Jailbreaking Apple Watch

Max Bazaliy

December 4-7, 2017

black hat

whoami

- Security researcher at Lookout
- iOS/tvOS/WatchOS jailbreak author
- Lead researcher on Pegasus exploit chain
- Focused on software and hardware exploitation

What is Apple Watch ?

- Released in 2015
- Apple S1/S2/S3 processor
- ARMv7k 32 bit architecture
- 512/768 MB RAM
- One/Dual-core processor
- WatchOS

How does it work ?

- Fetch data from a phone
- Data transfer over Bluetooth
- Sync over Bluetooth and WiFi

Why to jailbreak a watch ?

- Access to file system (messages, emails..)
- Run debug tools on a watch (radare, frida)
- iPhone attack vector ☺

Apple Watch security

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

Possible attack vectors

- Memory corruption over Webkit

Possible attack vectors

- ~~Memory corruption over Webkit~~
- Boot chain attack over usb (diags port ☺)

Possible attack vectors

- ~~Memory corruption over Webkit~~
- Boot chain attack over usb (diags port ☺)
- Application extension based

Jailbreak step by step

- Get initial code execution
- Leak kernel base
- Dump whole kernel (for encrypted kernels)
- Find gadgets and setup primitives
- Disable security restrictions
- Run ssh client on a watch

Bugs of interest

- WatchOS 2.x
  - CVE-2016-4656 & CVE-2016-4680
- WatchOS 3.1.2
  - CVE-2017-2370
- WatchOS 4.0.1
  - CVE-2017-13861 ?☺

Leaking kernel base – WatchOS 2.x

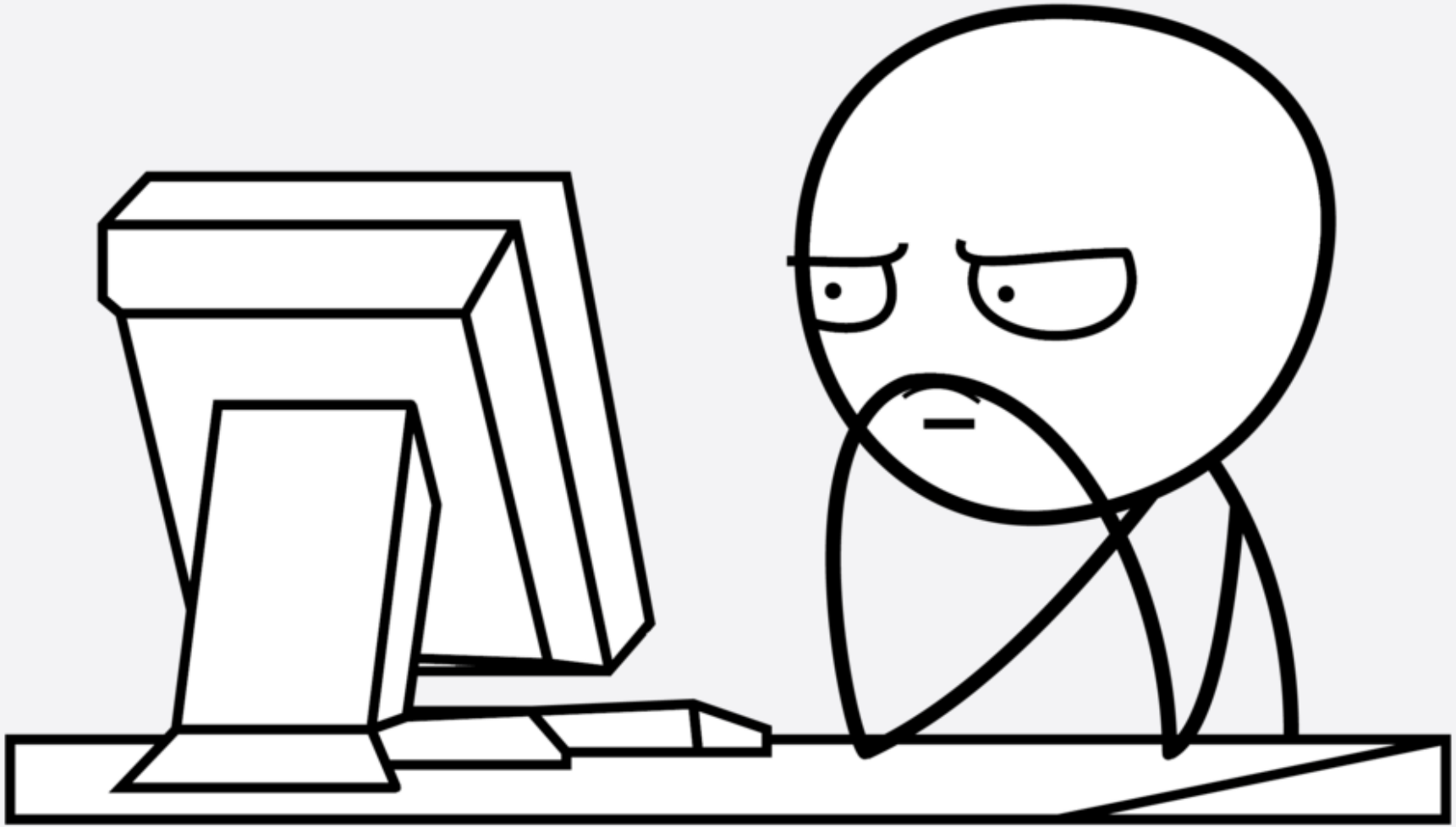
- CVE-2016-4680
- Object constructor missing bounds checking
- OSNumber object with high number of bits
- Object length used to copy value from stack
- Kernel stack memory leaked
- Can be triggered from an app's sandbox

CVE-2016-4656 exploitation

- Kernel mode UAF in OSUnserializeBinary
- OSString object deallocated
- retain() called on deallocated object
- Fake object with fake vtable –> code exec
- Can be triggered from an app's sandbox

Dumping WatchOS 2.x kernel

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



Getting OSString vtable

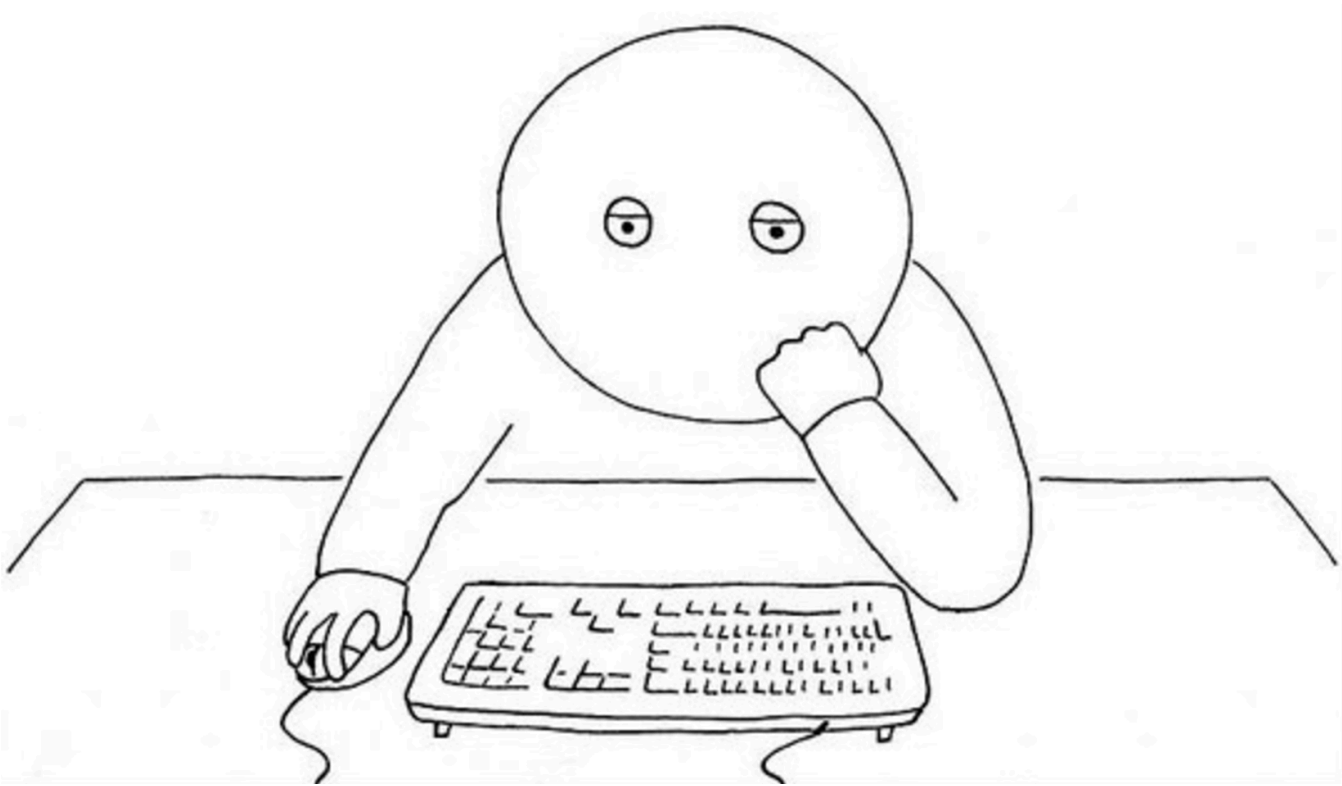
- OSString **vtable reference in** OSUnserializeBinary!
- OSUnserializeBinary **reference in** OSUnserializeXML

```
OSUnserializeBinary                     ; CODE XREF: OSUnserializeXML(char cons
                PUSH            {R4-R7,LR}
                ADD             R7, SP, #0xC
                PUSH.W          {R8,R10}
                MOV             R5, R0
                MOVS            R0, #0x14 ; this
                MOV             R8, R1
                BL              __ZN8OSObjectnwEm ; OSObject::operator new(ulo
                MOVW            R1, #:lower16:(__ZTV8OSString - 0x8031A9C0) ;
                MOV             R4, R0
                MOVT.W          R1, #:upper16:(__ZTV8OSString - 0x8031A9C0) ;
                MOV             R0, #(__ZN8OSString10gMetaClassE - 0x8031A9C2)
                ADD             R1, PC  ; `vtable for'OSString
```

Dumping kernel by panic logs

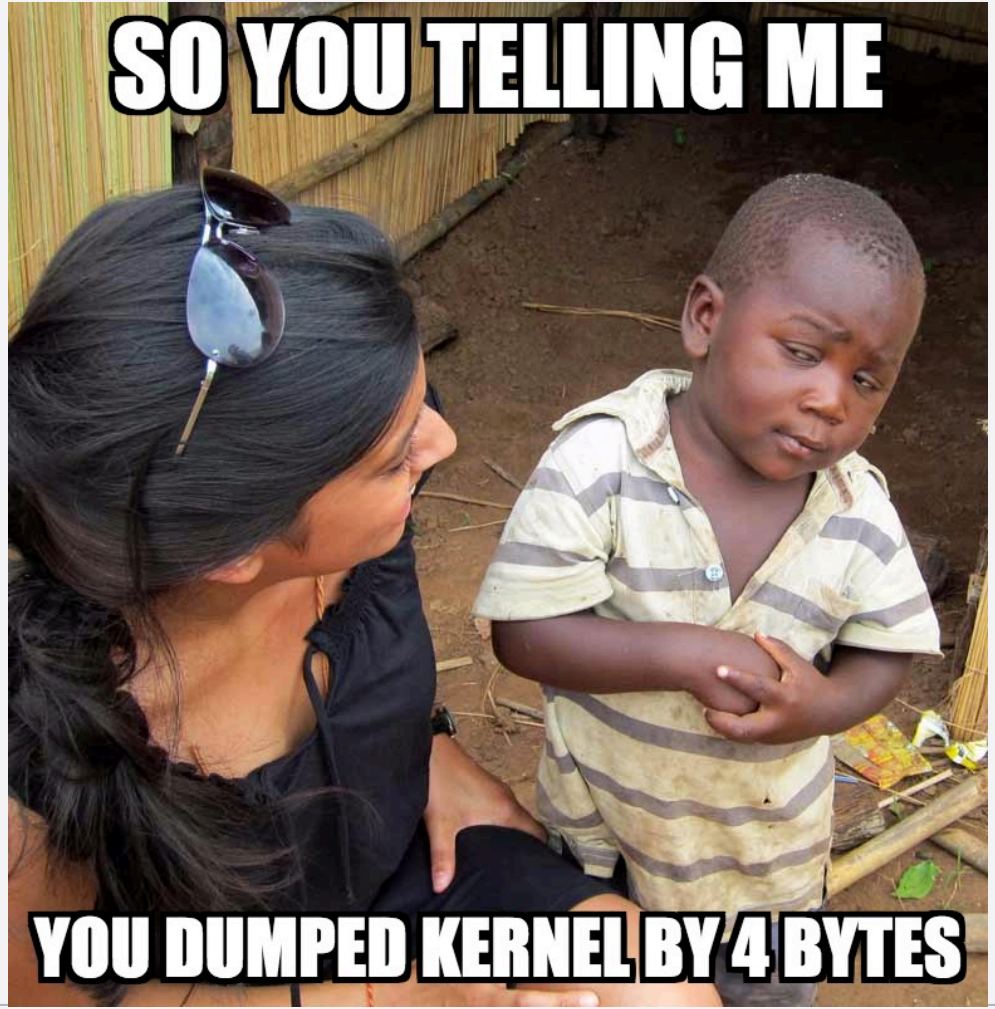
- We can control pointer to vtable
- Use address to leak as vtable address
- vtable will be dereferenced by retain() call
- Kernel will crash, but save panic log
- Address content appear in register state

It's fun !



Dumping kernel by 4 bytes

- Use address to leak as fake vtable address
- Watch will crash, wait until it restore
- ssh to a iPhone and run synchronization service
- Copy panic from Watch to iPhone and to Mac
- Parse panic, read 4 bytes and disassemble !
- Update address with 4 bytes delta and upload app
- Repeat


SO YOU TELLING ME
YOU DUMPED KERNEL BY 4 BYTES

Next step – full kernel dump

- Now use fake OSString obj to read kernel
- Read data via IORegistryEntryGetProperty

- 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

Jailbreaking Watch OS 3.x

- Kernels are not encrypted now
- No need to dump and symbolicate anymore
- New heap layout, some AMFI fixes
- More sandbox restrictions
- Vurnerable to CVE-2017-2370

CVE-2017-2370

- Kernel heap overflow
- mach_voucher_extract_attr_recipe
- Usermode pointer is used as copyin size arg
- We can corrupt mach message to get kernel RW
- Allocate userclient and read obj vtable -> KASLR
- Can be triggered from an app's sandbox

Next step – patchfinder

- String \ byte pattern + xref + analysis
- Simple arm emulator is helpful
- Resolve syscalls table, mach traps table

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 sself in 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
- Change mount flags in rootfs vnode

- Patch lwvm is_write_protected check
- Patch PE_i_can_has_debugger in lwvm

Spawning ssh client

- Compile dropbear ssh client for ARMv7k
- Compile basic tools package for ARMv7k
- More restricted sandbox than iOS
- Null out WatchOS specific sandbox ops

ssh connection problem…

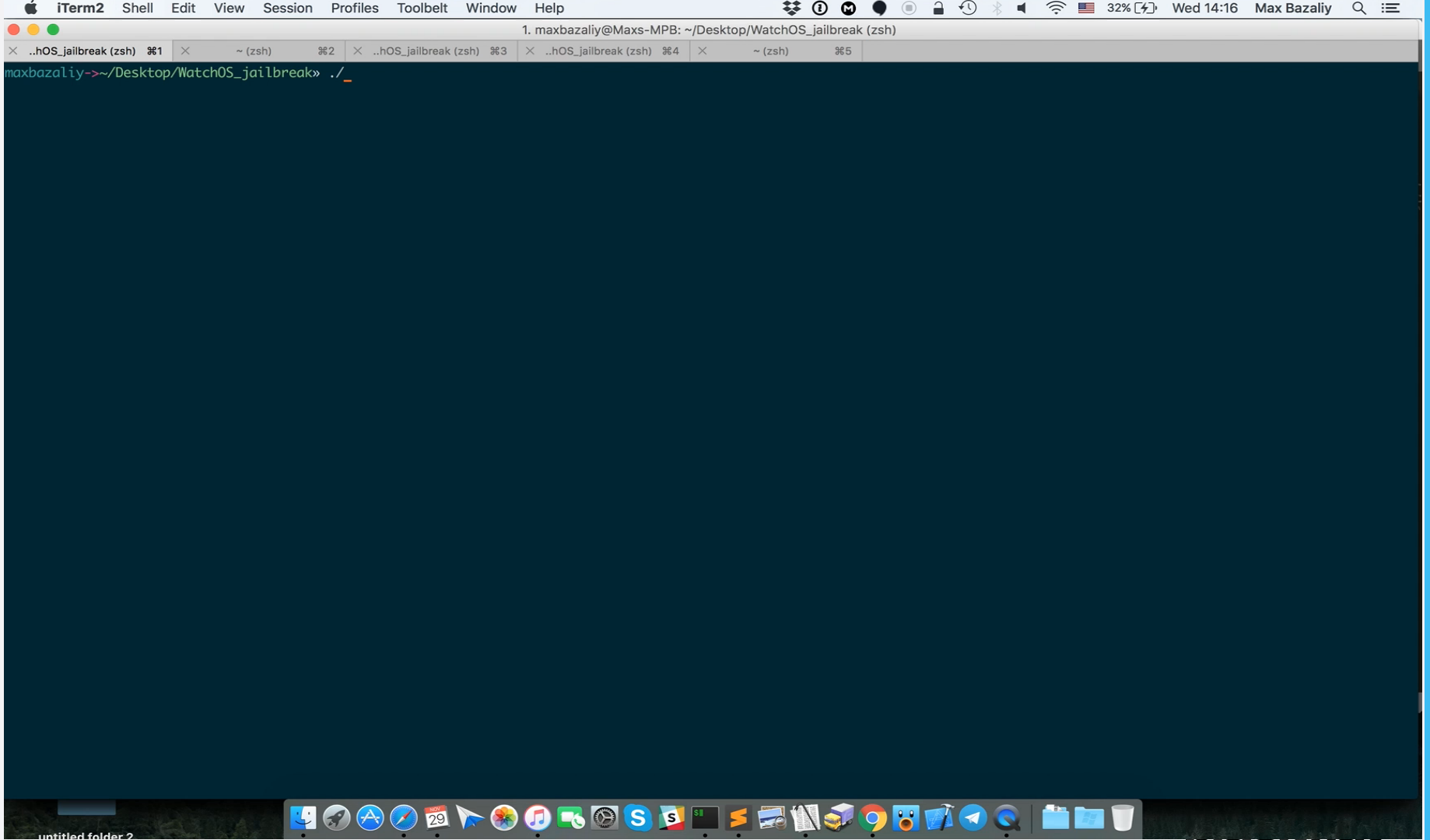
- WatchOS interfaces

```
"awdl0/ipv6" = "fe80::c837:8aff:fe60:90c2";
"lo0/ipv4”     = "127.0.0.1";
"lo0/ipv6"     = "fe80::1";
"utun0/ipv6" = "fe80::face:5e30:271e:3cd3";
```


ONE DOES NOT SIMPLY
CONNECT WATCH OVER SSH
imgflip.com

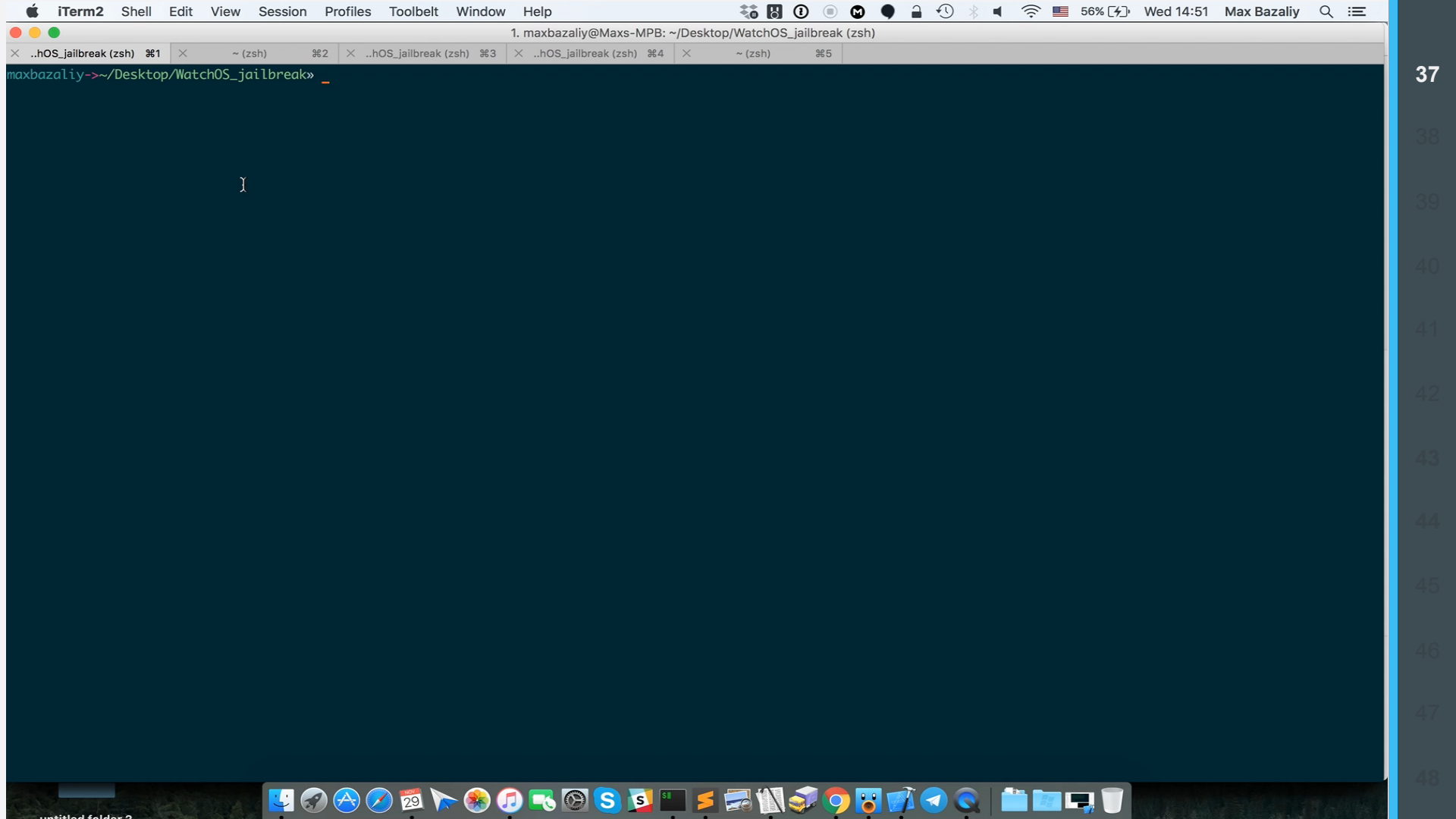
Watch <-> iPhone port forwarding

```
NSDictionary *comm = @{
        @"Command" :@"StartForwardingServicePort",
        @"ForwardedServiceName" :@"com.apple.syslog_relay",
        @"GizmoRemotePortNumber" :[NSNumber numberWithUnsignedShort: pt],
        @"IsServiceLowPriority" :@0,};

AMDServiceConnectionSendMessage(serviceConnection,
                                (__bridge CFPropertyListRef)(comm),
                                        kCFPropertyListXMLFormat_v1_0);

AMDServiceConnectionReceiveMessage(serviceConnection, &response,
                                        (CFPropertyListFormat*)&format);

NSNumber *iphone_port = response[@"CompanionProxyServicePort"];
```


iTerm2 Shell Edit View Session Profiles Toolbelt Window Help
32% Wed 14:16 Max Bazaliy
1. maxbazaliy@Maxs-MPB: ~/Desktop/WatchOS_jailbreak (zsh)
..hOS_jailbreak (zsh) ⌘1
~ (zsh) ⌘2
..hOS_jailbreak (zsh) ⌘3
..hOS_jailbreak (zsh) ⌘4
~ (zsh) ⌘5
maxbazaliy->~/Desktop/WatchOS_jailbreak» ./
35

SSH over WiFi

- Watch can be connected to 2.4Hz WiFi
- Can be a little bit tricky but it works
- iPhone is not involved at all ☺
- Just leak address and connect


iTerm2 Shell Edit View Session Profiles Toolbelt Window Help
56% Wed 14:51 Max Bazaliy
1. maxbazaliy@Maxs-MPB: ~/Desktop/WatchOS_jailbreak (zsh)
..hOS_jailbreak (zsh) ⌘1
~ (zsh) ⌘2
..hOS_jailbreak (zsh) ⌘3
..hOS_jailbreak (zsh) ⌘4
~ (zsh) ⌘5
maxbazaliy->~/Desktop/WatchOS_jailbreak»
37
38
39
40
41
42
43
44
45
46
47
48

Apple Watch usage

- Watch has access to SMS, Calls, Health
- Photos and emails synced to Watch
- Access to GPS location
- Microphone usage
- Apple Pay

Post jailbreak

- Full access to jailbroken watch file system
  - Messages
  - Call history
  - Contacts
  - Emails
  - GPS loacation

What's next ?

- Interpose or trampoline system functions
- Catch data on sync with a iPhone
- Call recordings
- Create tweaks for a watch
- Run frida and radare

Black Hat sound bytes

- WatchOS security is mostly equal to iOS
- Easier data forensics on a Watch
- Exploits became more valuable

@mbazaliy