black hat®
USA 2021
August 4-5, 2021
ARSENAL

black hat USA 2021
August 4-5, 2021
ARSENAL

Play with Fire

Uncover Fairplay DRM and Obfuscation for Fun and Profit

Junzhi Lu(@pwn0rz), Xindi Wang, Ju Zhu of Meituan

DRM Decryption

How Decryption Tool works

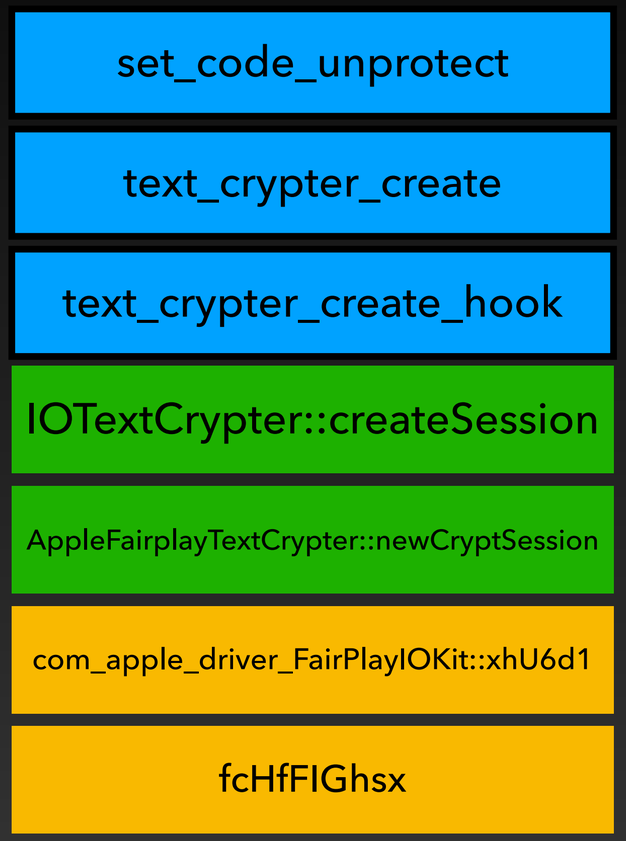
- Dynamic Way(Run & Dump)
  - Clutch
  - frida-ios-dump
  - bagbak
- Static Way(mremap_encrypted)
  - Flex decrypt

```
bash-3.2$ otool -l [illegible] | grep -i crypt
          cmd LC_ENCRYPTION_INFO_64
     cryptoff 16384
    cryptsize 4177920
      cryptid 1
```

Initial Decryption

Create Cryptor

- Kernel holds and calls the hook function
- IOTextEncryptionFamily acts as a bridge
- FairplayIOKit do the real stuff



Decrypt Page

- Kernel function text_crypter_create returns page_crypt_info
- IOTextEncryptionFamily acts as a bridge
- FairplayIOKit do the real stuff

text_crypter_create

```
/*
 * Interface for text decryption family
 */
struct pager_crypt_info {
    /* Decrypt one page */
    int     (*page_decrypt)(const void *src_vaddr, void *dst_vaddr,
            unsigned long long src_offset, void *crypt_ops);
    /* Pager using this crypter terminates - crypt module not needed anymore */
    void    (*crypt_end)(void *crypt_ops);
    /* Private data for the crypter */
    void    *crypt_ops;
    volatile int    crypt_refcnt;
};
```

IOTextCrypter::decryptPage

AppleFairplayTextCrypter::decryptPage

com_apple_driver_FairPlayIOKit::bvqhJ

EQlZPp

But Fairplay is highly Obfuscated

And Fairplay will be killed when SIP Off so we can not debug it directly

Poor man’s kernel debugger

uloader

- Write a macho loader for kext

```
int macho_loaded_for_each_sym(struct mach_header_64 *mh,int (^cb)

int macho_loaded_for_each_local_sym(struct mach_header_64 *mh, in
int macho_loaded_for_each_ext_def_sym(struct mach_header_64 *mh,
void* macho_loaded_resolve_sym(struct mach_header_64 *mh,const ch

int macho_loaded_for_each_local_reloc(struct mach_header_64 *mh,
int macho_loaded_for_each_external_reloc(struct mach_header_64 *m

int macho_loaded_for_each_chained_fixup(struct mach_header_64 *mh
```

uloader

- Write a macho loader for kext
- Write a runtime for imports

```
int aes_decrypt_key(const unsigned char *key, int key_len, void *ctx);
int aes_decrypt_cbc(const unsigned char *in_blk, const unsigned char *in_
                    unsigned char *out_blk, void* ctx);
void *IOMalloc(vm_size_t size);
void *IOMallocAligned(vm_size_t size, vm_offset_t alignment);
void IOFree(void *addr, size_t size);
void IOFreeAligned(void *addr, size_t size);

void *IORWLockAlloc();
void IORWLockWrite(void *lock);
void IORWLockRead(void *lock);
void IORWLockUnlock(void *lock);
void IORWLockFree(void *lock);

void *IOLockAlloc(void);
void IOUnlock(void *lock);
void IOLockFree(void *lock);
void IOLockLock(void *lock);
```

uloader

- Write a macho loader for kext
- Write a runtime for imports
- Notify debugger of kext image

```
kern_return_t kr = task_info(mach_task_self(),TASK_DYLD_INFO,(task_info_t)&dyld_info,&count);
assert(kr == KERN_SUCCESS);

struct dyld_all_image_infos *all_image_infos = (struct dyld_all_image_infos *)dyld_info.all_image_info_addr
struct mach_header_64* dyld = (struct mach_header_64*)all_image_infos->dyldImageLoadAddress;

void (*dyld_debugger_notification)(enum dyld_notify_mode mode, unsigned long count, struct mach_header_64 *
dyld_debugger_notification = macho_loaded_resolve_sym(dyld, "__dyld_debugger_notification");

assert(dyld_debugger_notification != NULL);      You, 32 minutes ago • using raw debugger interface so that

const struct dyld_image_info*  old_images = all_image_infos->infoArray;
const uint32_t old_image_count = all_image_infos->infoArrayCount;
all_image_infos->infoArray = NULL;

struct dyld_image_info*  new_images = malloc(sizeof(struct dyld_image_info) * (old_image_count  + 1));
memcpy(new_images, old_images, sizeof(struct dyld_image_info) * old_image_count);

new_images[old_image_count].imageFileModDate = mach_absolute_time();
new_images[old_image_count].imageFilePath = strdup(image_path);
new_images[old_image_count].imageLoadAddress = (const struct mach_header*)mh;

all_image_infos->infoArrayCount = old_image_count + 1;
all_image_infos->infoArray = new_images;

dyld_debugger_notification(dyld_notify_adding, 1, &mh);
```

uloader

- Write a macho loader for kext
- Write a runtime for imports
- Notify debugger of kext image
- Enjoy “Kernel” Debugging

```
0xfca0c ⟹ host_priv_self() ⇒ 0xc03
0xfca38 ⟹ host_get_special_port(priv=0xc03,node=-1,which=17,port=0x100808a10[0x
0xff1e0 ⟹ strlen("/Applications/[redacted]/WrappedBundle/[redacted]") ⇒ 0x31
0xff2b4 ⟹ kmem_alloc(map=0xa858000080000000,addrp=0x100809000,size=32,tag=0x106
0xff3b8 ⟹ vm_map_copyin(src_map=0xa858000080000000,src=0x100206930,len=32,src_c
Process 81249 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = breakpoint 2.1 3.1
    frame #0: 0x0000000100627adc FairPlayIOKit`uf_setup_from_fp
FairPlayIOKit`uf_setup_from_fp:
->  0x100627adc <+0>:  pacibsp
    0x100627ae0 <+4>:  sub    sp, sp, #0x180            ; =0x180
    0x100627ae4 <+8>:  stp    x24, x23, [sp, #0x140]
    0x100627ae8 <+12>: stp    x22, x21, [sp, #0x150]
Target 0: (uloader) stopped.
```

uloader

- Write a macho loader for kext
- Write a runtime for imports
- Notify debugger of kext image
- Enjoy “Kernel” Debugging

```
0xfca0c ⟹ host_priv_self() ⇒ 0xc03
0xfca38 ⟹ host_get_special_port(priv=0xc03,node=-1,which=17,port=0x100808a10[0x
0xff1e0 ⟹ strlen("/Applications/[redacted]/WrappedBundle/[redacted]") ⇒ 0x31
0xff2b4 ⟹ kmem_alloc(map=0xa858000080000000,addrp=0x100809000,size=32,tag=0x106
0xff3b8 ⟹ vm_map_copyin(src_map=0xa858000080000000,src=0x100206930,len=32,src_c
Process 81249 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = breakpoint 2.1 3.1
    frame #0: 0x0000000100627adc FairPlayIOKit`uf_setup_from_fp
FairPlayIOKit`uf_setup_from_fp:
->  0x100627adc <+0>:  pacibsp
    0x100627ae0 <+4>:  sub    sp, sp, #0x180            ; =0x180
    0x100627ae4 <+8>:  stp    x24, x23, [sp, #0x140]
    0x100627ae8 <+12>: stp    x22, x21, [sp, #0x150]
Target 0: (uloader) stopped.
```

Hardware Breakpoint/Memory Watchpoint are supported 😂

FairplayIOKit - open

Entrance

- Executable Path
- CPU Type
- CPU Sub Type
- **rp6S0jzg**: Handle(out, uint32_t)

uf_setup_from_fp

- Fairplay open is mainly done by a userspace daemon: fairplayd
- Work is transmitted via MIG, in function uf_setup_from_fp

```
0xfca38 ⟹ host_get_special_port(priv=0x1403,node=-1,which=17,port=0x14c809a10[0x1303])
 ⇒ 0
0xff1e0 ⟹ strlen("/Applications/[redacted]/WrappedBundle/[redacted]") ⇒ 0x34
0xff2b4 ⟹ kmem_alloc(map=0xa858000080000000,addrp=0x14c80a000,size=35,tag=0x4c808853)
⇒ 0x14c606b10
0xff3b8 ⟹ vm_map_copyin(src_map=0xa858000080000000,src=0x14c606b10,len=35,src_destroy=
0,copy_result=0x14c809a30)
0xcbb50 ⟹ mig_get_reply_port() ⇒ 0x507
0xcbb70 ⟹ mach_msg_rpc_from_kernel_proper(msg=0x16b1a73c0,send_size=0x50,recv_size=0x1
38)
[*] transforming kernel-space RPC request into user-space style
        send ool vm address 0x14c606b50, size 0x35 from 0x507 to port 0x1303
        path: /Applications/Bilibili HD.app/WrappedBundle/bili-hd2
        recv msg size = 0x128
        got ool vm1: 0x104e5c000, size: 0x39c
        got ool vm2: 0x104e58000, size: 0x28
[*] transforming user-space RPC response into kernel-space style
```

The MIG

The Unfreed MIG

From FairplayIOKit to fairplayd

fprpc.c

```c
#pragma pack(push, 4)
struct FPRequest{
    mach_msg_header_t header;
    mach_msg_body_t body;
    mach_msg_ool_descriptor_t ool;
    NDR_record_t ndr;
    uint32_t size;
    uint64_t cpu_type;
    uint64_t cpu_subtype;
};

struct FPResponse{
    mach_msg_header_t header;
    mach_msg_body_t body;
    mach_msg_ool_descriptor_t ool1;
    mach_msg_ool_descriptor_t ool2;
    uint64_t unk1;
    uint8_t unk2[136];
    uint8_t unk3[84];
    uint32_t size1;
    uint32_t size2;
    uint64_t unk5;
};
#pragma pack(pop)
```

⊗ 0 ⚠ 0 空格: 2 UTF-8 LF C ccls: 1/1 jobs

The Unfreed MIG

From FairplayIOKit to fairplayd

fprpc.c

```c
#pragma pack(push, 4)
struct FPRequest{
    mach_msg_header_t header;
    mach_msg_body_t body;
    mach_msg_ool_descriptor_t ool;
    NDR_record_t ndr;
    uint8_t unk3[64];
    uint32_t size1;
    uint32_t size2;
    uint64_t unk5;
};
#pragma pack(pop)
```

空格: 2 UTF-8 LF C ccls: 1/1 jobs

host_special_ports.h /Applications/Xcode.app/Contents

```c
#define host_get_unfreed_port(host, port)       \
    (host_get_special_port((host),              \
    HOST_LOCAL_NODE, HOST_UNFREED_PORT, (port)))
#define host_set_unfreed_port(host, port)       \
    (host_set_special_port((host), HOST_UNFREED_PORT, (port)))

#define host_get_amfid_port(host, port) \
    (host_get_special_port((host),      \
    HOST_LOCAL_NODE, HOST_AMFID_PORT, (port)))
#define host_set_amfid_port(host, port) \
```

制表符长度: 4 UTF-8 LF C++ ccls: 3/3 jobs

The Unfreed MIG

From FairplayIOKit to fairplayd

fprpc.c

```c
#pragma pack(push, 4)
struct FPRequest{
    mach_msg_header_t header;
    mach_msg_body_t body;
    mach_msg_ool_descriptor_t ool;    /path/to/exec
    NDR_record_t ndr;
    uint32_t size;
    uint64_t cpu_type;
    uint64_t cpu_subtype;
};

struct FPResponse{
    mach_msg_header_t header;
    mach_msg_body_t body;
    mach_msg_ool_descriptor_t ool1;
    mach_msg_ool_descriptor_t ool2;
    uint64_t unk1;
    uint8_t unk2[136];
    uint8_t unk3[84];
    uint32_t size1;
    uint32_t size2;
    uint64_t unk5;
};
#pragma pack(pop)
```

⊗ 0 ⚠ 0　空格: 2　UTF-8　LF　C　ccls: 1/1 jobs

The Unfreed MIG

From FairplayIOKit to fairplayd

fprpc.c

```c
#pragma pack(push, 4)
struct FPRequest{
    mach_msg_header_t header;
    mach_msg_body_t body;
    mach_msg_ool_descriptor_t ool;
    NDR_record_t ndr;
    uint32_t size;                    ool buffer size
    uint64_t cpu_type;
    uint64_t cpu_subtype;
};

struct FPResponse{
    mach_msg_header_t header;
    mach_msg_body_t body;
    mach_msg_ool_descriptor_t ool1;
    mach_msg_ool_descriptor_t ool2;
    uint64_t unk1;
    uint8_t unk2[136];
    uint8_t unk3[84];
    uint32_t size1;
    uint32_t size2;
    uint64_t unk5;
};
#pragma pack(pop)
```

0 0 空格: 2 UTF-8 LF C ccls: 1/1 jobs

The Unfreed MIG

From FairplayIOKit to fairplayd

fprpc.c

```c
#pragma pack(push, 4)
struct FPRequest{
    mach_msg_header_t header;
    mach_msg_body_t body;
    mach_msg_ool_descriptor_t ool;
    NDR_record_t ndr;
    uint32_t size;
    uint64_t cpu_type;
    uint64_t cpu_subtype;
};

struct FPResponse{
    mach_msg_header_t header;
    mach_msg_body_t body;
    mach_msg_ool_descriptor_t ool1;
    mach_msg_ool_descriptor_t ool2;
    uint64_t unk1;
    uint8_t unk2[136];
    uint8_t unk3[84];
    uint32_t size1;
    uint32_t size2;
    uint64_t unk5;
};
#pragma pack(pop)
```

0 0 空格: 2 UTF-8 LF C ccls: 1/1 jobs

The Unfreed MIG

From FairplayIOKit to fairplayd

fprpc.c

```c
#pragma pack(push, 4)
struct FPRequest{
    mach_msg_header_t header;
    mach_msg_body_t body;
    mach_msg_ool_descriptor_t ool;
    NDR_record_t ndr;
    uint32_t size;
    uint64_t cpu_type;
    uint64_t cpu_subtype;
};

struct FPResponse{
    mach_msg_header_t header;
    mach_msg_body_t body;
    mach_msg_ool_descriptor_t ool1;
    mach_msg_ool_descriptor_t ool2;
    uint64_t unk1;
    uint8_t unk2[136];
    uint8_t unk3[84];
    uint32_t size1;
    uint32_t size2;
    uint64_t unk5;
};
#pragma pack(pop)
```

0 0 空格: 2 UTF-8 LF C ccls: 1/1 jobs

fairplayd

File Access After MIG 502

15:45:25:312	1078	fairplayd	FILE_open	/Users/pwn0rz/Developer/fairplay/tmp/SC_Info/XXX.sinf
15:45:25:312	1078	fairplayd	EXEC_mmap	/Users/pwn0rz/Developer/fairplay/tmp/SC_Info/XXX.sinf
15:45:25:316	1078	fairplayd	FILE_open	/Users/pwn0rz/Developer/fairplay/tmp/SC_Info/XXX.supf
15:45:25:316	1078	fairplayd	EXEC_mmap	/Users/pwn0rz/Developer/fairplay/tmp/SC_Info/XXX.supf
15:45:25:316	1078	fairplayd	FILE_open	/Users/pwn0rz/Developer/fairplay/tmp/XXX
15:45:25:316	1078	fairplayd	EXEC_mmap	/Users/pwn0rz/Developer/fairplay/tmp/XXX
15:45:25:317	1078	fairplayd	FILE_open	/Users/pwn0rz/Developer/fairplay/tmp/XXX
15:45:25:317	1078	fairplayd	EXEC_mmap	/Users/pwn0rz/Developer/fairplay/tmp/XXX
15:45:25:317	1078	fairplayd	FILE_open	/Users/pwn0rz/Developer/fairplay/tmp/XXX
15:45:25:317	1078	fairplayd	EXEC_mmap	/Users/pwn0rz/Developer/fairplay/tmp/XXX
15:45:25:317	1078	fairplayd	FILE_open	/Users/pwn0rz/Developer/fairplay/tmp/XXX
15:45:25:317	1078	fairplayd	EXEC_mmap	/Users/pwn0rz/Developer/fairplay/tmp/XXX
15:45:25:317	1078	fairplayd	FILE_open	/Users/pwn0rz/Developer/fairplay/tmp/XXX
15:45:25:317	1078	fairplayd	EXEC_mmap	/Users/pwn0rz/Developer/fairplay/tmp/XXX
15:45:25:318	1078	fairplayd	FILE_open	/Users/pwn0rz/Developer/fairplay/tmp/XXX
15:45:25:318	1078	fairplayd	EXEC_mmap	/Users/pwn0rz/Developer/fairplay/tmp/XXX
15:45:25:604	1078	fairplayd	FILE_open	/private/var/db/fpsd/SC Info/SC Info.sidb

SInf File

The Master Key

```
bash-3.2$ tools/sinf_view.py tmp/SC_Info/ [redacted].sinf
sinf.frma: game
sinf.schm: itun
sinf.schi.user: [redacted]
sinf.schi.key : 0x00000002
sinf.schi.iviv: 0xf5632d3d20946473952366f9c51e8c2a
sinf.schi.righ.veID: 0x000561b5
sinf.schi.righ.plat: 0x00000000
sinf.schi.righ.aver: 0x01010100
sinf.schi.righ.tran: 0xdd099377
sinf.schi.righ.sing: 0x00000000
sinf.schi.righ.song: 0x1d13f514
sinf.schi.righ.tool: P550
sinf.schi.righ.medi: 0x00000080
sinf.schi.righ.mode: 0x00000000
sinf.schi.righ.hi32: 0x00000002
sinf.schi.name: [redacted]
sinf.schi.priv: [redacted]
59d484b7ddfeae5bbc5203ffcde1891127c1b5c4c5bcaa63525a8369d8eeca90d88e1c885217df45789bf4202818f09558164858772c
eadfd5410e78be3e6fd8323aaed53b68702f9e9681eb1e7c2288f55ee3d80b5f904169c552e1f60c1253ad7edd2ecf1acb0cb5f3c8ed
b42b5e84e6539f22ea42cfd06fc80046c92640a213f91205cf9bbc8b75f3ce11f02b6151631cdd2a9071f59b737ef067d07b63c2f4b4
8593493b4f66842feb59a024d7ff44c499130193525ba995eeeaa7f28a5c16e364b39b8b328e4e3ed4f7cb28ba80fb0b64b4c2693d10
bb6e1f69b20587d410ac8ebfa7840da600e69328adcd32da20bd3ed85806022dd916f9c82c4bac2e3725b5bb7d342a5023cc77543808
[redacted]
sinf.sign: 399a598101111a18a839c319191c732935f15b8bfa4b8a350f4b9d5f7e0d4b667bbfad92f7c9d2ff7b17573e26b8e68c1
021d1208f92ef33e3a18332d861aae334b1936576408287d935a26f04c3e69d0fb417dff9c2891abb89f4c561e675a753b4fff1c7ea6
54f27b0524038c7f25fae22abba5e749e8ade5b8032bbe78339
```

SInf File

The Master Key

```
bash-3.2$ tools/sinf_view.py tmp/SC_Info/ [redacted].sinf
sinf.frma: game
sinf.schm: itun
sinf.schi.user: [redacted]
sinf.schi.key : 0x00000002
sinf.schi.iviv: 0xf5632d3d20946473952366f9c51e8c2a
sinf.schi.righ.veID: 0x000561b5
sinf.schi.righ.plat: 0x00000000
sinf.schi.righ.aver: 0x01010100
sinf.schi.righ.tran: 0xdd099377
sinf.schi.righ.sing: 0x00000000
sinf.schi.righ.song: 0x1d13f514
sinf.schi.righ.tool: P550
sinf.schi.righ.medi: 0x00000080
sinf.schi.righ.mode: 0x00000000
sinf.schi.righ.hi32: 0x00000002
sinf.schi.name: [redacted]
sinf.schi.priv: [redacted]
59d484b7ddfeae5bbc5203ffcde1891127c1b5c4c5bcaa63525a8369d8eeca90d88e1c885217df45789bf4202818f09558164858772c
eadfd5410e78be3e6fd8323aaed53b68702f9e9681eb1e7c2288f55ee3d80b5f904169c552e1f60c1253ad7edd2ecf1acb0cb5f3c8ed
b42b5e84e6539f22ea42cfd06fc80046c92640a213f91205cf9bbc8b75f3ce11f02b6151631cdd2a9071f59b737ef067d07b63c2f4b4
8593493b4f66842feb59a024d7ff44c499130193525ba995eeeaa7f28a5c16e364b39b8b328e4e3ed4f7cb28ba80fb0b64b4c2693d10
bb6e1f69b20587d410ac8ebfa7840da600e69328adcd32da20bd3ed85806022dd916f9c82c4bac2e3725b5bb7d342a5023cc77543808
[redacted]
sinf.sign: 399a598101111a18a839c319191c732935f15b8bfa4b8a350f4b9d5f7e0d4b667bbfad92f7c9d2ff7b17573e26b8e68c1
021d1208f92ef33e3a18332d861aae334b1936576408287d935a26f04c3e69d0fb417dff9c2891abb89f4c561e675a753b4fff1c7ea6
54f27b0524038c7f25fae22abba5e749e8ade5b8032bbe78339
```

Unique for each user/download

SupF

One KeyPair per Page

```
bash-3.2$ tools/supf_view.py tmp/SC_Info/[redacted].supf
KeyPair Segments:
        Segment 0x0: arm_v7, Keys: 0x3d0/4k, sha1sum = df180f9fe4ad49017fc46be9c337bae3cf62559e
        Segment 0x1: arm64, Keys: 0x3fc/4k, sha1sum = 6eb062e921ca176e6fd160204efc6d2f68216d96

Fairplay Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            33:33:af:08:07:08:af:00:01:af:00:00:04
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, O=Apple Inc., OU=Apple Certification Authority, CN=Apple FairPlay Certification Authority
        Validity
            Not Before: Jul  8 00:48:27 2008 GMT
            Not After : Jul  7 00:48:27 2013 GMT
        Subject: C=US, O=Apple Inc., OU=Apple FairPlay, CN=AP.3333AF080708AF0001AF000004
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (1024 bit)
                Modulus:
                    00:cc:b6:ca:9a:f5:1c:49:e2:4d:27:54:66:33:31:
                    8b:20:b4:24:38:54:17:de:8a:46:04:4d:15:74:f5:
                    91:07:67:35:75:8b:16:38:77:eb:20:ec:b1:81:0f:
                    15:2e:f3:a5:b5:01:8b:84:8b:c4:23:62:e9:ac:4b:
                    c1:03:d0:0f:52:13:cf:4e:e1:70:25:77:82:91:14:
                    49:22:fc:53:2c:c9:c4:27:44:fc:5b:e5:13:13:a5:
                    06:ab:29:90:19:c8:65:ad:d3:5c:e1:66:26:85:ac:
                    e4:89:97:23:6a:c8:51:29:46:91:3e:82:a1:31:5c:
                    40:42:9b:1f:75:01:c2:77:73
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment, Data Encipherment, Key Agreement
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Subject Key Identifier:
                5A:44:7C:77:0D:FD:C6:B2:71:32:E8:2C:8C:37:64:0B:E3:D4:07:24
            X509v3 Authority Key Identifier:
                keyid:FA:0D:D4:11:91:1B:E6:B2:4E:1E:06:49:94:11:DD:63:62:07:59:64

    Signature Algorithm: sha1WithRSAEncryption
         5d:7f:35:ac:94:a0:87:e8:5a:20:a0:e3:ce:da:72:ae:cd:c8:
         a0:12:7d:78:9c:72:3f:de:43:56:1b:c8:b7:93:cf:26:d0:0b:
         d5:6d:a7:83:a8:b4:5e:6a:a3:c5:8d:6d:01:d9:08:b2:2e:22:
         1a:b8:ee:75:c9:0c:24:88:2e:1b:eb:ea:e2:cb:e0:50:15:2f:
         b4:c8:c9:7a:32:e2:04:17:51:61:0c:c6:78:40:0e:79:c1:59:
         29:e7:58:0e:d6:03:47:f4:c8:18:ea:d6:88:8f:d1:4b:fc:47:
         cc:1b:70:11:13:9a:06:a4:c6:e1:bc:a6:3d:67:41:7f:7a:02:
         80:31

RSA Signature: 5aebd6a2bc507ea8b62b207a37b3b6e8c3d56b6e7ff443452a93669f507066232379537d89dff98558b7627e17cb09aa7e4b3c54f978a8674d9c0cec8
a26a68a1695358fb3528f3ccfd74a263c547f3781786a2f58fdd0c4107913c47f8cb7e4cac1cc002b9ed6e2854f6edb68fcc01399da2ca86d070e637c832656e4aa9a82
```

FairplayIOKit - decryptPage

decryptPage

- rp6S0jzg: Handle(in,uint32_t)
- Crypt offset
- Encrypted Buff
- Decrypted Buffer

```
__int64 __cdecl com_apple_driver_FairPlayIOKit::bvqhJ(void *this, uint32_t handle, uint64_t pos, const char *from, char *to)
{
  __int64 *v6; // [xsp+8h] [xbp-8h] BYREF

  v6 = (__int64 *)&v6;
  return EQlZPp(handle, pos, from, to);
}
```

decryptPage

- Decrypt with aes-128-cbc
- No HW AES Engine involved
- “Page” size is 4k

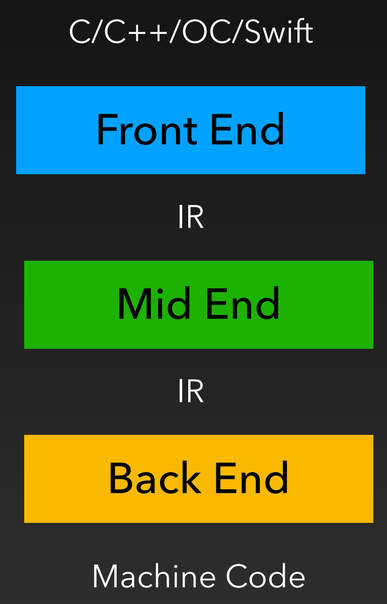
```
>>> calling fairplay decrypt page : EQlZPp(handle=0xe421a923,off=0x36e000,src=0x101e6e000,dst=0x10080e800)
0x10e2f0 ⟹ IOMallocAligned(0x4000,0x40) ⇒ 0x100808800
0xfea5c ⟹ IOMalloc(0x68) ⇒ 0x100206c80
0xfa7a4 ⟹ IOLockLock(0x1002061d0)
0xfb91c ⟹ IOUnlock(0x1002061d0)
0xfa7a4 ⟹ IOLockLock(0x100206220)
0x40924 ⟹ aes_decrypt_key(key=0x100206c82,len=0x10,ctx=0x16fdff1d0)
0x4093c ⟹ aes_decrypt_cbc(in=0x101e6e000,iv=0x100206c92,n_blk=0x100,out=0x10080e800,ctx=0x16fdff1d0)
        aes-128-cbc key: 3ea01b294cf8107fc2646b7f6acde653, iv: 0888e773bb76ef698981d0733f1b2172
0xfb91c ⟹ IOUnlock(0x100206220)
0x10c094 ⟹ IOFree(0x100206c80,0x68)
0x10e608 ⟹ IOFreeAligned(0x100808800,0x4000)
>>> calling fairplay decrypt page : EQlZPp(handle=0xe421a923,off=0x36f000,src=0x101e6f000,dst=0x10080e800)
0x10e2f0 ⟹ IOMallocAligned(0x4000,0x40) ⇒ 0x100808800
0xfea5c ⟹ IOMalloc(0x68) ⇒ 0x100206c80
0xfa7a4 ⟹ IOLockLock(0x1002061d0)
0xfb91c ⟹ IOUnlock(0x1002061d0)
0xfa7a4 ⟹ IOLockLock(0x100206220)
0x40924 ⟹ aes_decrypt_key(key=0x100206c82,len=0x10,ctx=0x16fdff1d0)
0x4093c ⟹ aes_decrypt_cbc(in=0x101e6f000,iv=0x100206c92,n_blk=0x100,out=0x10080e800,ctx=0x16fdff1d0)
        aes-128-cbc key: 660c99e61ac45c562986c89d350f2f67, iv: 97a8d8992c938907fffd89239c665fce
0xfb91c ⟹ IOUnlock(0x100206220)
0x10c094 ⟹ IOFree(0x100206c80,0x68)
0x10e608 ⟹ IOFreeAligned(0x100808800,0x4000)
Process_81249 exited with status = 0 (0x00000000)
```

By making this KEXT work, we are now able to explore more internals!!!

Obfuscation - Implementation

LLVM Pass

- Obfuscation is implemented as a LLVM Pass, which runs on LLVM Machine-Independent IR
- Famous Example: https://github.com/HikariObfuscator/Hikari



Control Flow Obfuscation - Pattern and weakness

Call Graph Recovery(Static)

- Every function call is obfuscated

```
__text:0000000000043B88 E1 0F 00 F9                STR             X1, [SP,#0x90+var_78]
__text:0000000000043B8C A8 63 00 D1                SUB             X8, X29, #-var_18
__text:0000000000043B90 09 0D 88 52+               MOV             W9, #0x36724068
__text:0000000000043B90 49 CE A6 72
__text:0000000000043B98 08 01 09 4A                EOR             W8, W8, W9
__text:0000000000043B9C A9 6C 86 52+               MOV             W9, #0x1AA53365
__text:0000000000043B9C A9 54 A3 72
__text:0000000000043BA4 08 7D 09 1B                MUL             W8, W8, W9
__text:0000000000043BA8 09 31 96 52+               MOV             W9, #0xD954B188
__text:0000000000043BA8 89 2A BB 72
__text:0000000000043BB0 08 01 09 4A                EOR             W8, W8, W9
__text:0000000000043BB4 A8 03 1F B8                STUR            W8, [X29,#var_10]
__text:0000000000043BB8 E8 23 00 91                ADD             X8, SP, #0x90+var_88
__text:0000000000043BBC A8 83 1E F8                STUR            X8, [X29,#var_18]
__text:0000000000043BC0 68 07 00 B0+               ADRL            X8, off_1303F0
__text:0000000000043BC0 08 C1 0F 91
__text:0000000000043BC8 09 01 40 F9                LDR             X9, [X8] ; _fp_dh_d2422d6a5f479457076ea15ed54da5a1
__text:0000000000043BCC A8 3F F6 F2                MOVK            X8, #0xB1FD,LSL#48
__text:0000000000043BD0 A0 63 00 D1                SUB             X0, X29, #-var_18
__text:0000000000043BD4 28 09 3F D7                BLRAA           X9, X8
__text:0000000000043BD8 A0 43 5F B8                LDUR            W0, [X29,#var_C]
__text:0000000000043BDC A8 83 5F F8                LDUR            X8, [X29,#var_8]
__text:0000000000043BE0 29 07 00 B0                ADRP            X9, #off_128168@PAGE
__text:0000000000043BE4 29 B5 40 F9                LDR             X9, [X9,#off_128168@PAGEOFF]
__text:0000000000043BE8 29 01 40 F9                LDR             X9, [X9]
__text:0000000000043BEC 3F 01 08 EB                CMP             X9, X8
__text:0000000000043BF0 81 00 00 54                B.NE            loc_43C00
__text:0000000000043BF4 FD 7B 49 A9                LDP             X29, X30, [SP,#0x90+var_s0]
__text:0000000000043BF8 FF 83 02 91                ADD             SP, SP, #0xA0
```

Call Graph Recovery(Static)

- Every function call is obfuscated
- Data obfuscation constant be be used to recover the original call graph

```
__text:0000000000043B88 E1 0F 00 F9          STR     X1, [SP,#0x90+var_78]
__text:0000000000043B8C A8 63 00 D1          SUB     X8, X29, #-var_18
__text:0000000000043B90 09 0D 88 52+         MOV     W9, #0x36724068
__text:0000000000043B90 49 CE A6 72
__text:0000000000043B98 08 01 09 4A          EOR     W8, W8, W9
__text:0000000000043B9C A9 6C 86 52+         MOV     W9, #0x1AA53365
__text:0000000000043B9C A9 54 A3 72
__text:0000000000043BA4 08 7D 09 1B          MUL     W8, W8, W9
__text:0000000000043BA8 09 31 96 52+         MOV     W9, #0xD954B188
__text:0000000000043BA8 89 2A BB 72
__text:0000000000043BB0 08 01 09 4A          EOR     W8, W8, W9
__text:0000000000043BB4 A8 03 1F B8          STUR    W8, [X29,#var_10]
__text:0000000000043BB8 E8 23 00 91          ADD     X8, SP, #0x90+var_88
__text:0000000000043BBC A8 83 1E F8          STUR    X8, [X29,#var_18]
__text:0000000000043BC0 68 07 00 B0+         ADRL    X8, off_1303F0
__text:0000000000043BC0 08 C1 0F 91
__text:0000000000043BC8 09 01 40 F9          LDR     X9, [X8] ; _fp_dh_d2422d6a5f479457076ea15ed54da5a1
__text:0000000000043BCC A8 3F F6 F2          MOVK    X8, #0xB1FD,LSL#48
__text:0000000000043BD0 A0 63 00 D1          SUB     X0, X29, #-var_18
__text:0000000000043BD4 28 09 3F D7          BLRAA   X9, X8
__text:0000000000043BD8 A0 43 5F B8          LDUR    W0, [X29,#var_C]
__text:0000000000043BDC A8 83 5F F8          LDUR    X8, [X29,#var_8]
__text:0000000000043BE0 29 07 00 B0          ADRP    X9, #off_128168@PAGE
__text:0000000000043BE4 29 B5 40 F9          LDR     X9, [X9,#off_128168@PAGEOFF]
__text:0000000000043BE8 29 01 40 F9          LDR     X9, [X9]
__text:0000000000043BEC 3F 01 08 EB          CMP     X9, X8
__text:0000000000043BF0 81 00 00 54          B.NE    loc_43C00
__text:0000000000043BF4 FD 7B 49 A9          LDP     X29, X30, [SP,#0x90+var_s0]
__text:0000000000043BF8 FF 83 02 91          ADD     SP, SP, #0xA0
```

Call Graph Recovery(Static)

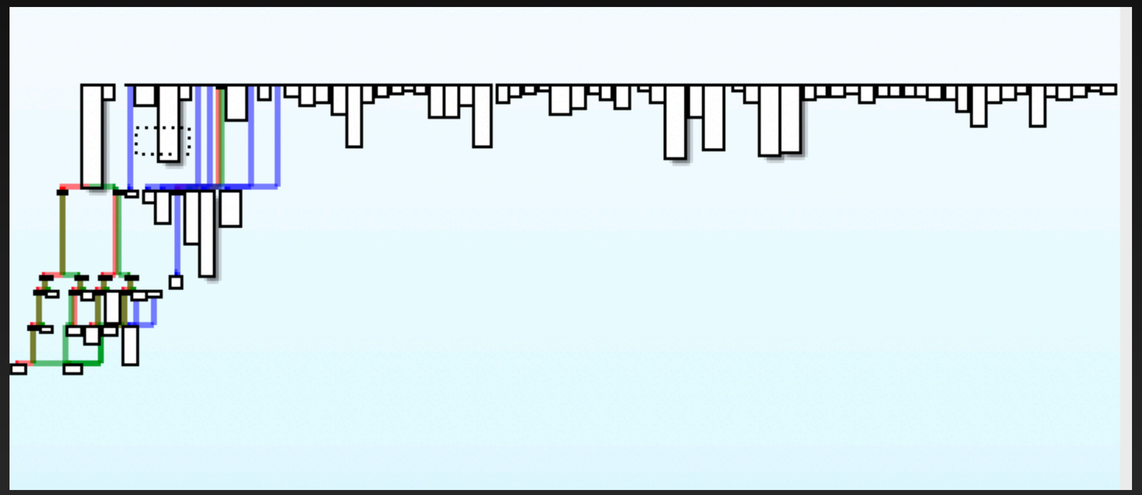
- Every function call is obfuscated
- Data obfuscation constant be be used to recover the original call graph

```
__text:0000000000043B88 E1 0F 00 F9          STR     X1, [SP,#0x90+var_78]
__text:0000000000043B8C A8 63 00 D1          SUB     X8, X29, #-var_18
__text:0000000000043B90 09 0D 88 52+         MOV     W9, #0x36724068
__text:0000000000043B90 49 CE A6 72
__text:0000000000043B98 08 01 09 4A          EOR     W8, W8, W9
__text:0000000000043B9C A9 6C 86 52+         MOV     W9, #0x1AA53365
__text:0000000000043B9C A9 54 A3 72
__text:0000000000043BA4 08 7D 09 1B          MUL     W8, W8, W9
__text:0000000000043BA8 09 31 96 52+         MOV     W9, #0xD954B188
__text:0000000000043BA8 89 2A BB 72
__text:0000000000043BB0 08 01 09 4A          EOR     W8, W8, W9
__text:0000000000043BB4 A8 03 1F B8          STUR    W8, [X29,#var_10]
__text:0000000000043BB8 E8 23 00 91          ADD     X8, SP, #0x90+var_88
__text:0000000000043BBC A8 83 1E F8          STUR    X8, [X29,#var_18]
__text:0000000000043BC0 68 07 00 B0+         ADRL    X8, off_1303F0
__text:0000000000043BC0 08 C1 0F 91
 text:0000000000043BC8 09 01 40 F9           LDR     X9, [X8] ; _fp_dh_d2422d6a5f479457076ea15ed54da5a1
                                             MOVK    X8, #0xB1FD,LSL#48
                                             SUB     X0, X29, #-var_18
                                             BLRAA   X9, X8
                                             LDUR    W0, [X29,#var_C]
                                             LDUR    X8, [X29,#var_8]
                                             ADRP    X9, #off_128168@PAGE
                                             LDR     X9, [X9,#off_128168@PAGEOFF]
                                             LDR     X9, [X9]
                                             CMP     X9, X8
                                             B.NE    loc_43C00
                                             LDP     X29, X30, [SP,#0x90+var_s0]
                                             ADD     SP, SP, #0xA0
```

```
:00000000000AC970 7F 23 03 D5                PACIBSP
:00000000000AC974 FC 6F BA A9                STP     X28, X27, [SP,#-0x10+var_50]!
:00000000000AC978 FA 67 01 A9                STP     X26, X25, [SP,#0x50+var_40]
:00000000000AC97C F8 5F 02 A9                STP     X24, X23, [SP,#0x50+var_30]
:00000000000AC980 F6 57 03 A9                STP     X22, X21, [SP,#0x50+var_20]
:00000000000AC984 F4 4F 04 A9                STP     X20, X19, [SP,#0x50+var_10]
:00000000000AC988 FD 7B 05 A9                STP     X29, X30, [SP,#0x50+var_s0]
:00000000000AC98C FD 43 01 91                ADD     X29, SP, #0x50
:00000000000AC990 FF 03 07 D1                SUB     SP, SP, #0x1C0
:00000000000AC994 F3 03 00 AA                MOV     X19, X0
:00000000000AC998 F8 63 02 91                ADD     X24, SP, #0x210+var_178
:00000000000AC99C E8 03 00 90                ADRP    X8, #off_128168@PAGE
:00000000000AC9A0 08 B5 40 F9                LDR     X8, [X8,#off_128168@PAGEOFF]
:00000000000AC9A4 08 01 40 F9                LDR     X8, [X8]
:00000000000AC9A8 A8 03 1A F8                STUR    X8, [X29,#var_60]
:00000000000AC9AC B9 6C 86 52+               MOV     W25, #0x1AA53365
:00000000000AC9AC B9 54 A3 72
:00000000000AC9B4 77 32 8B 52+               MOV     W23, #0x26A35993
:00000000000AC9B4 77 D5 A4 72
:00000000000AC9BC 08 0D 88 52+               MOV     W8, #0x36724068
:00000000000AC9BC 48 CE A6 72
:00000000000AC9C4 68 02 08 4A                EOR     W8, W19, W8
:00000000000AC9C8 08 7D 19 1B                MUL     W8, W8, W25
:00000000000AC9CC 09 08 40 B9                LDR     W9, [X0,#8]
:00000000000AC9D0 2B 01 08 4A                EOR     W11, W9, W8
:00000000000AC9D4 0D 00 40 F9                LDR     X13, [X0]
:00000000000AC9D8 A8 19 40 F9                LDR     X8, [X13,#0x30]
:00000000000AC9DC B4 0D 40 F9                LDR     X20, [X13,#0x18]
:00000000000AC9E0 A9 01 40 F9                LDR     X9, [X13]
:00000000000AC9E4 75 01 17 0B                ADD     W21, W11, W23
:00000000000AC9E8 AA 3A 29 51                SUB     W10, W21, #0xA4E
:00000000000AC9EC 2C 04 00 90+               ADRL    X12, _fp_dh_15d3f11e5574fba3d4266ee9e204f924
:00000000000AC9EC 8C 81 12 91
:00000000000AC9F4 8A D9 6A F8                LDR     X10, [X12,W10,SXTW#3]
:00000000000AC9F8 56 39 00 D1                SUB     X22, X10, #0xE
:00000000000AC9FC 0A 19 8B 52                MOV     W10, #0x58C8
:00000000000ACA00 2A 09 1F D7                BRAA    X9, X10
```

Function Boundary Recovery(Static)

- CFG Nodes might be orphan, some might not be discovered by IDA



Function Boundary Recovery(Static)

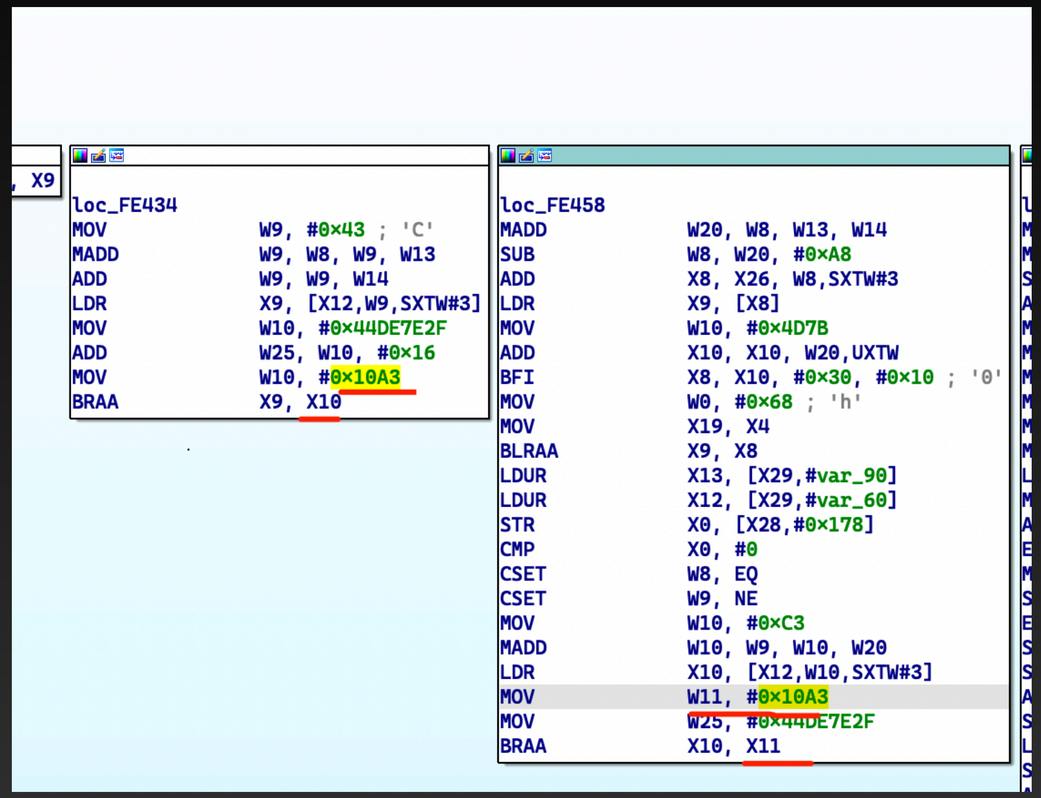
- CFG Nodes might be orphan, some might not be discovered by IDA
- Every basic block shares same PAC Context modifier for same function

```
loc_FE434
MOV     W9, #0x43 ; 'C'
MADD    W9, W8, W9, W13
ADD     W9, W9, W14
LDR     X9, [X12,W9,SXTW#3]
MOV     W10, #0x44DE7E2F
ADD     W25, W10, #0x16
MOV     W10, #0x10A3
BRAA    X9, X10
```

```
loc_FE458
MADD    W20, W8, W13, W14
SUB     W8, W20, #0xA8
ADD     X8, X26, W8,SXTW#3
LDR     X9, [X8]
MOV     W10, #0x4D7B
ADD     X10, X10, W20,UXTW
BFI     X8, X10, #0x30, #0x10 ; '0'
MOV     W0, #0x68 ; 'h'
MOV     X19, X4
BLRAA   X9, X8
LDUR    X13, [X29,#var_90]
LDUR    X12, [X29,#var_60]
STR     X0, [X28,#0x178]
CMP     X0, #0
CSET    W8, EQ
CSET    W9, NE
MOV     W10, #0xC3
MADD    W10, W9, W10, W20
LDR     X10, [X12,W10,SXTW#3]
MOV     W11, #0x10A3
MOV     W25, #0x44DE7E2F
BRAA    X10, X11
```

Function Boundary Recovery(Static)

- CFG Nodes might be orphan, some might not be discovered by IDA
- Every basic block shares same PAC Context modifier for same function
- Then it's possible to detect function boundary and add more orphan nodes



Indirect Call

- Call target is obfuscated but determined

```
__int64 __fastcall EQlZPp(int a1, __int64 a2, __int64 a3, __int64 a4)
{
  __int64 v6; // x22
  unsigned int v8; // w8
  int v9; // w27
  __int64 v10; // x19
  unsigned int v11; // w8
  unsigned int v12; // w9
  unsigned int v13; // w27
  int v14; // w9
  int v15; // w8
  __int64 v16; // x20

  v6 = a2;
  v8 = (unsigned __int8)((unsigned int)(char)((((unsigned int)a2 >> 4) & 8) - 102) >> 3) / 0x4Bu;
  v9 = 2
     * ((int)((((((char)(((_BYTE)v8 << 7) | 0x41) % 3) & 0xFFFFFFF1)
               * (((char)(((_BYTE)v8 << 7) | 0x41) % 3) & 0xFFFFFFF1)) & 0xC0) << 24) >> 28)
     + 27;
  v10 = ((__int64 (__fastcall *)(__int64, __int64))fp_dh_04861724ecb347e32e20b160688591c9[v8 + 55])(0x4000LL, 64LL);
  v11 = (int)(v9 & 0x39 ^ 0xFFFFFFC1) % 102;
  v12 = (v11 >> 1) & 0x3C;
  v13 = ((-29
        * v11
        * (v12 & v9 & (((int)(((1431655765 * (unsigned __int64)v12) >> 32) - v12) >> 1)
                     + ((unsigned int)(((1431655765 * (unsigned __int64)v12) >> 32) - v12) >> 31)
                     + 61))
        - 88) & 0xFC)
      / 0x68;
  ((void (__fastcall *)(__int64, __int64))fp_dh_04861724ecb347e32e20b160688591c9[(((v13 | 0xD0) % 0x53) >> 4) + 101])(
    v10 + 128,
    v10 + 0x4000);
  *(_DWORD *)(v10 + 40) = a1;
  *(_QWORD *)(v10 + 48) = v6;
  *(_QWORD *)(v10 + 56) = a3;
  *(_QWORD *)(v10 + 64) = a4;
  v14 = 1903446887 * (v10 ^ 0x6FF60040);
  *(_DWORD *)(v10 + 28) = 870508864 - v14;
  *(_DWORD *)(v10 + 32) = v14 + 1093796153;
```

Indirect Call

- Call target is obfuscated but determined
- Use LLDB/Frida to trace BLRAA

```
* thread #1, queue = 'com.apple.main-thread', stop reason = breakpoint 2.1
    frame #0: 0x000000010066a224 FairPlayIOKit`EQlZPp
FairPlayIOKit`EQlZPp:
->  0x10066a224 <+0>:  pacibsp
    0x10066a228 <+4>:  stp    x28, x27, [sp, #-0x60]!
    0x10066a22c <+8>:  stp    x26, x25, [sp, #0x10]
    0x10066a230 <+12>: stp    x24, x23, [sp, #0x20]
Target 0: (uloader) stopped.
(lldb) sb
Process 60585 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = instruction step into
    frame #0: 0x000000010066a2f0 FairPlayIOKit`EQlZPp + 204
FairPlayIOKit`EQlZPp:
->  0x10066a2f0 <+204>: blraa  x10, x9
    0x10066a2f4 <+208>: mov    x19, x0
    0x10066a2f8 <+212>: add    x24, x0, #0x80            ; =0x80
    0x10066a2fc <+216>: add    x25, x0, #0x4, lsl #12    ; =0x4000
[*] hit branch instruction
        source: FairPlayIOKit[0x10e2f0]: blraa  x10, x9
        target: uloader[0x100009bec]: sub    sp, sp, #0x40             ; =0x40
        symbol: IOMallocAligned
```

Indirect Call

- Call target is obfuscated but determined
- Use LLDB/Frida to trace BLRAA
- KeyPatch with IDA, let IDA eliminate dead code

```
__int64 __fastcall EQlZPp(uint32_t handle, uint64_t offset, const char *from, char *to)
{
  ObfsArgs *buff; // x19
  unsigned int v9; // w9
  __int64 v10; // x20

  buff = (ObfsArgs *)IOMallocAligned(0x4000uLL, 0x40uLL);
  LODWORD(buff->arg1) = handle;
  buff->arg2 = offset;
  buff->arg3 = (__int64)from;
  buff->arg4 = (__int64)to;
  v9 = 1903446887 * ((unsigned int)buff ^ 0x6FF60040);
  buff->context = 0x33E2E940 - v9;
  buff->constant = v9 + 0x41320139;
  buff->args_ptr = (__int64)&buff->arg1;
  buff->buff_end = (__int64)&buff->end_of_buff;
  buff->red_zone = (__int64)&buff->char80;
  fp_dh_5e7bd3a952e988b6c0416d5dac36ca7b_dispatch(buff);
  v10 = (unsigned int)(buff->retval - 1155474011);
  IOFreeAligned(buff, 0x4000uLL);
  return v10;
}
```

Indirect Branch

- w10 = qword[x12 + (EQ * 0xB + w19) << 3]
- CSET implies B.NE/B.EQ
-

```
0FEA2C ; ---------------------------------------------------------------------------
0FEA2C
0FEA2C loc_FEA2C                                ; DATA XREF: __const:0000000000131D98↓o
0FEA2C                 SUB             W9, W13, #0xB
0FEA30                 MADD            W19, W8, W9, W14
0FEA34                 SUB             W8, W19, #0x9D
0FEA38                 ADD             X8, X26, W8,SXTW#3
0FEA3C                 LDR             X9, [X8]
0FEA40                 MOV             W10, #0x4D86
0FEA44                 ADD             X10, X10, W19,UXTW
0FEA48                 BFI             X8, X10, #0x30, #0x10 ; '0'
0FEA4C                 MOV             W0, #0x68 ; 'h'
0FEA50                 STUR            X15, [X29,#var_E8]
0FEA54                 STUR            X17, [X29,#var_C0]
0FEA58                 MOV             X20, X4
0FEA5C                 BLRAA           X9, X8
0FEA60                 LDUR            X13, [X29,#var_90]
0FEA64                 LDUR            X12, [X29,#var_60]
0FEA68                 STR             X0, [X28,#0x178]
0FEA6C                 CMP             X0, #0
0FEA70                 CSET            W8, NE
0FEA74                 CSET            W9, EQ
0FEA78                 MOV             W10, #0xB
0FEA7C                 MADD            W10, W9, W10, W19
0FEA80                 LDR             X10, [X12,W10,SXTW#3]
0FEA84                 MOV             W11, #0x10A3
0FEA88                 MOV             W25, #0x44DE7E2F
0FEA90                 BRAA            X10, X11
0FEA94 ; ---------------------------------------------------------------------------
0FEA94                 STLUR           X4, [X9,#0xA2]
0FEA98
0FEA98 loc_FEA98                                ; DATA XREF: __const:00000000001315A0↓o
0FEA98                 MOV             X15, X20
0FEA9C                 MOV             X14, X0
0FEAA0                 STR             X24, [SP,#0x230+var_128]
```

Indirect Branch

- w10 = qword[x12 + (EQ * 0xB + w19) << 3]
- CSET implies B.NE/B.EQ
- Taken/Not Taken? : Not
- B.EQ XXX ? 0xb and w19 implies LUT Delta + LUT Index?

```
(lldb) sb
0xfea5c ⟹ IOMalloc(0x68) ⇒ 0x102d04080
Process 62047 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = instruction step over
    frame #0: 0x000000010065aa90 FairPlayIOKit`fp_dh_5e7bd3a952e988b6c0416d5dac36ca7b +
10776
FairPlayIOKit`fp_dh_5e7bd3a952e988b6c0416d5dac36ca7b:
→   0x10065aa90 <+10776>: braa   x10, x11
    0x10065aa94 <+10780>: .long  0xdd0a2124                ; unknown opcode
    0x10065aa98 <+10784>: mov    x15, x20
    0x10065aa9c <+10788>: mov    x14, x0
[*] hit cmp/hit instruction:
        FairPlayIOKit[0xfea6c]: cmp    x0, #0x0                  ; =0x0
        NZCV: 0 0 1 0
        condtions: ne cs hi ge gt
[*] hit branch instruction
        source: FairPlayIOKit[0xfea90]: braa   x10, x11
        target: FairPlayIOKit[0xfea98]: mov    x15, x20
        symbol: fp_dh_5e7bd3a952e988b6c0416d5dac36ca7b
```

Indirect Branch

- w10 = qword[x12 + (EQ * 0xB + w19) << 3]
- CSET implies B.NE/B.EQ
- Taken/Not Taken? : Not
- B.EQ XXX ? 0xb and w19 implies LUT Delta + LUT Index?
- Let's KeyPatch

```
0FEA30                  MADD            W19, W8, W9, W14
0FEA34                  SUB             W8, W19, #0x9D
0FEA38                  ADD             X8, X26, W8,SXTW#3
0FEA3C                  LDR             X9, [X8]
0FEA40                  MOV             W10, #0x4D86
0FEA44                  ADD             X10, X10, W19,UXTW
0FEA48                  BFI             X8, X10, #0x30, #0x10 ; '0'
0FEA4C                  MOV             W0, #0x68 ; 'h' ; size
0FEA50                  STUR            X15, [X29,#var_E8]
0FEA54                  STUR            X17, [X29,#var_C0]
0FEA58                  MOV             X20, X4
0FEA5C                  BL              _kernel_alloc ; allocate AES Key Context
0FEA60                  LDUR            X13, [X29,#var_90]
0FEA64                  LDUR            X12, [X29,#var_60]
0FEA68                  STR             X0, [X28,#0x178]
0FEA6C                  CMP             X0, #0
0FEA70                  CSET            W8, NE
0FEA74
0FEA74 loc_FEA74
0FEA74                  CSET            W9, EQ
0FEA78
0FEA78 loc_FEA78
0FEA78                  MOV             W10, #0xB
0FEA7C                  MADD            W10, W9, W10, W19
0FEA80                  LDR             X10, [X12,W10,SXTW#3] ; qword(X12 +  ((EQ * 0xb) + w19) << 3)
0FEA84                  MOV             W11, #0x10A3
0FEA88                  MOV             W25, #0x44DE7E2F
0FEA90                  B.EQ            b_fail  ; Keypatch modified this from:
0FEA90                                          ;   BRAA X10, X11
0FEA90                                          ; Keypatch modified this from:
0FEA90                                          ;   B loc_FEA98
0FEA94
0FEA94 loc_FEA94                                ; Keypatch modified this from:
0FEA94                  NOP                     ;   STLUR X4, [X9,#0xA2]
0FEA98
0FEA98 loc_FEA98                                ; DATA XREF: __const:00000000001315A0↓o
0FEA98                  MOV             X15, X20
0FEA9C                  MOV             X14, X0
0FEAA0                  STR             X24, [SP,#0x230+var_128]
0FEAA4                  STUR            X21, [X29,#var_F8]
```

Indirect Branch

- w10 = qword[x12 + (EQ * 0xB + w19) << 3]
- CMP implies B.NE/B.EQ
- Taken/Not Taken? : Not
- B.EQ XXX ? 0xb and w19 implies LUT Delta + LUT Index?
- Let's KeyPatch
- Magic Time for IDA Pro

```
v36 = 0x41320139 - 1093795995 * v28;    // fairplay decrypt page
v37 = v5;
aes_key_info = (char *)kernel_alloc(0x68uLL);
*(_QWORD *)(v1 + 376) = aes_key_info;
LODWORD(v10) = 1155431983;
if ( aes_key_info )
{
  v39 = v37;
  v40 = aes_key_info;
  v41 = v36 - 109 * (aes_key_info ≠ 0LL) + 1093795995 * (aes_key_info == 0LL);
  v42 = 238749221 * (v1 ^ 0x919C4051754C40F9LL);
  *(_QWORD *)(v1 + 8) = (char *)*(&fp_dh_4396ae8ed4349da05f55800960ab32a2 + v41) + v42;
  *(_QWORD *)(v1 + 16) = v736;
  *(_QWORD *)(v1 + 24) = v2;
  v728 = (v41 - 2001003789) ^ v42;
  *(_DWORD *)v1 = v728;
  vs = (ObfsArgs_02a1db *)v1;
  v43 = v1;
  v737 = v40;
  v44 = v39;
  fp_dh_854ff69355f49d2621e587d7db58812d(v43);// call IOLockLock
```

Data Flow Obfuscation - Pattern and weakness

Opaque Constant

- Complex Expression that evaluated to be constant for can not be simplified at compile time

For example, the call target is a function pointer that can be looked up in a table but can not be guessed easily.

```
uint32_t v3 = ((((-3 * (a1 & 0x11 | 0xAAAAAAAA)) | 0xFFFFFFEC) ^ 2) - 7) ^ 0x6E;
uint64_t v4 = fp_func_dispatch_table[((unsigned __int8)((int)((v3 - 54) & 0x7F) / -76) | 0xEAu) - 197])(0x4000LL,64LL);
```

However we will find that the index of that table is in fact an **opaque constant** after some evaluation in LLDB, and soon the result can patch to be:

```
uint64_t v4 = IOMallocAligned(0x4000LL,64LL);
```

MBA Expression

- OperatorSet(+,-,*,&,|,~)
- Simple => Complex Expression that can not be simplified
- Can be defeated via pattern matching

- Paper1: https://hal.archives-ouvertes.fr/hal-01388109/document
- Paper2: https://tel.archives-ouvertes.fr/tel-01623849/document

```
uint32_t x = ?;
uint32_t y = (x ^ 0xFFF6BBDF) + ((2 * x) & 0xFFED77BE) + 0x7B9FECF9;
```

And thanks to lldb, we can find the equivalent formula with some simple evaluation:

```
uint32_t x = ?;
uint32_t y = x - 0x7b96a8d8 = x + 0x84695728;
```

By investigation, we have found that Apple uses **only 1 MBA Formula** as the rule, that is

```
x - c == (x ^ ~c) + ( (2 * x) & ~(2 * c + 1) ) + 1;

//or
x == (x ^ ~c) + ( (2 * x) & ~(2 * c + 1) ) + c + 1
```

Take Away

https://github.com/pwn0rz/fairplay_research

EOF