



CrowdStrike

EXPLOITING A COALMINE

ABUSING COMPLEX BUGS IN WEBKIT'S
RENDERARENA

GEORG WICHESKI

SENIOR SECURITY RESEARCHER

WEDNESDAY, APRIL 11, 2012

WEBKIT

- Based on KHTML (KDE)
 - Apple forked in 2001
- Chrome, (Mobile) Safari, **Android Browser**, Qt, PS 3 Vita, ...
 - Rule of Thumb: If it's not Internet Explorer or Firefox, it uses WebKit
- SLOC: 753,572 (Android 2.3.5)
 - If that's not enough: libpng, libtiff, ...

TCMALLOC



DLMALLOC

```
#if PLATFORM(ANDROID)
    #define WEBCORE_NAVIGATOR_VENDOR "Google Inc."
    #define USE_SYSTEM_MALLOC 1
    ...
```

- bionic (Android's libc) uses Doug Lea's malloc
 - This is the same allocator glibc uses
 - Without safe unlinking checks
- DLMalloc coalesces adjacent free chunks
 - No per thread caches or free-lists



DOCUMENT OBJECT MODEL TREE

What is the Document Object Model?

Editors

Philippe Le Hégaré, W3C
Lauren Wood, SoftQuad Software Inc., WG Chair
Jonathan Robie, Texcel (for DOM Level 1)

Introduction

The Document Object Model (DOM) is an application programming interface ([API](#)) for valid [HTML](#) and well-formed [XML](#) documents. It defines the logical structure of documents and the way a document is accessed and manipulated. In the DOM specification, the term "document" is used in the broad sense - increasingly, XML is being used as a way of representing many different kinds of information that may be stored in diverse systems, and much of this would traditionally be seen as data rather than as documents. Nevertheless, XML presents this data as documents, and the DOM may be used to manage this data.

With the Document Object Model, programmers can build documents, navigate their structure, and add, modify, or delete elements and content. Anything found in an HTML or XML document can be accessed, changed, deleted, or added using the Document Object Model, with a few exceptions - in particular, the DOM [interfaces](#) for the XML internal and external subsets have not yet been specified.

As a W3C specification, one important objective for the Document Object Model is to provide a standard programming interface that can be used in a wide variety of environments and [applications](#). The DOM is designed to be used with any programming language. In order to provide a precise, language-independent specification of the DOM interfaces, we have chosen to define the specifications in Object Management Group (OMG) IDL [\[OMGIDL\]](#), as defined in the CORBA 2.3.1 specification [\[CORBA\]](#). In addition to the OMG IDL specification, we provide [language bindings](#) for Java [\[Java\]](#) and ECMAScript [\[ECMAScript\]](#) (an industry-standard scripting language based on JavaScript [\[JavaScript\]](#) and JScript [\[JScript\]](#)).

Note: OMG IDL is used only as a language-independent and implementation-neutral way to specify [interfaces](#). Various other IDLs could have been used ([\[COM\]](#), [\[JavaIDL\]](#), [\[MIDL\]](#), ...). In general, IDLs are designed for specific computing environments. The Document Object Model can be implemented in any computing environment, and does not require the object binding runtimes generally associated with such

The screenshot displays a web browser's developer tools interface. The top toolbar includes tabs for Elements, Resources, Network, Scripts, Timeline, Profiles, Audits, and Console. The 'Elements' tab is active, showing a DOM tree with the following structure:

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
  <head>...</head>
  <body>
    <div class="navbar" align="center">...</div>
    <div class="noprint" style="text-align: right">...</div>
    <div class="div1">
      <a id="Introduction" name="Introduction"></a>
      <h1 id="Introduction-h1" class="div1">What is the Document Object Model?</h1>
    </div>
    <div class="div2">...</div>
    <!-- div2 ID-E7C3082 -->
    <div class="div2">...</div>
    <!-- div2 ID-E7C30821 -->
    <div class="div2">...</div>
  </body>
</html>
```

The 'h1#Introduction-h1' element is selected, and its 'Computed Style' is shown on the right. The style includes:

- font: 170% sans-serif; (from W3C-REC.css:39)
- color: #005A9C; (from W3C-REC.css:38)
- background: white;

The breadcrumb at the bottom of the developer tools reads: html > body > div.div1 > h1#Introduction-h1.

RENDER TREE

“At the heart of rendering is the render tree. The render tree is very similar to the DOM in that it is a tree of objects, where each object can correspond to the document, elements or text nodes. The render tree can also contain additional objects that have no corresponding DOM node.

The base class of all render tree nodes is `RenderObject`.”

<http://www.webkit.org/blog/114/webcore-rendering-i-the-basics/>

RENDER OBJECT CREATION

- The Render Tree is updated every time rendering changes
 - This includes when objects are repositioned and text flow changes
 - Resizing Window, Scrolling (on Android anyway), ...
- A simple DOM Text Node can get additional associated Render Tree children just by resizing your window
- Allocations and deallocations happen very frequently

THE RENDERARENA

“YO DAWG, I HEARD YOU LIKE ALLOCATORS, SO I PUT AN ALLOCATOR INTO YOUR ALLOCATOR, SO YOU CAN ALLOCATE WHILE YOU’RE ALLOCATING!”

- RenderArena is the allocator for RenderObjects
- A RenderArena consists of multiple Arenas that are allocated with... fastMalloc!
 - Recall that fastMalloc is an alias for DLmalloc on Android



RENDERARENA INTEGRATION

```
void* RenderObject::operator new(size_t sz, RenderArena* renderArena) throw()  
{  
    return renderArena->allocate(sz);  
}
```

- Every Render Tree element is derived from RenderObject
 - operator new is inherited by every Render*
- All allocations for RenderObjects happen on the Arena sub-heap!
 - Unfortunately, this means also nothing else can be allocated there.

RENDERARENA ALLOCATION



- Allocation sizes rounded up to 8 bytes
 - Only for alignment, low bits are meaningless
- Attempts to recycle a free chunk of requested size
 - Simple single-linked list, much like FreeList
- Simple forward allocation (current & limit pointers)
 - No chunk headers or other inline information

RENDERARENA DEALLOCATION



- Free chunks are put into a single-linked list
 - Pointer to next free chunk is first 32bit word in chunk
- There is no coalescing of free chunks!
 - This allows for easier (sub-)heap massaging

ENTER THE COALMINE

- There is a lot of bugs in the Render Tree
 - And they are mostly considered “just crashes”
 - Fixes are not backported for Android, takes some time until they end up in Chrome mainline (after being public on Webkit Trac)
- Invalid Casts / Type Confusion
 - Pass around `RenderObject *`, cast to `Render*` with wrong expectations
- Use-after-free
 - Happens when stuff gets removed due to re-CSS-ing

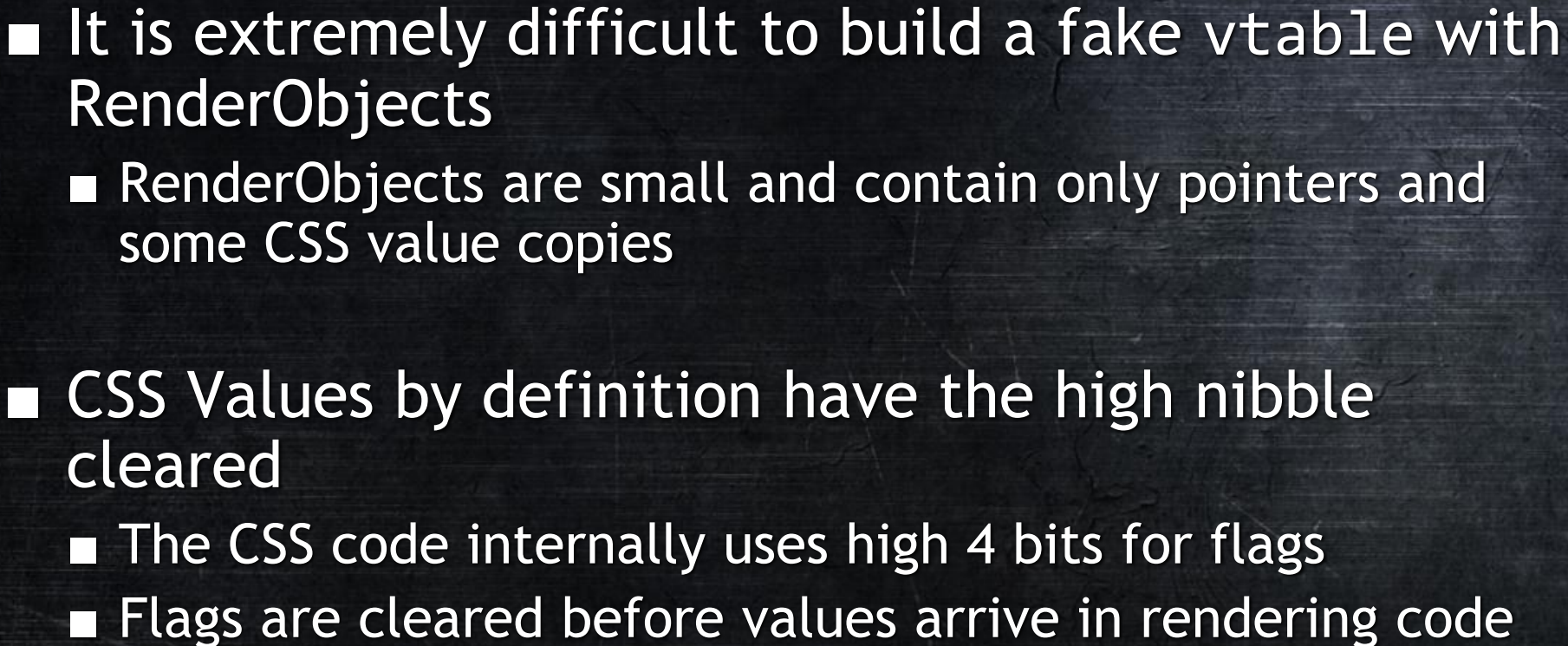
USE-AFTER-FREE EXPLOITATION

“THE WICHESKI”



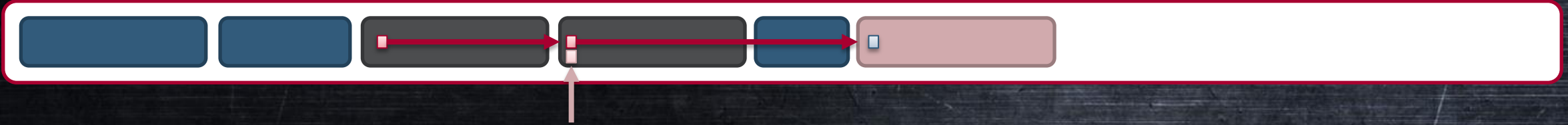
- All allocations in the RenderArena are by definition C++ objects
 - RenderObject has virtual functions, so all allocations have a vtable
- vtable overlaps with the free chunks single-linked list pointers
 - a. Free element that resembles fake vtable
 - b. Trigger free of buggy element, so it points to fake vtable
 - c. Trigger Use-After-Free virtual call

“THE WICHESKI”

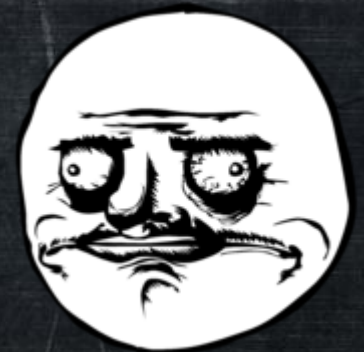


USE-AFTER-FREE EXPLOITATION

“THE WICHESKI”



- The heap is rwx on Android $\leq 2.2.1$ and is reachable
- There is one plain integer that is copied
 - `` List Item *value* (for setting numbered list item values)
 - For our convenience, assigned to two consecutive members
- Read the code, we can even get a pointer to an arbitrary long list of integers we control...



USE-AFTER-FREE EXPLOITATION

“THE REFINED AUBIZZIERE”



- RenderArena allocations come from the system allocator
 - We can control memory contents if not overwritten
- a. Spray Arena sized objects (e.g. strings) and free holes
- b. Free small dummy element at end of Arena
- c. Trigger Use-After-Free virtual call

USE-AFTER-FREE EXPLOITATION

“THE REFINED AUBIZZIERE”

- + Enables arbitrary vtable contents
- Requires reliable heap allocation and free primitives
 - We cannot “just create strings in JavaScript” because of GC
 - unescape (praised by Immunity) is only one free
 - DLmalloc will happily coalesce chunks, split large chunks, ...
 - Have fun debugging this with GDB!
- We need something better (and faster) than GDB to debug the heap allocations!



CROWDSTRIKE ANDROPROBE

```
enabled hooking points:
    AFD14520 ( absolute address ), tag BABEFACE, r0
    A8403422 ( absolute address ), tag BABECAFE, r0
    A84033EC ( absolute address ), tag ABADCAFE, r1
    A84033D4 ( absolute address ), tag 500DCAFE, r2
    libc +00014510 (symbol " malloc"), tag ABAD1DEA, r0
    libc +0001452C (symbol " free"), tag 500D1DEA, r0
found `svc 0' instruction at AFD0AF78h
code has been allocated at AFCFF000h
assembling trampoline for AFD14520 at AFCFF03C
    encountered pop {..., pc} instruction at copy offset 00
assembling trampoline for A8403422 at AFCFF054
    encountered pop {..., pc} instruction at copy offset 00
assembling trampoline for A84033EC at AFCFF06C
    building LDR link to original code at AFCFF084h
assembling trampoline for A84033D4 at AFCFF090
    building LDR link to original code at AFCFF0AAh
assembling trampoline for AFD14510 at AFCFF0B8
    cloning constant #1 from AFD14524h for load: LDR r3, =0002D00Ch
    cloning constant #1 from AFD14528h for load: LDR r2, =FFFFFF28h
    building B link to original code at AFCFF0CCh
assembling trampoline for AFD1452C at AFCFF0DC
    cloning constant #1 from AFD14540h for load: LDR r3, =0002CFF0h
    cloning constant #1 from AFD14544h for load: LDR r2, =FFFFFF28h
    building B link to original code at AFCFF0F0h
passed buffer location 48B07000h to debuggee
everything in place, resuming execution
]
```

```
]
ελεγχόμενες οι διευθ. εκτέλεσης
δράση ρυθμ. τοποθ. 48B07000h το debuggee
ρητορική η τμήκ το ολίσθητ code στ νεφέλες
χρονική συνάρτ #1 έlow AFD14524h το τοπ: ΓΩX L3' =FFFFFF38h
χρονική συνάρτ #1 έlow AFD14528h το τοπ: ΓΩX L3' =0003CFF0h
ελεγχόμενες οι συνάρτ το νητο23C στ νεφέλες
```





CrowdStrike