Digital content protection
How to crack DRM and make them more resistant

Jean-Baptiste Bédrune
jean-baptiste.bedrune(at)sogeti.com
Introduction

- DRM: access control
- Limits the usage rights for a given content
- Controversial subject
- In effect, all the current systems have been broken
Plan

1. What is protecting content?
   - What is protected?
   - DRM scope

2. How to protect content?

3. Breaking DRM
Digital Rights Management (DRM)

How to avoid content to fell into the wrong hands?

Problems
- Computer: no hardware security module
  - Smartcards, etc.
- Attacker: execution in a supervised environment
  - Can steal keys, information, etc.

DRM = security through obscurity
Limit the media broadcasting

Different content, different methods

- **Software protection**
  - Very used in video games
  - Protections mixed in the software code
- **Copyright protection**
  - Online music, Video On Demand, ...
  - Control external to the content

Bad players: preventing interoperability

- Preventing open source clones: Skype, WoW, ...
- Force the use of a special hardware: wma/Zune, m4p/iPod, ...
Limit the media broadcasting

Different content, different methods

- **Software protection**
  - Very used in video games
  - Protections mixed in the software code

- **Copyright protection**
  - Online music, Video On Demand, ...
  - Control external to the content

Bad players: preventing interoperability

- Preventing open source clones: Skype, WoW, ...
- Force the use of a special hardware: wma/Zune, m4p/iPod, ...
What is protecting content? How to protect content? Breaking DRM

DRM = access control . . .

with a few constraints!

- Secrets protection: keys as well as data
- Environment: closed (ex. : iPhone, Kindle) vs. open (iTunes, Windows Media Player)
- New versions: possibility to update keys, implementation, authentication scheme, else any attack could be disastrous
- Analogic hole: analogic playing of a media being digitally protected ⇒ watermarking
- Technical constraints: speed and size, bandwidth, platform processing abilities . . .
Computers: some players

Most famous

- Microsoft: Windows Media Player, .NET, based on InterTrust
  - Licenced granted to MS by InterTrust after a trial and a 440 million $ agreement
  - Used by FNAC (France), Virgin, ... for their audio files
- Apple: iTunes Store, using FairPlay
  - Developed by Apple, can be used by Apple only
- Adobe: multimedia support in the Flash Player
  - Flash Media Server: Flash streaming server with DRM support
  - Used by Amazon, Hulu, ... for their VoD
Plan

1. What is protecting content?
2. How to protect content?
   - Components of a protection system
   - Protecting execution with anti-debugs
   - Protecting secrets with white-box cryptography
   - Protecting algorithms with code flattening
   - Protecting code with obfuscation
3. Breaking DRM
Implementation

From network to system

- Content access through a network
- Rights are given by a rights management server
- A software on the client computer manipulates the file according to the given rights
What is protecting content?
How to protect content?
Breaking DRM

Components of a protection system
Protecting execution with anti-debugs
Protecting secrets with white-box cryptography
Protecting algorithms with code flattening
Protecting code with obfuscation

Streaming
What is protecting content?
How to protect content?
Breaking DRM

Components of a protection system
Protecting execution with anti-debugs
Protecting secrets with white-box cryptography
Protecting algorithms with code flattening
Protecting code with obfuscation

Streaming
What is protecting content?  
How to protect content?  
Breaking DRM

Components of a protection system  
Protecting execution with anti-debugs  
Protecting secrets with white-box cryptography  
Protecting algorithms with code flattening  
Protecting code with obfuscation

System view

Protected media

Player

Protected file

Codec

File dissector

AAC

Jean-Baptiste Bédruine
Digital content protection How to crack DRM and make them more resistant
Security through obscurity

Goal: make the suppression of the protection as difficult as possible

- Obfuscating code
- Protecting secrets
  - Encryption keys, sensitive algorithms
- Preventing memory analysis
  - Anti-debugs
Anti-debug

Goals
- Prevent dynamic analysis of the program
- Make code stealing harder

Compatibility
- Anti-debugs must not detect false positives
  \[\Rightarrow\] Not intrusive, often easy to suppress
Examples

**Generic**
- Integrity checks

**Windows**
- IsDebuggerPresent, IsRemoteDebuggerPresent
- Detection of WinDbg debugging extensions
- Vectored exception handler to detect HWBP

**Linux, Mac OS X**
- Calls to ptrace
Protecting secrets

Context

An attacker:
- controls the execution of the program
- control all the paths;
- has access to all the data;
- and can modify them!

How to protect the keys / the secrets?
⇒ Hide them in the code
Key and messages easily recoverable

- **Plaintext data**
  - **AES_set_key**
  - **AES_encrypt**
  - **Encrypted data**
- **Key**
White-box AES: security through obscurity

White-box context
- An attacker has a full control on the code and the data
- He must not be able to extract the secrets

⇒ Algorithms obfuscation
- Key never present plaintext in memory
- Encodings on internal operations
- Message encodings
- etc.
AES: towards white-box

Plaintext message → Context

- addRoundKey
- subBytes
- shiftRows
- mixColumns
- addRoundKey

9 times

Encrypted message
AES: towards white-box

Plaintext message

- addRoundKey
- subBytes
- shiftRows
- mixColumns
- addRoundKey

9 times

Encrypted message
AES: towards white-box

Plaintext message

9 times

addRoundKey + subBytes

shiftRows

mixColumns

addRoundKey + subBytes

shiftRows

addRoundKey

Encrypted message
AES: towards white-box

Plaintext message

\[ \text{addRoundKey} + \text{subBytes} \]

\[ \text{shiftRows} \]

\[ \text{mixColumns} \]

\[ \text{addRoundKey} + \text{subBytes} + \text{addRoundKey} \]

\[ \text{shiftRows} \]

Encrypted message

9 times
What is protecting content?
How to protect content?
Breaking DRM

Components of a protection system
Protecting execution with anti-debugs
Protecting secrets with white-box cryptography
Protecting algorithms with code flattening
Protecting code with obfuscation

AES: towards white-box

Plaintext message

```
addRoundKey
+ subBytes
+ shiftRows
+ mixColumns
```

9 times

```
addRoundKey
+ subBytes
+ addRoundKey
+ shiftRows
```

Encrypted message
What is protecting content? 
How to protect content? 
Breaking DRM

Components of a protection system
Protecting execution with anti-debugs
Protecting secrets with white-box cryptography
Protecting algorithms with code flattening
Protecting code with obfuscation

AES: towards white-box

Modified plaintext message

\[
\text{addRoundKey} + \text{subBytes} + \text{shiftRows} + \text{mixColumns} \\
\text{9 times}
\]

Modified encrypted message
What is protecting content?
How to protect content?
Breaking DRM

Components of a protection system
Protecting execution with anti-debugs
Protecting secrets with white-box cryptography
Protecting algorithms with code flattening
Protecting code with obfuscation

Code flattening

PoC script (metasm) to reverse it
Code flattening

- PoC script (metasm) to reverse it
A small problem

Solutions:
- Tracing data modifications
- Praying
Code modifications: what does this function?

Step 1: unprotected function

```c
int f(int x) {
    return 16 * x + 16;
}
```
What is protecting content?

How to protect content?

Breaking DRM

Components of a protection system

Protecting execution with anti-debugs

Protecting secrets with white-box cryptography

Protecting algorithms with code flattening

Protecting code with obfuscation

Code modifications: what does this function?

Step 2: addition / subtraction

```c
int f(int x) {
    return 129441535 - (-16*x + (129441535 - 16));
}
```
Code modifications: what does this function?

Step 3: expansion

```c
int f(int x) {
    return 129441535 - 1793574399 * (-16*x * 3488889343 + 129441535 * 3488889343 - 16 * 3488889343);
}
```

1793574399 * 3488889343 = 1 \mod 2^{32}
Code modifications: what does this function?

Step 4: another expansion

```c
int f(int x) {
    return 129441535 - 1793574399 * (1584987567
    * (−16*x * 3488889343 * 3319691599 + 129441535
    * 3488889343 * 3319691599) − 16 * 3488889343);
}
```
Code modifications: what does this function?

Step 5: reduction

```c
int f(int x) {
    return 129441535 - 1793574399 * (1584987567
                                * (3781768432 * x + 2881946191) - 4282621936);
}
```
Plan

1. What is protecting content?

2. How to protect content?

3. Breaking DRM
   - Retrieving unprotected content
   - Finding decryption functions in QuickTime
   - Code stealing
   - Authentication: DRM module for all
   - DRM: big playground for 0days?
3 points to look at:

- Can you produce a decrypted video?
- Is authentication secure?
- Can you retrieve encryption keys?
Audio capture

What is protecting content?
How to protect content?
Breaking DRM

Retrieving unprotected content
Finding decryption functions in QuickTime
Code stealing
Authentication: DRM module for all
DRM: big playground for 0days?

Audio capture: Converting AAC to MP3
What is protecting content?
How to protect content?
Breaking DRM

Retrieving unprotected content
Finding decryption functions in QuickTime
Code stealing
Authentication: DRM module for all
DRM: big playground for 0days?

Video capture
Bad solutions?

- Time consuming
- Tedious
- Quality loss during recompression
- Loss of metadata
Retrieving the stream

Retrieving the plaintext stream $\Leftrightarrow$ producing an unencrypted file

Cracking tool

<table>
<thead>
<tr>
<th>File header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flag: encrypted file</td>
</tr>
</tbody>
</table>

Legit player

- Search
- Replace

decrypt_stream

Plaintext chunk
Retrieving the plaintext stream ⇔ producing an unencrypted file

- Cracking tool
  - file1.m4v
  - File header
    - Flag: encrypted file
  - Header
    - Plaintext chunk
  - Header
    - Encrypted chunk
  - Header
    - Encrypted chunk
  - Header
    - Encrypted chunk
  - Header
    - Encrypted chunk

- Legit player
  - decrypt_stream
  - Encrypted chunk
  - Plaintext chunk

Search → Replace
Retrieving the plaintext stream $\Leftrightarrow$ producing an unencrypted file
Retrieving the plaintext stream ⇔ producing an unencrypted file
Memory leakages: plaintext streams

Diagram:
- **DRM module**
  - Outputs: Encrypted chunk, Plaintext chunk
- **Protected media**
- **File dissector**
- **Codec**

Key terms:
- **Encrypted chunk**
- **Plaintext chunk**
- **File dissector**
- **Codec**

Discussion:
- What is protecting content?
- How to protect content?
- Breaking DRM
- Retrieving unprotected content
- Finding decryption functions in QuickTime
- Code stealing
- Authentication: DRM module for all
- DRM: big playground for 0days?
Memory leakages: plaintext streams

Protected media -> DRM module
  Encrypted chunk + Player key
  Chunk encrypted with the player key
  Decryption with DRM key + Encryption with player key
  File dissector -> Decryption -> Codec -> Music
Memory leakages: plaintext streams

Protected media

File dissector

Decryption

Modified chunk

Modified codec

DRM module

Decryption with DRM key

Encryption with player key

Encrypted chunk

+ Player key

Chunk encrypted with the player key

Jean-Baptiste Bédruine
Digital content protection How to crack DRM and make them more resistant
Codec scrambling

Want steganography?

Plaintext chunk

Valid header  Marker  Encoded data

000000240605217E421F63B381563C609746F1B6059A950100003C64419A0C186008AB7F03C0E3C3F7AF4F640F

00000024060521F833002F858A36B045DCB80BB775C604B3B9719AE185A19ECF3B7E209E8FF5B601000040
Finding key functions

Interactions with DRM module?

- CoreFP module (QuickTime)
- 5 exported functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Ordinal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win9UJ86JKdV4dM</td>
<td>7C812110</td>
<td>1</td>
</tr>
<tr>
<td>X46051eS</td>
<td>7C821200</td>
<td>2</td>
</tr>
<tr>
<td>YlCJ31g</td>
<td>7C82D110</td>
<td>3</td>
</tr>
<tr>
<td>lxpgvVMLd0S7uR1</td>
<td>7CA4CE80</td>
<td>4</td>
</tr>
<tr>
<td>DllEntryPoint</td>
<td>7C801040</td>
<td></td>
</tr>
</tbody>
</table>
Finding key functions

Interactions with DRM module?

- CoreFP module (QuickTime)
- 5 exported functions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Ordinal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win9UJ86JKdV4dM</td>
<td>7C812110</td>
<td>1</td>
</tr>
<tr>
<td>X46051eS</td>
<td>7C821200</td>
<td>2</td>
</tr>
<tr>
<td>YlCJ31g</td>
<td>7C82D110</td>
<td>3</td>
</tr>
<tr>
<td>1xpgvVMLd0S7uRl</td>
<td>7CA4CE80</td>
<td>4</td>
</tr>
<tr>
<td>DllEntryPoint</td>
<td>7C801040</td>
<td></td>
</tr>
</tbody>
</table>

- Called hundreds of times per minute! Decryption function?
What is protecting content?
How to protect content?
Breaking DRM

Retrieving unprotected content
Finding decryption functions in QuickTime
Code stealing
Authentication: DRM module for all
DRM: big playground for 0days?

Obfuscated pointers

Obfuscated parameters

DRM module (YICJ3lg)
What is protecting content?
How to protect content?
Breaking DRM

Retrieving unprotected content
Finding decryption functions in QuickTime
Code stealing
Authentication: DRM module for all
DRM: big playground for 0days?

Obfuscated pointers

Obfuscated parameters

Obfuscated parameters

DRM module (YlCJ3lg)
## Code stealing

- Recoding all the algorithms takes time
- Several functions are very well protected
- Global process has been understood, but subroutines have not been analysed

⇒ Stealing the code from the software

- The cracking program calls the protected program
Stealing code: injection

Cracking tool
- read_keystore
- parse_audio_file
- write_decrypted_stream

Legit player
- read_keystore
- decrypt_keystore
- parse_audio_file
- decrypt_stream
- play_stream
Stealing code: rip functions

Cracking tool

- read_keystore
- decrypt_keystore
- parse_audio_file
- decrypt_stream
- write_decrypted_stream

Legit player

- read_keystore
- decrypt_keystore
- parse_audio_file
- decrypt_stream
- play_stream

Copy

Breaking DRM

- What is protecting content?
- How to protect content?
- Retriving unprotected content
- Finding decryption functions in QuickTime
- Code stealing
- Authentication: DRM module for all
- DRM: big playground for 0days?
White-box cryptography

Goal

- Goal = avoiding keys / secrets to be retrieved
- Are keys necessary?

- Code stealing
  - Removing code obfuscation
  - Removing algorithm obfuscation
    - Time consuming
    - ... but the cracking method can be re-used
Major problem: external encodings must be **external**!
White-box cryptography: easy breaking

Major problem: external encodings must be **external**!
Authentication : DRM module for all

- Challenge / response (RSA signature)
- Only authorized players can use the DRM module

Player

32 bytes challenge

Player certificate

+ 

Signed challenge

CoreFP
White-box RSA. 3 possibilities:

- Extract RSA private key
- Rip RSA code + boxes + encoded private key
- Replay attack

Authentication: DRM module for all

Player certificate + Signed challenge

32 bytes challenge
Replay attack

- Extract QuickTime player certificate

Player → CoreFP

32 bytes challenge

QuickTime certificate + Signed challenge
Replay attack

- Full control of CoreFP
- Fixed challenge/response, sniffed from QuickTime response

**Diagram**

- **Player**
- **CoreFP (Hooked)**
- Fixed 32 bytes challenge
- QuickTime certificate + Fixed signed challenge
Vulnerabilities?

- Code difficult to analyze
- Undocumented file formats
  ⇒ Not really audited

2 bugs found during analysis, without any work.
VLC 1.1.4 (Windows)

Good tool for analysis, verbose output
- Module that handles QuickTime DRMs
- Old code, not working anymore

Source code
- This is so ugly and uses so many MD5 checksums that it is most certainly one-way, though why it needs to be so complicated is beyond me.
- This is even uglier.
**VLC 1.1.4 (Windows)**

Good tool for analysis, verbose output
- Module that handles QuickTime DRMs
- Old code, not working anymore

**Source code**
- This is so ugly and uses so many MD5 checksums that it is most certainly one-way, though why it needs to be so complicated is beyond me.
- This is even uglier.

Opened a video bought from the iTunes store...
- Crash on opening ⇒ Exploitable? :)

Retrieving unprotected content
Finding decryption functions in QuickTime
Code stealing
Authentication: DRM module for all
DRM: big playground for 0days?
iTunes 10.0.0.68

- Removed a box to avoid VLC crashing
  - $skcr$, information about zones to be decrypted
- Fields initialized to zero by iTunes
- Division by one of these fields $\Rightarrow$ Crash
Conclusion

**DRM = Code protection**

- Methods used = the ones used for intellectual property protection

**No system stays unbroken for a long time**

- Content can be read ⇒ Content can be unprotected
- Major problem: deprotecting is easier than protecting
  - Sometimes, understanding a minor part of the code is enough
- Once the protection has been broken, it is difficult to really improve it
  - Compatibility with previous versions
  - A whole new implementation is time consuming
Conclusion

The video games case

- One/two attackers per protection
  - ... almost full-time
- First cracking: a few weeks
- Development of automatic tools: a few weeks
- Protection improvement: a few months
- New cracking, development of new cracking tools: a few days
Questions?

- What is protecting content?
- How to protect content?
- Breaking DRM
- Retrieving unprotected content
- Finding decryption functions in QuickTime
- Code stealing
- Authentication: DRM module for all
- DRM: big playground for 0days?

Jean-Baptiste Bédru...
Laboratoire Sogeti-ESEC
6-8 rue Duret
75016 Paris - France

jean-baptiste.bedrune@sogeti.com