Audible DRM scheme

Dhiru Kholia

March 19, 2017

v1.05

KTH Royal Institute of Technology

About Dhiru

- Looking inside the (Drop) box (WOOT '13 paper)
- John the Ripper developer
- @DhiruKholia on Twitter
- `https://github.com/kholia`

Anti-Piracy Notice

You may use this work to backup only your own Audible audiobooks.
Respect the authors, they need to eat too.

Responsible Disclosure

All vulnerabilities, which are described in this paper, have been reported to the vendor. Unfortunately, some of them cannot be patched or repaired, practically speaking.

Full Disclosure

This work was completed around July, 2015 when I was employed at Spotify AB. Needless to say, this caused ”some” friction at work.

Agenda

- About Audible
- DRM problems
- Existing Work
- Reversing Audible
- Borrowing audio books
- Breaking DRM
- DEMO :-)

About Audible

- Largest seller, producer, and retailer of digital audio books
- Supports over 600 ”AudibleReady” devices (even GPS devices, cars)
- But not my Linux computer!

DRM (Digital Rights Management) problems

- No support for free systems like Linux, FreeBSD, and Rockbox
- I can't use Audible audiobooks on my mp3 player!
- Outages, Obsolescence, Accessibility
- Increases piracy, reduces profits (Preyas Desai, 2009)
- Flies in the face of traditional media sharing culture? (e.g. sharing books)

DRM graveyard?

- Microsoft Zune, Microsoft Reader
- Fictionwise
- Sony Reader Store
- Amazon PDF and LIT ebooks

DRM in a coffee machine?

DRM in near future?

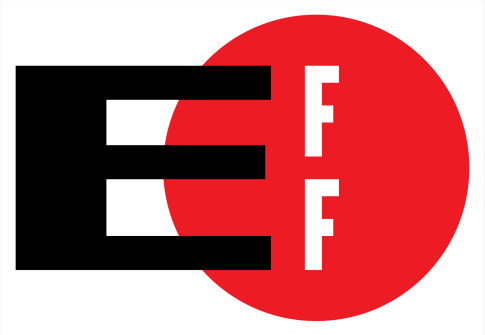
From Elysium (2013) movie. How absurd can DRM get?

DMCA

- Digital Millennium Copyright Act (law against DRM circumvention)
- DRM and reversing is a tricky area
- DeCSS (DVD Jon)
- United States v. ElcomSoft and Dmitry Sklyarov (props to EFF)
- Chilling effect (e.g. Niels Ferguson)

Sanity prevails in Europe?

A July 2015 European Parliament report emphasizes problems with ”portability and geoblocking” and notes that ”lack of interoperability hampers innovation, reduces competition and harms the consumer” (Jeremy Malcolm, EFF)

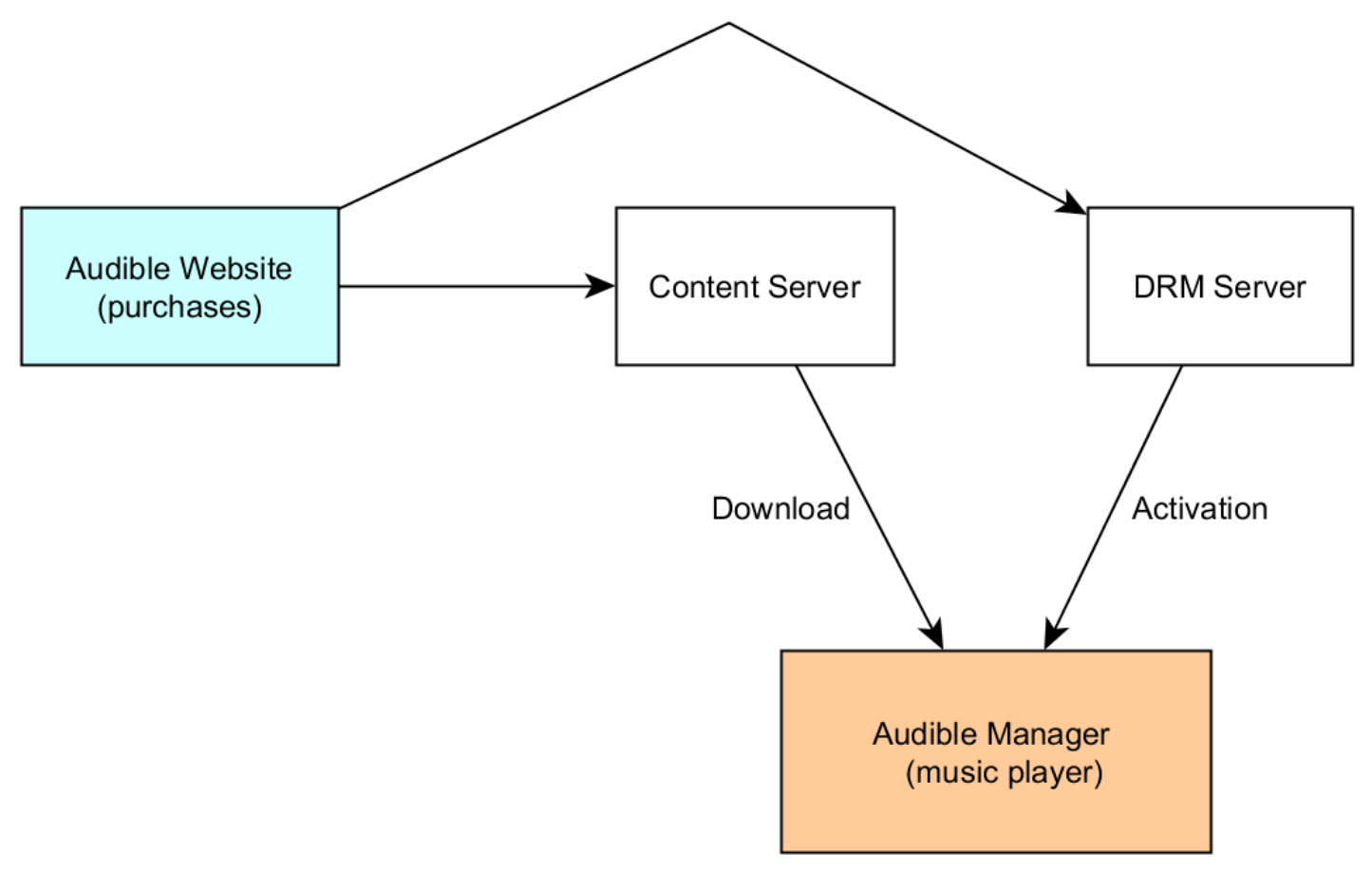


Support EFF and FSF’s Defective by Design campaign.

Our Position

- DRM is absurd but it is sometimes a practical necessity
- We understand that implementing DRM is sometimes necessary (or at least a forced choice)
- ”Reconciling Mozilla’s Mission and W3C EME” (Web DRM)
- Our work must only be used for making personal backups of your own content!

Audible - How it works?



Official Audible Software

- Audible Inc. depends on GStreamer (an open source multimedia framework) on Kindle devices, but sadly chooses not to support the underlying FOSS ecosystem itself

- Audible Manager (AM) cannot handle Audible produced AAX files, which are over 2GB!

AA, AAX and AAX+ file formats

- AAX (and AAX+) are the top-tier Audible audio quality formats
- .aax files are almost M4B files (modified MPEG-4 container format with proprietary extensions)
- MediaInfo is able to extract metadata from .aax files
- AA file format is proprietary and dates back to 1995
- .aa files internally can use different three different codecs

Existing Work

- Audible DRM has already been bypassed multiple times (commercially too!)
- DRM sometimes bypassed due to ”features” in the official software supplied by Audible itself (See GoldWave case)
- Jakob Heinemann’s aax2wav ”specifications”
- AAX to MP3 (sf.net)
- Some other projects (which essentially rely on Jakob Heinemann’s approach)

Limitations of existing work

- Mostly equivalent to analog loophole attack?
- Transcoding is inevitable (audio quality losses)
- All Audible audio formats aren't handled
- Still relies on upstream proprietary DLLs (AAXSDKWin.dll)
- Dependant on Audible servers for authentication, activation, and key distribution
- This said, this existing work is good enough, and sufficient for all practical purposes

Our Results

- Credentials, Audible account, authentication, and activation are *not* needed any more!
- We are able to play arbitrary .aa and .aax files!
- Lossless extraction of audio data is possible
- All the Audible file formats (AA, AAX, AAX+) are fully supported!
- Not a simple DRM bypass but a complete break of the entire Audible system

”Borrowing” audio books

- We found a way to abuse the book reviews to download audio books for free!
- Book reviews expose private cust_id values belonging to reviewers. These values when combined with public product_sku in a special URL, allow downloading audiobooks for free.
- We can exploit known fans of an author in case of no published reviews!
- We reported this bug to Amazon, fixed now.

Audible DRM

Downloaded books can’t be played!

DRM stripping + reversing strategies

- Wine with patched audio driver
- INSOMNIA approach (DBI with DynamoRIO)
- Easy DBI with Frida
- The IDA Pro way

Wine with patched audio API

- Audible Manager still uses the old WinMM audio API
- We patch the WinMM audio API in Wine to not block on audio output
- Less than 10 lines of new code
- The audiobook is dumped very quickly to a file
- Patch is available on GitHub (github.com/kholia/wine)

DynamoRIO

- DynamoRIO is a dynamic binary instrumentation (DBI) framework
- It allows us to hook (and skip) original function calls, change sleep(1) to nop, for example
- Similarly, we can intercept audio function calls
- Our plugin emulates ALSA and PulseAudio sound API functions, allows faster-than-realtime DRM stripping
- Available at GitHub (github.com/kholia/dynamorio)

DBI with Frida (2017)

- Frida is an easy-to-use, super fast dynamic binary instrumentation framework
- Our plugin called AudioShark emulates ALSA, PulseAudio and SDL API functions
- Generic in nature and allows faster-than-realtime DRM stripping
- Available at GitHub (github.com/kholia/SoundCore)

Reversing Audible

- FindCrypt2 + IDA Pro combination works well
- Combine static analysis with dynamic analysis from x86 and Android versions
- ”Audible for Android” is cleaner to decompile and reason about
- Hopper is pretty convenient for a quick look

Audible Manager (target)

- Official player for MS Windows
- Uses HTTP (snooping on the underlying API is easy)
- Authenticates using ROT(n) cipher
- Reported to Amazon, fixed in late-2015
- Find key derivation, and decryption functions
- Locate and trace calls to SHA-1 and AES (found by FindCrypt2)

Authentication ”Security”

```
def password_encoder(password):  # rot(8)
    shift = 8
    output = "8"

    for c in password:
        t = ord(c) - shift
        output = output + "{:02x}".format(t)

    return output

# username and password_encoder(password) are posted below
url = "http://www.audible.com/cgi-bin/licensemgr.cgi"
```

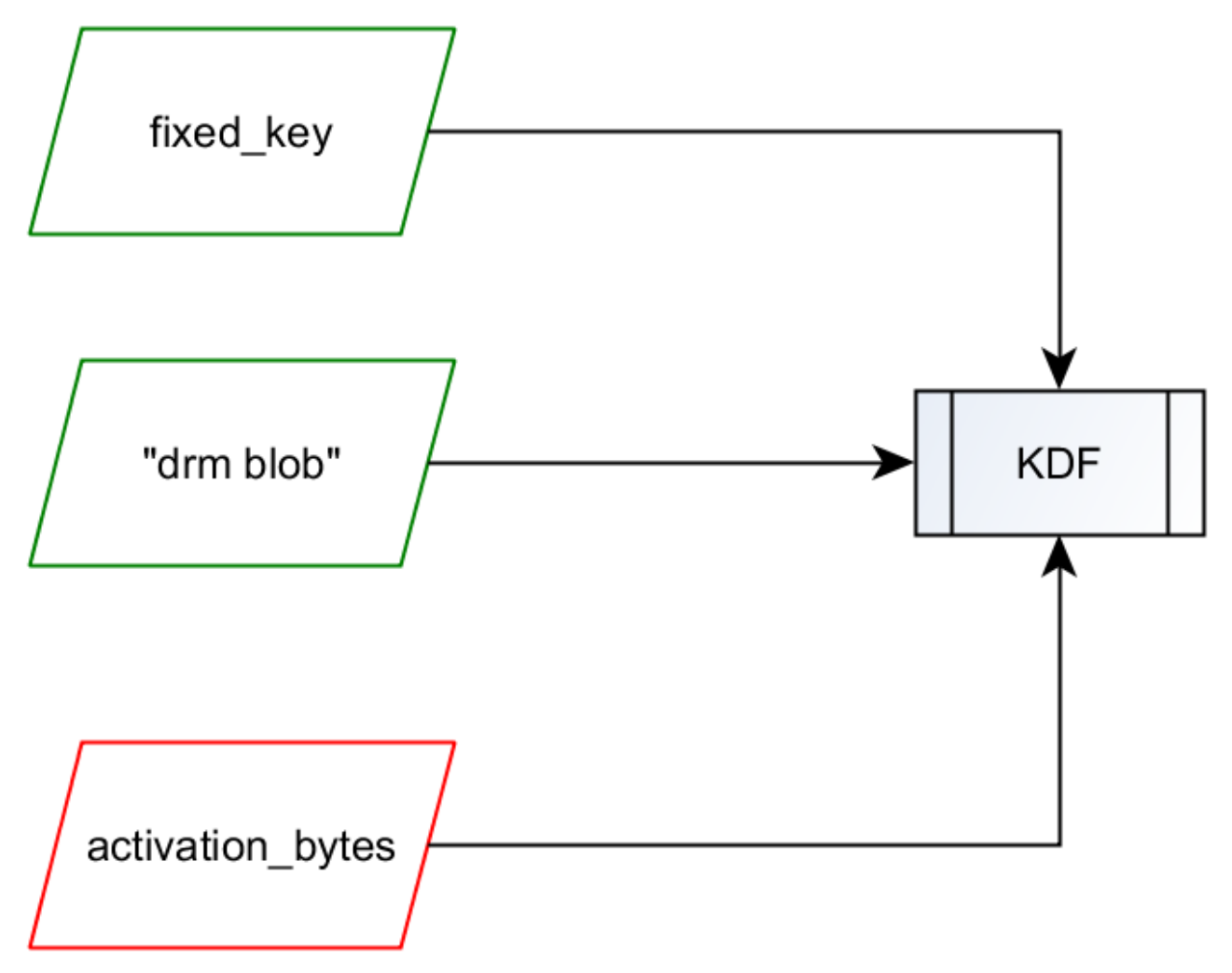
AAX / AAX+ Key Derivation

- Work backwards to see where the ”material” for key derivation is coming from
- Windows Registry
- Some cryptographic material comes from the .aax file, we call this the ”drm blob”
- Sysinternals Process Explorer and Rohitab’s API Monitor are really awesome

Snooping on the Audible Manager activation process

- Easy to see the network traffic (isn't HTTP awesome?)
- Activation data is written to the Windows Registry
- Trace registry reads (using API Monitor)
- Link registry keys with activation data and the key derivation function
- We now have all the information involved in the key derivation process!

Inputs for Key Derivation



Weaknesses in the KDF

- Unknown ”activation_bytes” value is only 4 bytes, reducing the input key-space to 2^{32} :-)
- This AES key derivation process is easily broken by Rainbow Tables, in under 30 seconds (on a 400$ Chromebook)
- A user needs to use the Rainbow Tables only once (for all his purchases, including the ones to come)
- For details see ”mov_read_adrm” function in mov.c file

AAX KDF in a nutshell

- Read the ”adrm” atom (”drm blob”) from input.aax file
- Grab activation bytes from registry (or use rainbow tables against the embedded sha1 checksum)
- Do a bunch of SHA1 operations on a 16-byte fixed key and activation data (outputs are calc_checksum, intermediate key, and IV)
- Confirm that calc_checksum == embedded_checksum
- Decrypt the ”drm blob” using AES-128 in CBC mode
- The decrypted ”data blob” undergoes further SHA1 operations, and finally produces a key + IV which is specific to input.aax file

Game Over?

AA file format and Key Derivation

- AA format dates back to year 1995
- AA files use Tiny Encryption Algorithm (TEA) with a fixed key
- Took us a longer time to reverse engineer
- ”Signsrch” tool detects the usage of TEA immediately but we did not know about Signsrch earlier
- For details see aadec.c file

Mitigations and fixes

- Fixing the whole AAX KDF weakness requires changing a single byte (size of activation_bytes)!
- Upgrading over 600 AudibleReady devices is the real challenge
- Why did Audible introduce this weakness? Was it by design?

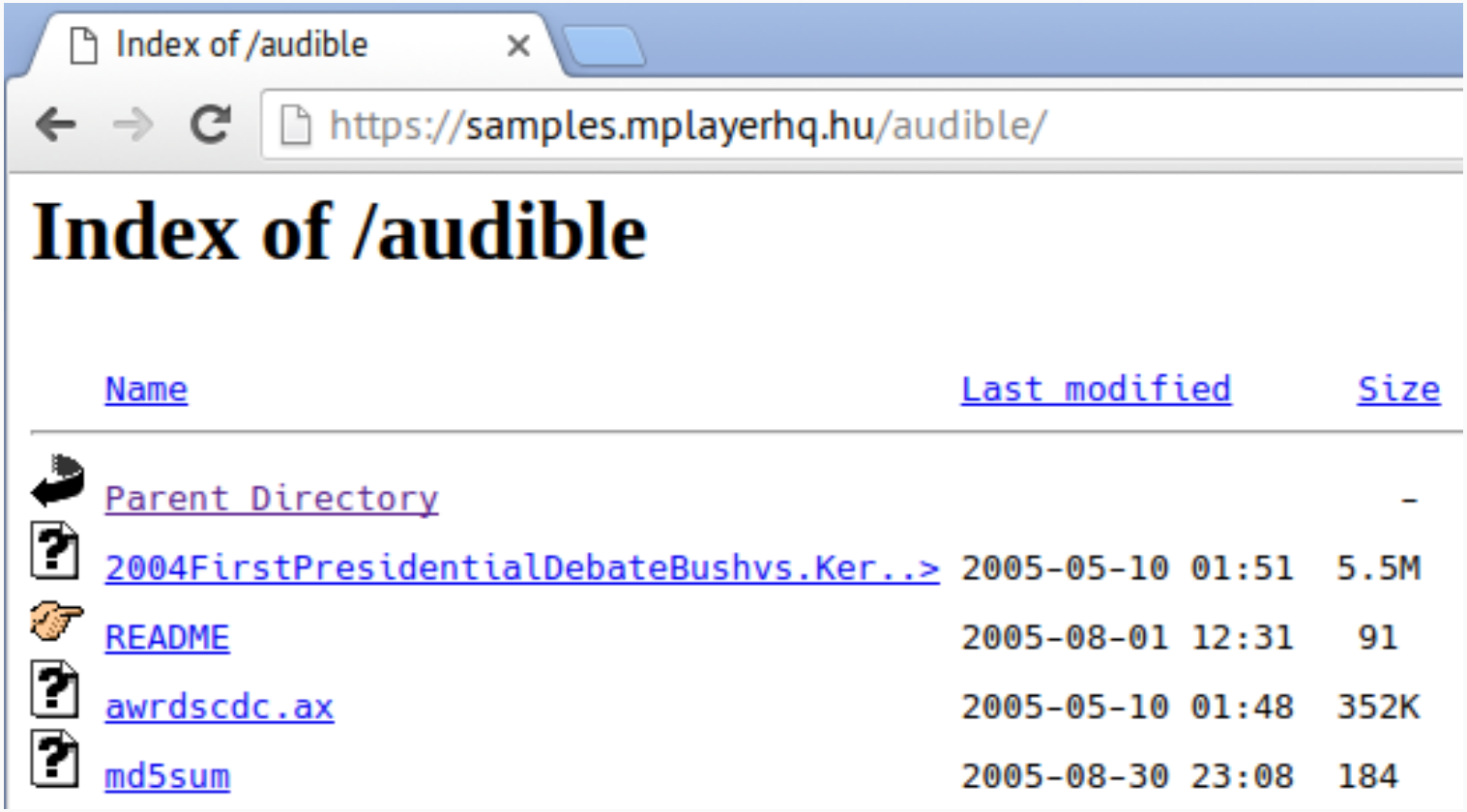
Responsible Disclosure

- All these problems were reported to Amazon's security team (with full details, and POCs) around April of 2015.

DEMO!

- Crack secret ”activation bytes” using Rainbow Tables
- Losslessly extract audio data (personal content only!)
- Play Audible files directly with FFmpeg

Results



A sample AA file from year 2005 is now playable using FFmpeg!

Credits

- Jakob Heinemann
- RainbowCrack, for a working ”rtgen” binary
- Jim Teeuwen, Header parser for Audible.com audio book files
- #ffmpeg-devel on freenode and ffmpeg-devel mailing list
- ExifTool

Credits

- Game Over image is ©2009-2015 Gironimo (DeviantArt)
- Keurig coffee machine image is borrowed from an article written by Josh Dzieza
- Sad smiley is borrowed from simpleicon.com
- happy.jpg is borrowed from lifeinspirationtoday.com

Source Code

Source code will be made public at the following URL,

`https://github.com/kholia/inAudible`

Thanks

- Alex Rad
- Friends and family for their support

Questions?

Thank You!