Trust in Apple’s secret garden: Exploring & Reversing Apple’s Continuity Protocol

Outline

- Motivation
- Introduction to Continuity Protocol
- iCloud, APNS, iMessage
- Continuity
    - Previous Research
    - Software Stack
    - Protocol
- Remarks

About me

- Working as DevOps + Fullstack
- Independent Security Research
- Tinkering / Hacking devices or new things
- “Security in communication process”

Motivation

- Study on how Apple actually implements security
- Shed light into Apple’s secret garden
- Make iOS device more usable on non-macOS device

Motivation

- Responsible disclosure? Bug bounty?
  - No apparent vulnerability found yet
  - No bug bounty for such domain

Continuity

Continuity

- “Move seamlessly between your devices with Handoff, Universal Clipboard, iPhone Cellular Calls, SMS/MMS messaging, Instant Hotspot, Continuity Camera, AirDrop, Apple Pay, and Auto Unlock.”
- Heavily relies on BLE and iMessage / iCloud
- Most things won’t work without Bluetooth

Why Continuity

- Instant Hotspot (macOS + iOS)
  - Open Wi-Fi menu
  - Wait for device to appear
  - Click on device’s name
- Not-so-Instant Hotspot (!macOS + iOS)
  - Grab your phone & enable hotspot
  - Scan for Wi-Fi stations
  - It usually works, but sometimes it won’t
    - New association only when Hotspot page is active

Contuniuty

- “...Continuity takes advantage of technologies like iCloud...”
- “...encryption of the individual messages, which is similar to how iMessage is encrypted...”

iCloud, APNs, IDS, iMessage

iCloud

- Debuted around end of 2011
- Est. 850M users
- Multiple Services
  - Backup, Device Locater, Messaging
- Push Service (APNS)

APNs

- Apple Push Notification Service
- Device ID
  - APNs address (deviceToken), per device
- Public-Key Cryptography + TLS

iMessage

- Proprietary Messaging Service
- Supports text & attachments

- End-to-End encryption
- Continuity message are encrypted similar to iMessage

identityservicesd (IDS)

- Directory Service
  - iMessage keys
- Links with iCloud
  - Able to grab any other device’s public key from iCloud, with corresponding phone # or email

iMessage “onboarding”

- Keys are generated
  - RSA + ECDSA
- Public key will be send to iCloud
  - Associated with (phone # / email) + APNs address
  - Private key never leaves device
    - Easily accessible with Keychain

Sending with iMessage

- Generates message bplist
- Concats
  - Target public key + aes(bplist) + session key
- Encrypts AES key with RSA public key
- Appends ECDSA-SHA1 to the end

iMessage Attachment Mode

- >4KB or >16KB payload, or attachment
  - Depends on iOS version

- Content encrypted with AES-CTR (256b)
- Sends URI and content’s SHA-1 instead

Continuity

Cellular Call Relay

- Incoming Call
  - Bootstrapping
    - iPhone (TCP) → APNs → Local Mac/iPad
  - Call
    - iPhone (UDP) → Local Mac/iPad
  - Ring is terminated with BLE when answered

Cellular Call Relay

- iPad / Mac must be on the same Wi-Fi network as phone
- Receive/Make cellular calls using iPad / Mac
- Relies on APNs to work

- “Upon answering the call, the audio is seamlessly transmitted from the user’s iPhone using a secure peer-to-peer connection between the two devices.”

Cellular Call Relay

- Call is terminated via APNs
  - Local Mac/iPad → APNs → iPhone
  - iPhone terminated call

- Martin Vigo: DIY Spy Program: Abusing Apple’s Call Relay Protocol
  - DoS, Spying, impersonation

AirDrop

- Based on BLE & AWDL (Apple Wireless Direct Link)
  - “Apple-created peer-to-peer Wi-Fi technology”
- Bootstrapping using BLE
  - Detect devices nearby (broadcast)
  - Set up transfer
- Milan Stute et, al. 2018. One Billion Apples’ Secret Sauce: Recipe for the Apple Wireless Direct Link Ad hoc Protocol

Continuity Stack

- Module hooks with sharingd
  - Calls method upon message received
- Host → bluetoothd → sharingd → Target module

Flow

- Messages are encrypted-then-signed
- Message received via HCI
- Passed on to sharingd
- sharingd → IDS → MessageProtection → sharingd → target service

Flow (Instant Hotspot)

- Device decides to connect to hotspot
- Connects to device and uses GATT to exchange connection info
  - SSID / PSK included
- Device sends Probe Request
  - Hotspot sends Probe Response & Beacon
- Device establishes Wi-Fi connection with hotspot

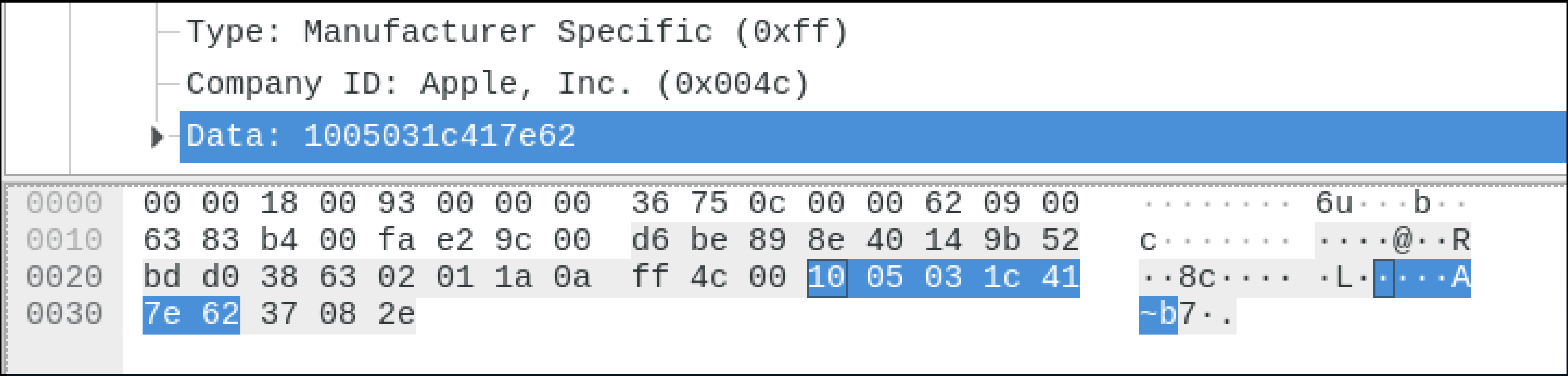
Continuity Broadcast Protocol

- Device sends broadcast continuously
  - MAC
  - Type ID
  - Payload
- Always on CH37

Broadcast Sender Validation

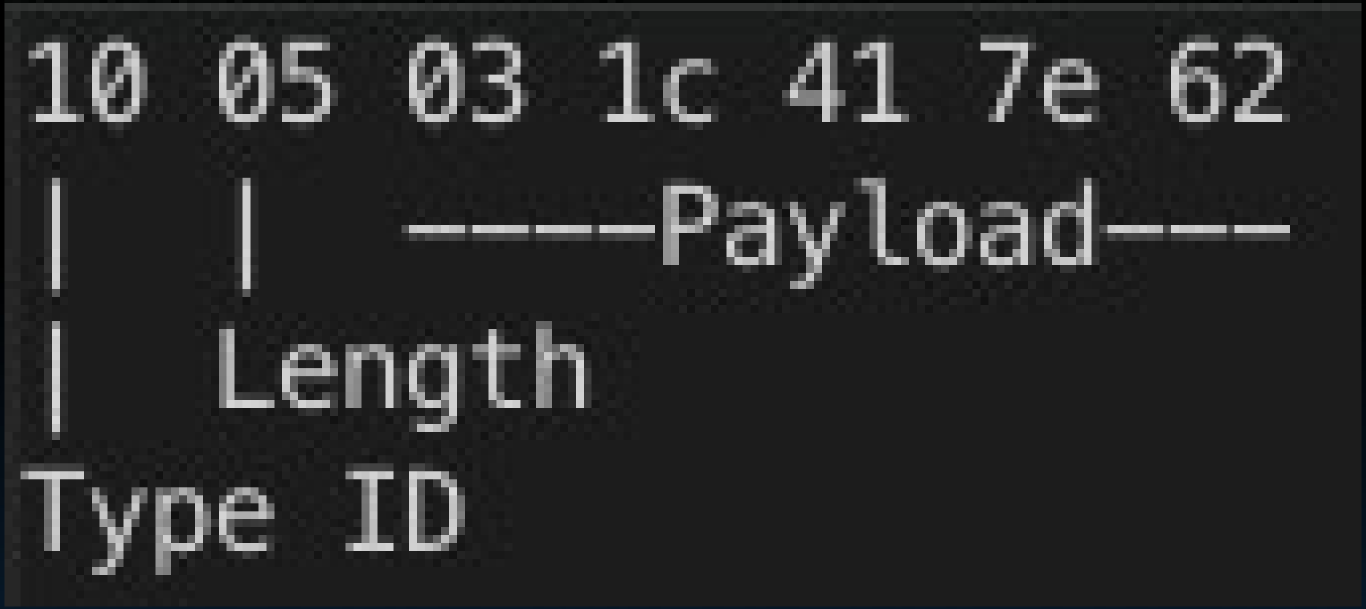
- None
- Broadcast using private address
  - Uses VSC to translates private address to public
  - Change on each power cycle

Continuity BLE Broadcast Protocol



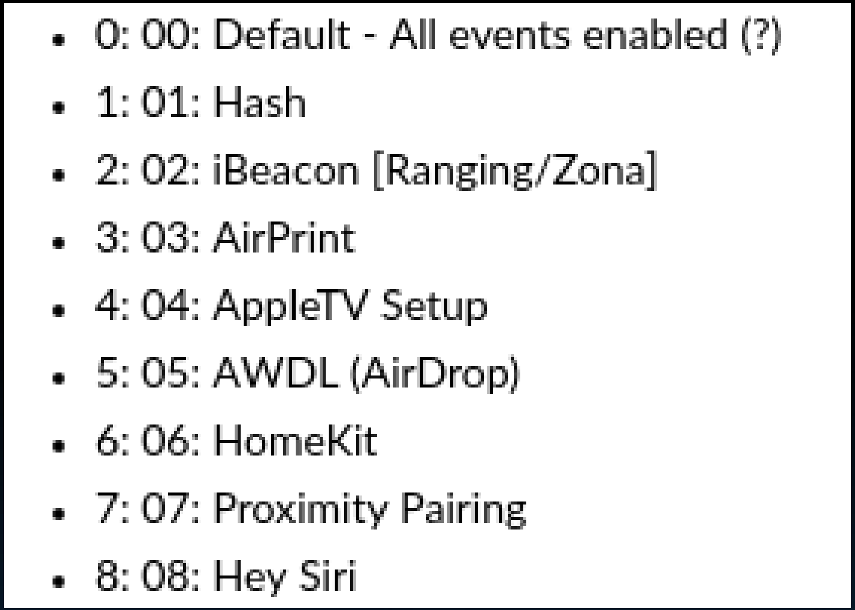
```
Type: Manufacturer Specific (0xff)
Company ID: Apple, Inc. (0x004c)
Data: 1005031c417e62
```

```
0000  00 00 18 00 93 00 00 00  36 75 0c 00 00 62 09 00   ········ 6u···b··
0010  63 83 b4 00 fa e2 9c 00  d6 be 89 8e 40 14 9b 52   c······· ····@··R
0020  bd d0 38 63 02 01 1a 0a  ff 4c 00 10 05 03 1c 41   ··8c···· ·L·····A
0030  7e 62 37 08 2e                                     ~b7·.
```

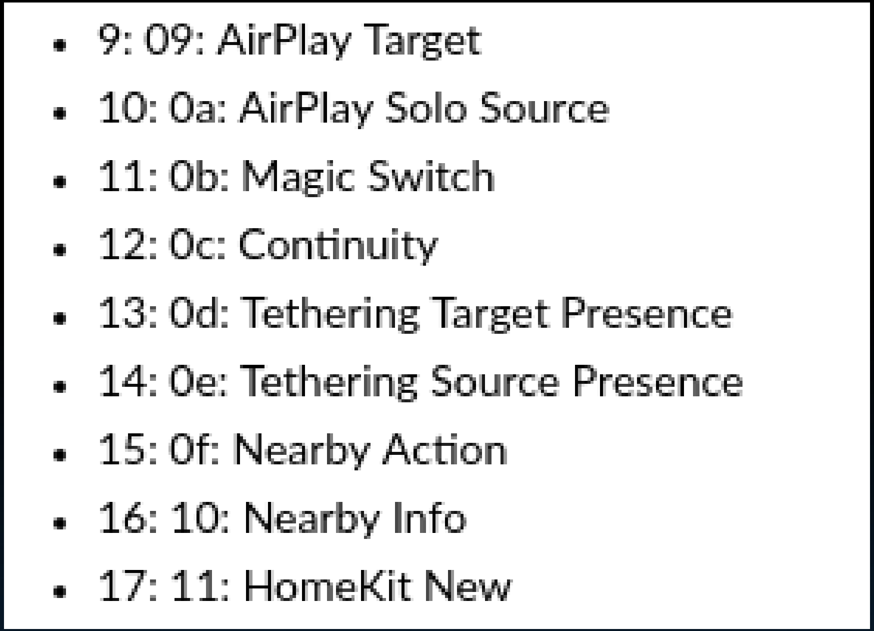


```
10 05 03 1c 41 7e 62
|  |  -----Payload----
|  Length
Type ID
```

Continuity BLE Broadcast Type ID



- 0: 00: Default - All events enabled (?)
- 1: 01: Hash
- 2: 02: iBeacon [Ranging/Zona]
- 3: 03: AirPrint
- 4: 04: AppleTV Setup
- 5: 05: AWDL (AirDrop)
- 6: 06: HomeKit
- 7: 07: Proximity Pairing
- 8: 08: Hey Siri

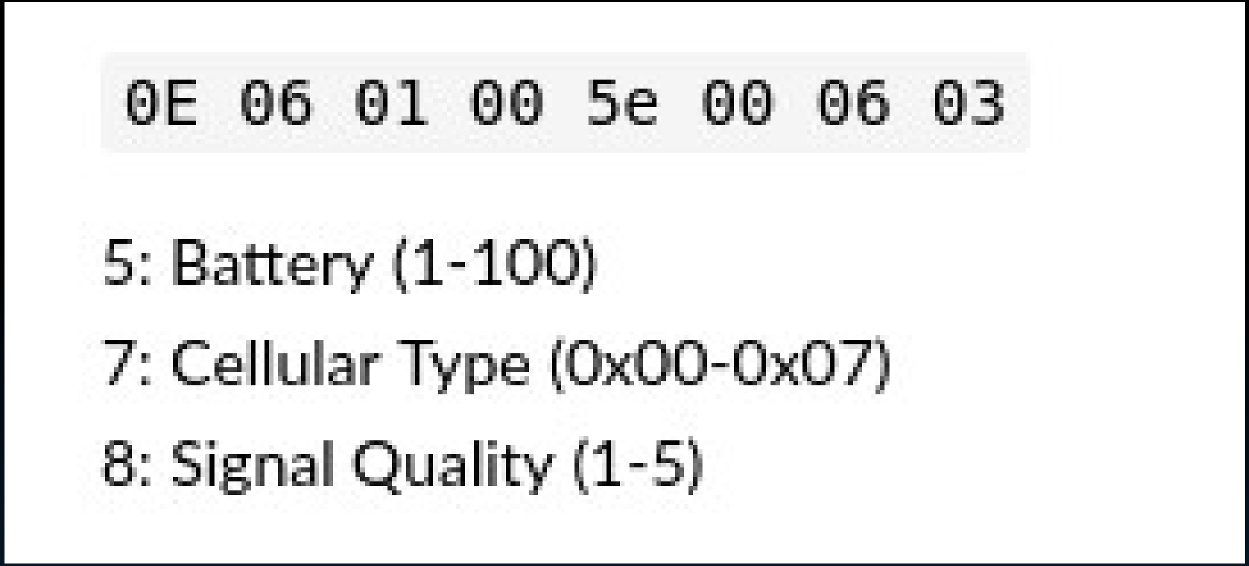


- 9: 09: AirPlay Target
- 10: 0a: AirPlay Solo Source
- 11: 0b: Magic Switch
- 12: 0c: Continuity
- 13: 0d: Tethering Target Presence
- 14: 0e: Tethering Source Presence
- 15: 0f: Nearby Action
- 16: 10: Nearby Info
- 17: 11: HomeKit New

Additional Tools for Xcode <=10.2 PacketDecoder

Tethering Source Presence

- Type 0x0e
- Battery 5e (94%)
- Cell 0x06 (LTE)
- Quality 3/5



```
0E 06 01 00 5e 00 06 03
```

5: Battery (1-100)

7: Cellular Type (0x00-0x07)

8: Signal Quality (1-5)

Continuity: Attack Vectors

Attack Vectors

- Privacy Leak
- Spoofing

Privacy Leak

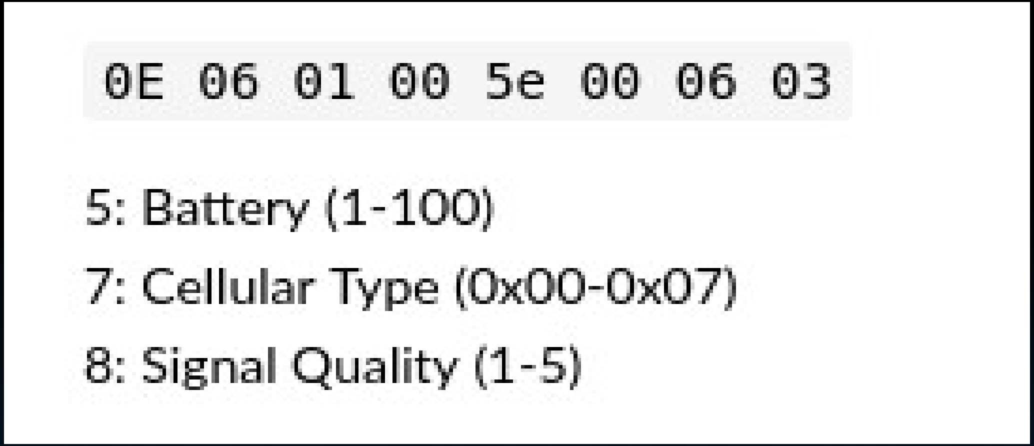
- Device Tracking
  - Device Fingerprinting
  - Attributes
  - Activity
  - Identity
    - Deanonymize random MAC

Device Fingerprinting

- Type ID
  - Device type
    - e.g. No instant hotspot for iPad Wi-Fi
  - OS Version
    - Apple watch: iOS >= 11
    - Nearby: iOS >= 10

Attributes

- Instant Hotspot
  - Battery Life
  - Cell Service Type
  - Cell Quality



```
0E 06 01 00 5e 00 06 03
```

5: Battery (1-100)

7: Cellular Type (0x00-0x07)

8: Signal Quality (1-5)

Activity

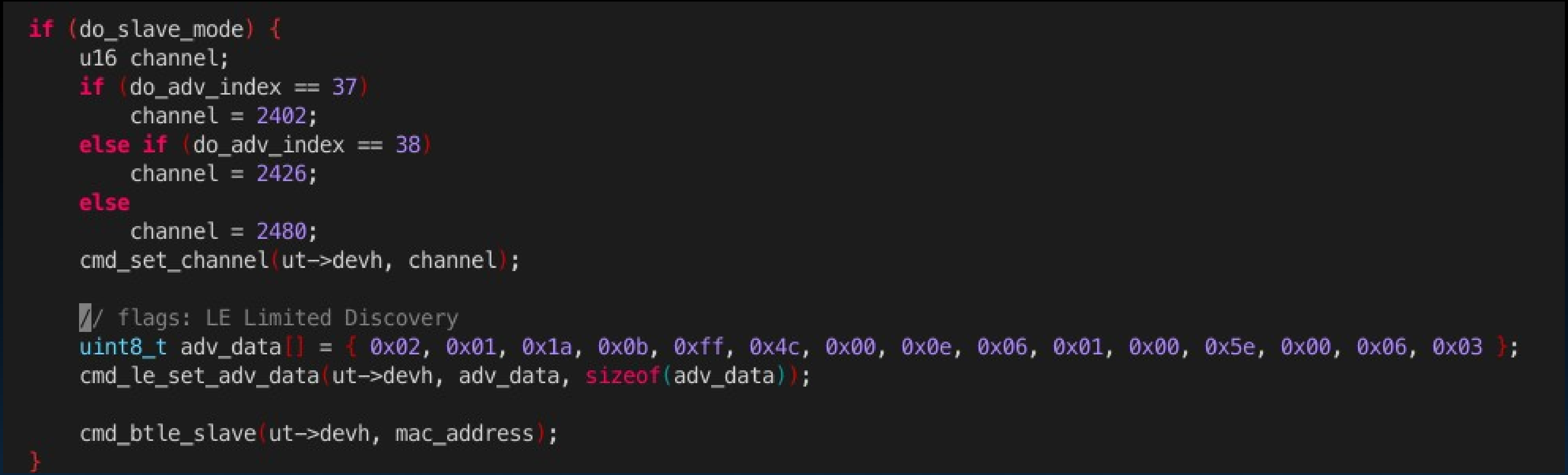
- Handoff
  - Broadcasts when applicable, e.g. Firefox open & Foreground
- Instant Hotspot
  - Handshake only occurs when device in proximity
- Nearby
  - Always broadcasting

Identity

- Instant Hotspot
  - Wi-Fi connection is made with public MAC
- Contextual Analysis
  - Wi-Fi connection after Continuity message
    - Wi-Fi MAC + 1 = Bluetooth MAC

Spoofing

- ubertooth-btle faux slave mode
- Needs sender/receiver public MAC



```
if (do_slave_mode) {
    u16 channel;
    if (do_adv_index == 37)
        channel = 2402;
    else if (do_adv_index == 38)
        channel = 2426;
    else
        channel = 2480;
    cmd_set_channel(ut->devh, channel);

    // flags: LE Limited Discovery
    uint8_t adv_data[] = { 0x02, 0x01, 0x1a, 0x0b, 0xff, 0x4c, 0x00, 0x0e, 0x06, 0x01, 0x00, 0x5e, 0x00, 0x06, 0x03 };
    cmd_le_set_adv_data(ut->devh, adv_data, sizeof(adv_data));

    cmd_btle_slave(ut->devh, mac_address);
}
```

Continuity Protocol

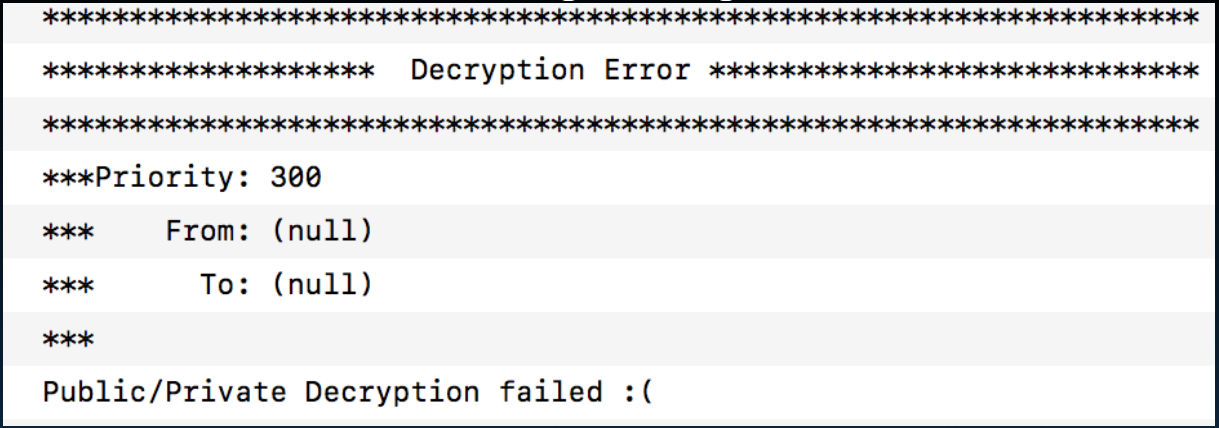
lldb

Verify / Decryption

- Every connection is associated with a UUID
- If UUID is added before, don’t fetch public key again

Accidently broke IDS

- I deleted “iMessage ____ Key” in keychain
- Fixed by rebooting phone & mac
- Hypothesis: Keys are downloaded & uploaded / regenerated on iMessage login

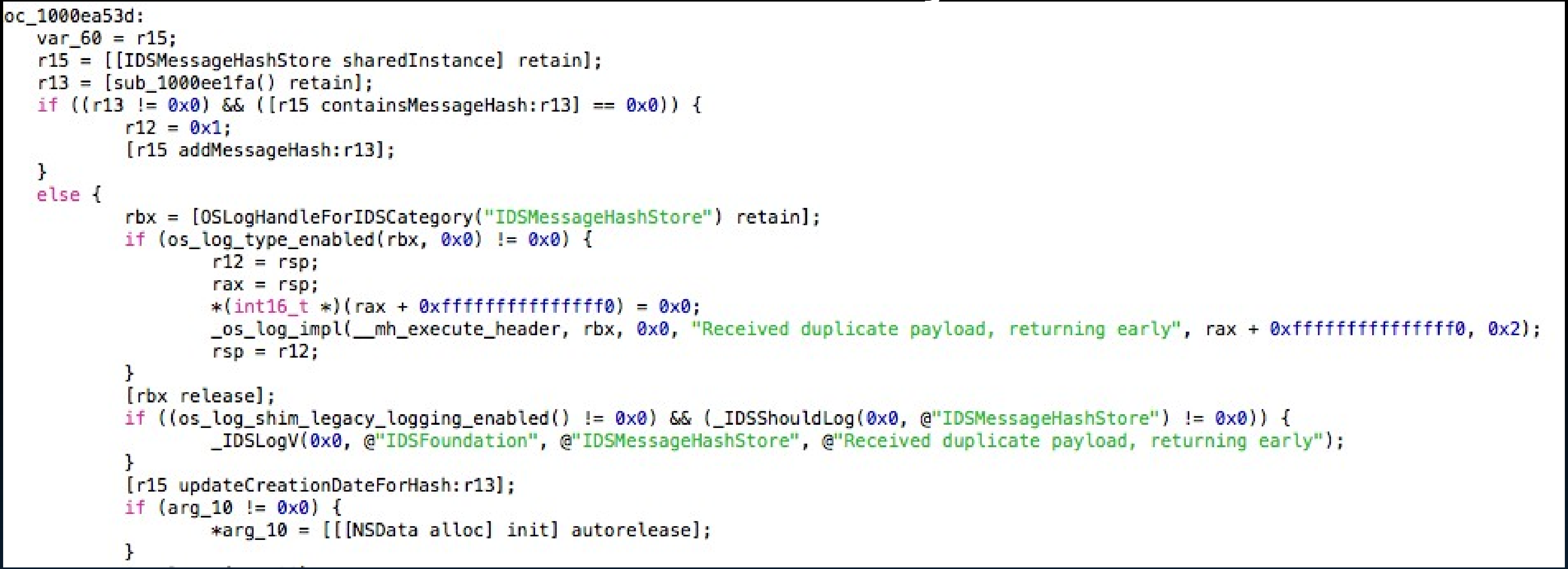


```
******************************************************************
*******************  Decryption Error ****************************
******************************************************************
***Priority: 300
***    From: (null)
***      To: (null)
***
Public/Private Decryption failed :(
```

Verify / Decryption

- If public key is not found, returns false
- Checks SHA1 of message with existing caches
- Calls verification & decryption
    - MessageProtection
    - SecMPVerifyAndExposeMessage

~~Malleability~~



```
oc_1000ea53d:
    var_60 = r15;
    r15 = [[IDSMessageHashStore sharedInstance] retain];
    r13 = [sub_1000ee1fa() retain];
    if ((r13 != 0x0) && ([r15 containsMessageHash:r13] == 0x0)) {
            r12 = 0x1;
            [r15 addMessageHash:r13];
    }
    else {
            rbx = [OSLogHandleForIDSCategory("IDSMessageHashStore") retain];
            if (os_log_type_enabled(rbx, 0x0) != 0x0) {
                    r12 = rsp;
                    rax = rsp;
                    *(int16_t *)(rax + 0xfffffffffffffff0) = 0x0;
                    _os_log_impl(__mh_execute_header, rbx, 0x0, "Received duplicate payload, returning early", rax + 0xfffffffffffffff0, 0x2);
                    rsp = r12;
            }
            [rbx release];
            if ((os_log_shim_legacy_logging_enabled() != 0x0) && (_IDSShouldLog(0x0, @"IDSMessageHashStore") != 0x0)) {
                    _IDSLogV(0x0, @"IDSFoundation", @"IDSMessageHashStore", @"Received duplicate payload, returning early");
            }
            [r15 updateCreationDateForHash:r13];
            if (arg_10 != 0x0) {
                    *arg_10 = [[[NSData alloc] init] autorelease];
            }
```

~~Malleability~~

- Messages are only signed
  - No MAC
- Replay was allowed
- Certificate Pinning wasn’t implemented
- Message is Compressed-then-Encrypted
  - Compression Oracle

SecMPVerifyAndExposeMessage

- SecMPVerifyMessageContents(payload)
  - sizeof(payload) > 0x11
- Here lies raw payload from HCI
  - Calls SecKeyDigestAndVerifyWithError to verify against it
    - Signing is made against SHA-1 digest of payload
  - If passed, actual decryption is called

Message Verification & Decryption

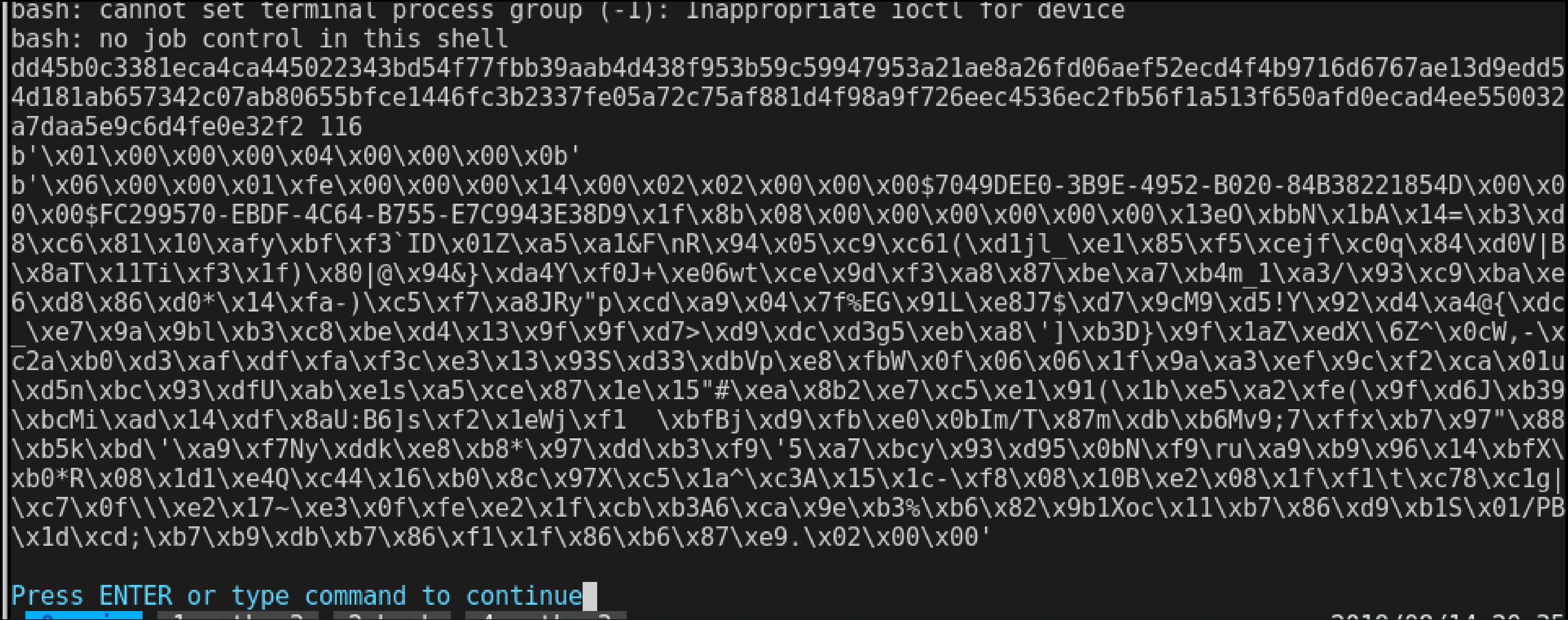
- Relies on Security.framework
- Security Transforms
  - SecVerifyTransformCreate
  - SecDecryptTransformCreate

Data Structure

- HCI payload
  - Data can be split into multiple packets
  - 0x27-0x28 mentions payload length
  - 0x28-end is the payload + signature
  - Some kind of “header” before length (0x03-0x27)
    - If not exists, packet is continuation of previous one
  - Total length – payload length = Signature length

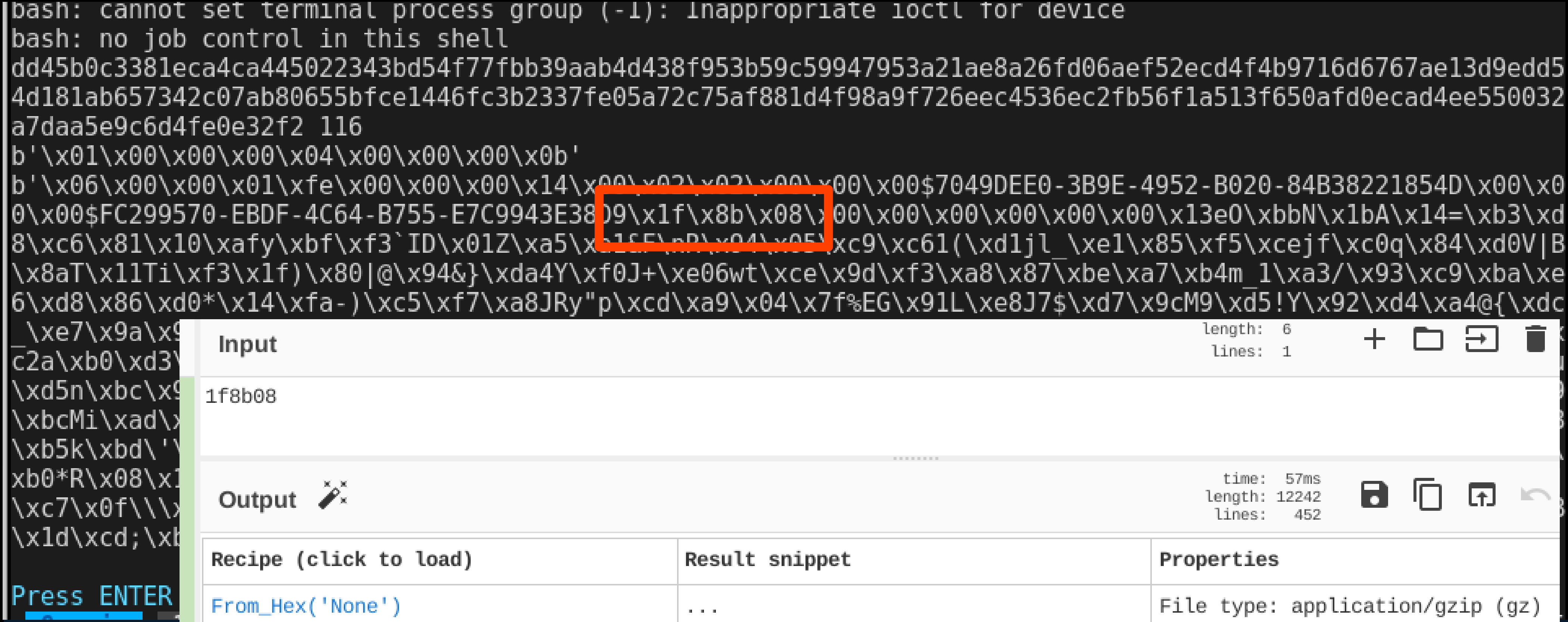
Actual Decryption

- First Stage: RSA-OAEP of first 160 bytes
- Second Stage
  - rsa_decrypt(data)[:16] → AES-128 CTR Key, PK = 1
  - aes_decrypt(rsa_decrypt(data)[16:] + raw_data[160:])
- Third Stage
  - Gzipped bplist



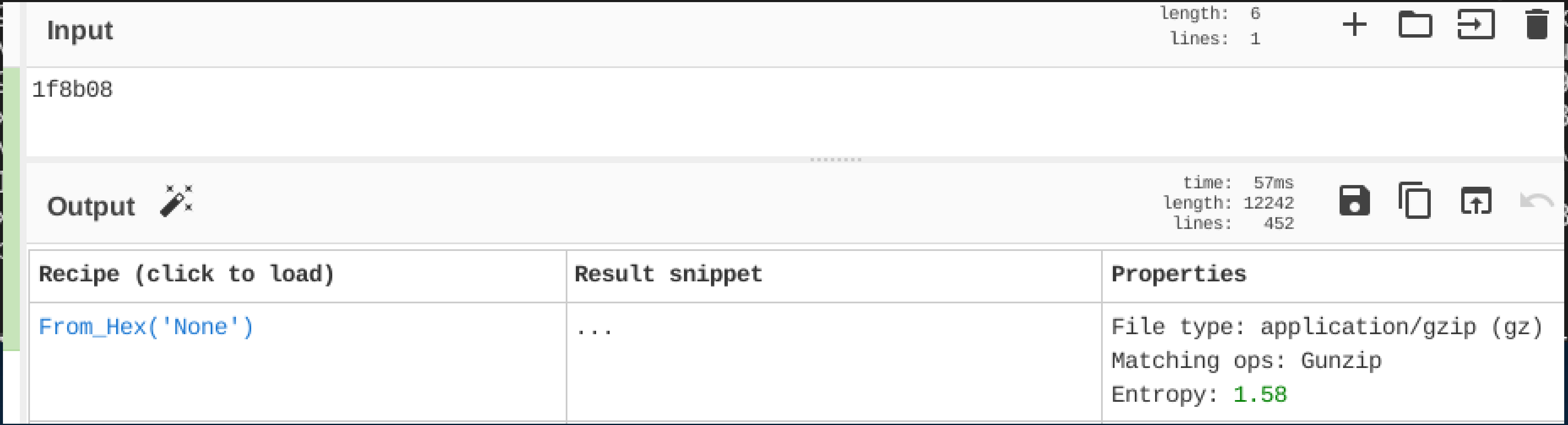
```
bash: cannot set terminal process group (-1): Inappropriate ioctl for device
bash: no job control in this shell
dd45b0c3381eca4ca445022343bd54f77fbb39aab4d438f953b59c59947953a21ae8a26fd06aef52ecd4f4b9716d6767ae13d9edd5
4d181ab657342c07ab80655bfce1446fc3b2337fe05a72c75af881d4f98a9f726eec4536ec2fb56f1a513f650afd0ecad4ee550032
a7daa5e9c6d4fe0e32f2 116
b'\x01\x00\x00\x00\x04\x00\x00\x00\x0b'
b'\x06\x00\x00\x01\xfe\x00\x00\x00\x14\x00\x02\x02\x00\x00\x00$7049DEE0-3B9E-4952-B020-84B38221854D\x00\x0
0\x00$FC299570-EBDF-4C64-B755-E7C9943E38D9\x1f\x8b\x08\x00\x00\x00\x00\x00\x00\x13eO\xbbN\x1bA\x14=\xb3\xd
8\xc6\x81\x10\xafy\xbf\xf3`ID\x01Z\xa5\xa1&F\nR\x94\x05\xc9\xc61(\xd1jl_\xe1\x85\xf5\xcejf\xc0q\x84\xd0V|B
\x8aT\x11Ti\xf3\x1f)\x80|@\x94&}\xda4Y\xf0J+\xe06wt\xce\x9d\xf3\xa8\x87\xbe\xa7\xb4m_l\xa3/\x93\xc9\xba\xe
6\xd8\x86\xd0*\x14\xfa-)\xc5\xf7\xa8JRy"p\xcd\xa9\x04\x7f%EG\x91L\xe8J7$\xd7\x9cM9\xd5!Y\x92\xd4\xa4@{\xdc
_\xe7\x9a\x9bl\xb3\xc8\xbe\xd4\x13\x9f\x9f\xd7>\xd9\xdc\xd3g5\xeb\xa8\']\xb3D}\x9f\x1aZ\xedX\\6Z^\x0cW,-\x
c2a\xb0\xd3\xaf\xdf\xfa\xf3c\xe3\x13\x93S\xd33\xdbVp\xe8\xfbW\x0f\x06\x06\x1f\x9a\xa3\xef\x9c\xf2\xca\x01u
\xd5n\xbc\x93\xdfU\xab\xe1s\xa5\xce\x87\x1e\x15"#\xea\x8b2\xe7\xc5\xe1\x91(\x1b\xe5\xa2\xfe(\x9f\xd6J\xb39
\xbcMi\xad\x14\xdf\x8aU:B6]s\xf2\x1eWj\xf1  \xbfBj\xd9\xfb\xe0\x0bIm/T\x87m\xdb\xb6Mv9;7\xffx\xb7\x97"\x88
\xb5k\xbd\'\xa9\xf7Ny\xddk\xe8\xb8*\x97\xdd\xb3\xf9\'5\xa7\xbcy\x93\xd95\x0bN\xf9\ru\xa9\xb9\x96\x14\xbfX\
xb0*R\x08\x1d1\xe4Q\xc44\x16\xb0\x8c\x97X\xc5\x1a^\xc3A\x15\x1c-\xf8\x08\x10B\xe2\x08\x1f\xf1\t\xc78\xc1g|
\xc7\x0f\\\xe2\x17~\xe3\x0f\xfe\xe2\x1f\xcb\xb3A6\xca\x9e\xb3%\xb6\x82\x9b1Xoc\x11\xb7\x86\xd9\xb1S\x01/PB
\x1d\xcd;\xb7\xb9\xdb\xb7\x86\xf1\x1f\x86\xb6\x87\xe9.\x02\x00\x00'

Press ENTER or type command to continue
```



```
bash: cannot set terminal process group (-1): Inappropriate ioctl for device
bash: no job control in this shell
dd45b0c3381eca4ca445022343bd54f77fbb39aab4d438f953b59c59947953a21ae8a26fd06aef52ecd4f4b9716d6767ae13d9edd5
4d181ab657342c07ab80655bfce1446fc3b2337fe05a72c75af881d4f98a9f726eec4536ec2fb56f1a513f650afd0ecad4ee550032
a7daa5e9c6d4fe0e32f2 116
b'\x01\x00\x00\x00\x04\x00\x00\x00\x0b'
b'\x06\x00\x00\x01\xfe\x00\x00\x00\x14\x00\x02\x02\x00\x00\x00$7049DEE0-3B9E-4952-B020-84B38221854D\x00\x0
0\x00$FC299570-EBDF-4C64-B755-E7C9943E38D9\x1f\x8b\x08\x00\x00\x00\x00\x00\x00\x13eO\xbbN\x1bA\x14=\xb3\xd
8\xc6\x81\x10\xafy\xbf\xf3`ID\x01Z\xa5\xe1{F}nP\x04\x05\xc9\xc61(\xd1jl_\xe1\x85\xf5\xcejf\xc0q\x84\xd0V|B
\x8aT\x11Ti\xf3\x1f)\x80|@\x94&}\xda4Y\xf0J+\xe06wt\xce\x9d\xf3\xa8\x87\xbe\xa7\xb4m_1\xa3/\x93\xc9\xba\xe
6\xd8\x86\xd0*\x14\xfa-)\xc5\xf7\xa8JRy"p\xcd\xa9\x04\x7f%EG\x91L\xe8J7$\xd7\x9cM9\xd5!Y\x92\xd4\xa4@{\xdc
_\xe7\x9a\x9
c2a\xb0\xd3\
\xd5n\xbc\x9
\xbcMi\xad\x
\xb5k\xbd\'\
xb0*R\x08\x1
\xc7\x0f\\\x
\x1d\xcd;\xb

Press ENTER
```



Input

```
1f8b08
```

length: 6
lines: 1

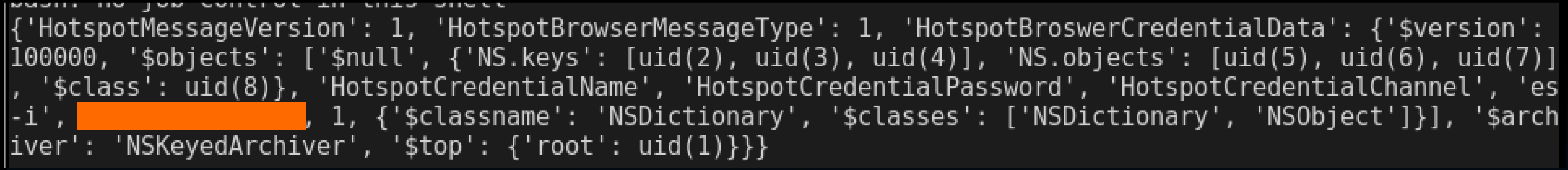
Output

time: 57ms
length: 12242
lines: 452

Recipe (click to load)	Result snippet	Properties
From_Hex('None')	...	File type: application/gzip (gz) Matching ops: Gunzip Entropy: 1.58

bplist00

- Apple Binary Plist



```
bash: no job control in this shell
{'HotspotMessageVersion': 1, 'HotspotBrowserMessageType': 1, 'HotspotBroswerCredentialData': {'$version':
100000, '$objects': ['$null', {'NS.keys': [uid(2), uid(3), uid(4)], 'NS.objects': [uid(5), uid(6), uid(7)]
, '$class': uid(8)}, 'HotspotCredentialName', 'HotspotCredentialPassword', 'HotspotCredentialChannel', 'es
-i', [redacted], 1, {'$classname': 'NSDictionary', '$classes': ['NSDictionary', 'NSObject']}], '$arch
iver': 'NSKeyedArchiver', '$top': {'root': uid(1)}}}
```

evanslify/continuity

- https://github.com/evanslify/continuity
- Other features would be added along the way

Remarks

- Blindly trusting a device is dangerous
  - Especially closed-sourced
- Moar encryption can be used
  - Encryption in broadcast, backed with IDS?
- No blatant exploit found