



# Surreptitious Software

## Exercise

### Attacks

### Executable Information

Christian Collberg

Department of Computer Science, University of Arizona

February 26, 2014

## Introduction

`player1` is a digital rights management program. You call it like this:

```
> player1 userkey sample1 sample2 sample3
```

where `userkey` is a 32-bit cryptographic key and the samples are integers that you want to “play”. In actuality, all that happens is that decode samples are written to the file `audio`. Example:

```
> player1 0xca7ca115 10000 20000 30000 60000
```

```
Please enter activation code: 42
```

```
> cat audio
```

```
3133074688.000000
```

```
3133047808.000000
```

```
3133062912.000000
```

```
3133022208.000000
```

Figure 1 shows a block diagram of the DRM player. Figure 2 shows the actual C code.

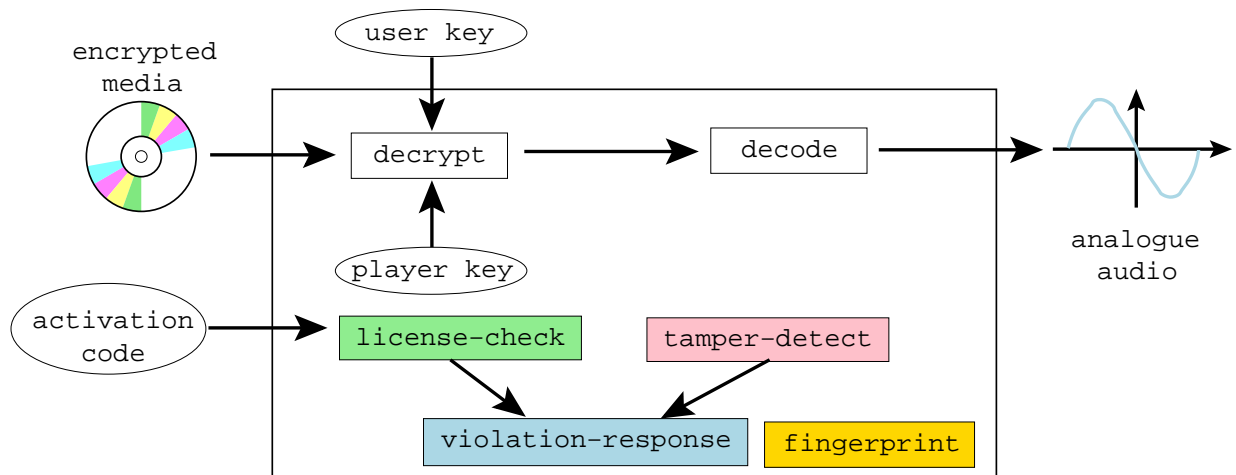


Figure 1: Block diagram of the player.

```

typedef unsigned int uint32;
typedef char* caddr_t;
typedef uint32* waddr_t;

uint32 the_player_key = 0xbabeca75;
FILE* audio;

uint32 play(uint32 user_key, uint32 encrypted_media[], int media_len) {
    int code;
    int i;
    for(i=0;i<media_len;i++) {
        uint32 key = user_key ^ the_player_key;
        uint32 decrypted = key ^ encrypted_media[i];
        if (time(0) > 1221011472) {
            fprintf(stderr,"%s!\n", "Program expired!");
            *((int*)NULL)=99;
        }
        float decoded = (float)decrypted;
        fprintf(audio,"%f\n",decoded); fflush(audio);
    }
}

uint32 player_main (uint32 argc, char *argv[]) {
    uint32 user_key = atoi(argv[1]);
    int i;
    uint32 encrypted_media[100];

    for(i=2; i<argc; i++)
        encrypted_media[i-2] = atoi(argv[i]);
    int media_len = argc-2;

    play(user_key, encrypted_media, media_len);
}

int main (uint32 argc, char *argv[]) {
    printf("This is player1. Usage: player1 0xca7ca115 10000 20000 30000 60000\n");
    audio = fopen("audio", "w");
    player_main(argc,argv);
    return 0;
}

```

Figure 2: The code.

## Prerequisites

Before working the exercise make sure you download, install, and build the following:

1. Install the following tools:

tool	url	Linux	MacOS X	Windows
<b>gdb</b>	<a href="ftp.gnu.org/gnu/gdb/">ftp.gnu.org/gnu/gdb/</a>	✂ gdb		
<b>gcc</b>		✂ gcc build-essential		
<b>objdump</b>	<a href="http://www.gnu.org/software/binutils">www.gnu.org/software/binutils</a>	✂ binutils		

2. Download program and data files:

- (a) `wget 'http://www.cs.arizona.edu/~collberg/tmp/ssx.zip'`
- (b) `unzip ssx.zip`

(c) `cd ssx/attack-defense_attack0`

3. Build the `player1` executable which you will be working on from now on:

`> make`

## Find information about the program!

1. Use the `file` command to find out what kind of executable we're dealing with

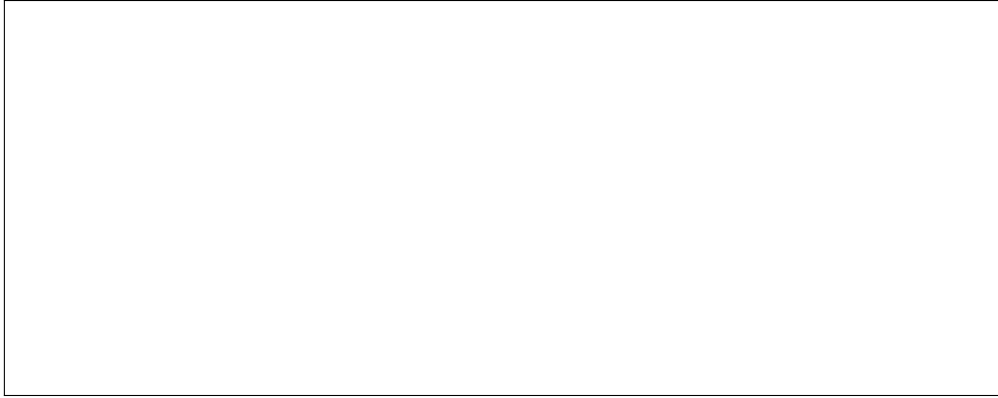
2. Use `objdump` to find the program's entrypoint.

3. Use `objdump` to find the beginning of the text segment.

p. 73



4. Use `objdump` to find the beginning of the *read-only* data segments.

A large, empty rectangular box with a thin black border, intended for the user to write the answer to question 4.

5. Use `objdump` to find out which symbols the executable has defined.

A large, empty rectangular box with a thin black border, intended for the user to write the answer to question 5.