yahoo, Microsoft Live ID, Box, WordPress, remote control servers, arbitrary email accounts, and IBM Sametime, among others.” (Fahl et al., 2012)

UI Issues

- Phones have limited user interfaces compared with “real” computers
- How do you verify the certificate of a web site? (Apple is trying to address some of this by insisting that apps use its encrypted connection API)
- Entering long, complex passwords is painful—and autocorrect isn’t helpful if you’re trying to type `cw@*pKEotWd@{3Y`

BYOD: Bring Your Own Device

- Most people carry smartphones today
- Many people want to use them for work purposes, especially email
- This means that employee-owned phones live both inside and outside of the corporate firewall
- It's cheaper for the company not to buy phones, and people don't like carrying two devices
- But—is this a good idea?

Phone Security

- As noted, phones are pretty secure
- Reasonable polices aren't burdensome. For example, CUIT requires that phones used for university business are password-protected, have autolock, encrypt their data, have limited PIN retry, and have remote wipe
- It may be worthwhile to use a separate email app, to prevent autocomplete from helping send corporate data to the wrong people
- And: is location within the corporate facility sensitive?

Attacking iPhones: The Real World

- In the real world, sophisticated attackers have been able to attack iPhones
- Two categories:
  - Phone present: Mobile Device Forensic Tools (MDFTs)
  - Phone absent: remote/over-the-air hacking
- Interesting questions: legitimacy; cost; time to do the extraction; user awareness
- Most important: *how*?

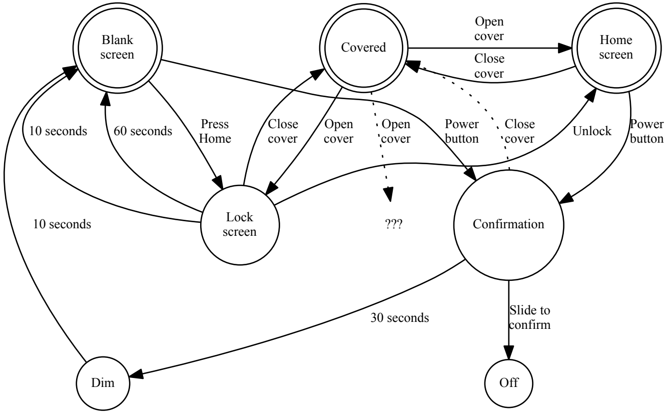
Phone Present: Scenarios

- Who is doing the unlock?
- Law enforcement? Do they have a warrant? (Generally required under US law.)
- Border agents? (Under US law, probably no warrant needed)
- The (corporate?) owner of the phone?
- How long does a search+data extraction take? Can it easily be done surreptitiously?

Phone Present: Attack Surface

- Anything over the air
- The screen
- The USB port
- The latter is probably the most important, because there’s a huge amount of complexity in dealing with USB connections

Even Handling an iPad Cover is Complex



What happens if the cover is closed at an unexpected time?

Phone Absent: Scenarios

- Law enforcement? Do they have a warrant? (Generally required under US law.)
- Intelligence agencies? Who are the targets?
- Secret police? Who are the targets?

Phone Absent: Attack Surface

- Anything over the air
- Large attack surface—*if* you know the version of iOS, the applications running on it, and their version
- ☞ Most phones will be up to date—Apple’s easy patching can work against the defender!
- Crucial issue: is there any need for user interaction?

How it’s Done

- We don’t know in detail
- Both forms of attack almost certainly involve bugs in assorted services
- Apple tries to find and close these holes, which means that a major ongoing effort for these companies is finding new holes and keeping them in reserve

Is the Secure Enclave Safe?

- What is the internal attack surface?
- ☞ The more functions it performs, the greater the attack surface—you can't put everything into the enclave
- iPhones are probably more secure before first unlock after power-up, when there's no PIN to create the decryption key
- But—is there a bug that bypasses the retry limit?

Privacy Issues

- “Modern cell phones are not just another technological convenience. With all they contain and all they may reveal, they hold for many Americans ‘the privacies of life’” (U.S. Supreme Court, Riley v. California, 134 S.Ct. 2473 (2014))
- How much is available to apps?
- The sandbox helps—but what permissions do the apps request? Why? What do users do?

Example: A Flashlight App

- A popular Android flashlight app requested permissions for precise user location, access to user’s contacts, send SMS messages, permission to directly call phone numbers, permission to reroute outgoing calls, access to camera, record audio via microphone, read/write contents of USB storage, read phone status and identity (Cleary, 2018)
- It had features that used all of those permissions—but what else did it do with the information?
- Apps that use Google or Facebook for authentication leak much more

Location Privacy

- Many apps request location information for legitimate reasons: weather forecasts, navigation, geofenced sports broadcasts, etc.
- However: the apps use location for advertising, and collect and sell the data
- Privacy policies are obscure, often deliberately

Platform Privacy

- Android phones send some data to Google
- ☞ Some of this is changeable by the user, but most people don’t know to look for the setting
- iPhones send some data to Apple, but Apple says that they deidentify it and delete it after a few months
- Anything based on machine learning, including navigation and speech recognition, *requires* lots of data; vendors collect it because they really need it to improve their service
- But there are techniques, e.g., differential privacy, to limit the privacy impact

Loss of Data

- What about availability—the availability of user data?
- What happens to a user’s data if the phone is destroyed (the “mud puddle” test)?
- Which is more important, protecting the data from compromise or recovery in event of damage?
- This is app-specific

Protecting Data from Compromise

- Store the data only on the phone
- But—if stored elsewhere, securely store the encryption key only on the phone
- Is the platform's authentication strong enough? Are other forms of authentication usable enough?

Protecting the Data From Loss

- The data *must* be stored elsewhere, encrypted
- You need some sort of secure storage for the key
- Apple has an elaborate scheme based on AppleIDs, where one device on an account is required to authenticate any new devices
- Use of multiple HSMs to impose retry limits on remote password guesss
- Apple itself *cannot* get at your encrypted keys
- But—could they add a new public key to your account, to let them read new data? Some say it seems possible; others say no

Writing Secure Mobile Applications

- Collect minimum data
- Encrypt connections, and encrypt them properly
- Use your platform’s secure storage APIs
- Demand authentication before doing dangerous things
- Maybe give the user the ability to use platform authentication, e.g., a biometric—but maybe not
- (You’re better off using the platform, since it’s better engineered for usability, but think about children using their parents’ phones)
- Think about lost credentials

Questions?

(Northern cardinal, Morningside Park, November 21, 2021)