# Pwing Apple Watch

Max Bazaliy

June 23, 2017

MOSEC

## About me

- Security researcher at Lookout
- Pegasus malware lead researcher
- Software and hardware exploitation
- Fried Apple team co-founder
- Made a various jailbreaks for iOS

# What is Apple Watch ?

- Released in 2015
- Apple S1/S2 processor
- ARMv7k 32 bit architecture
- 512 MB RAM
- WatchOS

# Apple Watch security

- Secure boot chain
- Mandatory Code Signing
- Sandbox
- Exploit Mitigations
- Secure Enclave Processor (S2)
- Data Protection

## Possible attack vectors

- Malformed USB decriptor (special cable)
- Malformed email, Message, Photo, etc
- Application extension based

## Attack plan

- Leak kernel base
- Dump whole kernel
- Find gadgets and setup primitives
- Disable security restrictions
- Run ssh client on a watch

# Bugs of interest

- CVE-2016-4656 - osunserialize bug
- CVE-2016-4669 - mach_port register

- CVE-2016-7644 - set_dp_control_port
- CVE-2016-2370 - voucher extract recipe

## Leaking kernel slide

- CVE-2016-4655
- CVE-2016-4680

- OSNumber with high number of bits
- bcopy with attacker controlled length
- kernel stack memory leaked

# CVE-2016-4656

- UAF in OSUnserializeBinary
- OSString object deallocated
- retain() called on deallocated object
- Fake object with fake vtable -> code exec
- kernel dump required
- Bonus:we can deref any address via vtable

# Dumping kernel as OSString

- No WatchOS kernel dumps in public
- No keys for WatchOS 2.x kernels
- Idea: read kernel as OSString chunks
- vtable offset required to fake OSString
- vtable stored in __DATA.__const in kernel

## Getting vtable - __Data.__const

- __DATA.__const address is in Mach-O header
- kernel base + 0x224 == __DATA.__const
- Deref and branch via fake table

# Getting vtable - next code trick

- vtable ptr is first 8 bytes of a on object
- OSString size is 0x20 (64 bit)
- retain() off is vtable start + 0x20 (64 bit)
- Next node ptr as deallocated object vtable
- retain() - out of bounds to next code
- If next node is OSString - branch vtable

# Getting vtable - next code trick

- OSString vtable reference is OSUnserialize 😀
- We can deref any address as fake vtable ptr

# Getting vtable - dump over panic

- Crash in OSUnserializeBinaryXML
- Get LR register value from panic
- Use fake vtable to deref LR value
- Get panic from a watch
- We just dump 4 bytes of a kernel 😀

## Getting vtable - dump kernel

- Use address to leak as vtable_addr - 0x10
- We need to tune by retain() offset in vtable
- Crash and get panic log
- Copy panic from Watch to iPhone and Mac
- Parse panic log, grab 4 bytes of a kernel
- Update address with +/- 4 bytes delta

# Next step - full kernel dump

- Now use fake OSString obj to read kernel
- Read data back to userland

- Leak kernel header, calculate kernel size
- Dump full kernel to userland by chunks

## Next step - kernel symbolication

- Find and list all kexts
- Find sysent and resolve syscalls
- Find and resolve mach traps
- Resolve IOKit objects vtable

## Next step - setting up primitives

- Scan kernel dump for gadgets
- Set up exec primitive
- Set up kernel read & write primitives

## Next step - kernel structs layout

- Look for proc_* functions
- Restore proc structure layout
- Dump memory, check for known values

# Getting root and sandbox bypass

- Patch setreuid (no KPP 😀)

- Patch ucred in proc structure in kernel
- Patch sandbox label value in ucred

## Getting kernel task

- Patch task_for_pid()

- Or save kernel self in our task bootstrap port
- Read it back via task_get_special_port()
- Restore original bootstrap port value

# Disable codesign checks

- Patch _debug to 1
- Patch _nl_symbol_ptr(got) entries

- Patch amfi variables
  - cs_enforcement_disable
  - allow_invalid_signatures

# Remount rootfs

- Patch __mac_mount
- Patch v_flags for rootfs vnode and mount

- Patch lwvm is_write_protected check
- Patch PE_i_can_has_debugger in lwvm

## Spawning ssh client

- Compile dropbear for ARMv7k
- Compile basic tools package
- Problem: WatchOS has more sandbox restrictions than iOS
- Kill watch specific sandbox operations (bind, connect,…)

## Spyware on a watch

- Watch have access to SMS, Calls, Health
- Photos and emails synced to Watch
- Fetch GPS location from a phone
- Microphone usage
- Apple pay 😀

# Messages, Contacts, Emails …

- Just dump from DB or de-serialize data
  - *private/var/mobile/Library/AddressBook/*
  - *private/var/mobile/Library/NanoMailKit/*
  - *private/var/mobile/Library/SMS/*
- Hook on fly on device sync\notification

## Health, Caches, App Data

- Just dump from DB or de-serialize data
  - *private/var/mobile/Library/Health/*
  - *private/var/mobile/Library/Caches/*
  - *private/var/mobile/Library/Application Data/*
- Hook on fly on device sync\notification

# Call recorder, Caches, App Data

- AudioToolbox.framework exists
- Add Observer on CTTelephonyCenter
- Catch kCTCallStatusChange in a callback
- Hook AudioUnitProess function
- Create file via ExtAudioFileCreateWithURL
- Use ExtAudioFileWrite to dump call data

# References

- Stefan Esser - iOS 10 - Kernel Heap Revisited
- Luca Todesco - com.apple.companion_proxy client
- Lookout - Technical Analysis of the Pegasus Exploits on iOS
- Siguza - tfp0 powered by Pegasus

@ mbazaliy