SESSION ID: HTA-W04

Understanding the Attack Surface and Attack Resilience of EdgeHTML

Mark Vincent Yason
Security Researcher, IBM X-Force
IBM
@MarkYason
Agenda

- Introduction
- Initial Recon
- Attack Surface
- Exploit Mitigations
- Conclusion
Introduction

Understanding the Attack Surface and Attack Resilience of EdgeHTML
The web is your canvas. Write or type directly on web pages with Web Note, then easily share your brilliance with others.
Overview

EdgeHTML Attack Surface Map & Exploit Mitigations

The web is your canvas.

Write or type directly on web pages with Web Note, then easily share your brilliance with others.
Initial Recon

Understanding the Attack Surface and Attack Resilience of EdgeHTML
MSHTML and EdgeHTML

- EdgeHTML is forked from Trident (MSHTML)
- Problem: Quickly identify major code changes (features/functionalities) from MSHTML to EdgeHTML
- One option: Diff class names and namespaces
Diffing MSHTML and EdgeHTML

**MSHTML.DLL**
- `CFontSystemQuery::s_genericFontFamilyMap`
- `ChtmPre::Tokenize(...)`
- `CImgTaskWmf::Decode(...)`

**Namespace List**
(Unique + Sorted)
- `CFontSystemQuery`
- `ChtmPre`
- `CImgTaskWmf`

**EdgeHTML.DLL**
- `CFontSystemQuery::MatchByGenericFamily(...)`
- `CHtmPre::Exec(...)`
- `CXPathEvaluator::Evaluate(...)`

**Namespace List**
(Unique + Sorted)
- `CFontSystemQuery`
- `CHtmPre`
- `CXPathEvaluator`

**Diff**
- `- CImgTaskWmf`
- `+ CXPathEvaluator`
Suggests change in image support:
- CImgTaskEmf
- CImgTaskWmf

Suggests new DOM object types:
+ CFastDOM::{...more...}
+ CFastDOM::CXPathEvaluator
+ CFastDOM::CXPathExpression
+ CFastDOM::CXPathNSResolver
+ CFastDOM::CXPathResult
+ CFastDOM::CXSLTProcessor
Suggests ported code from another rendering engine (Blink) for Web Audio support:

+bink::WebThread
+WebCore::AnalyserNode
+WebCore::AudioArray<float>
+WebCore::AudioBasicInspectorNode
+WebCore::Audio{...more...}
Further analysis needed
- Renamed class/namespace results into a new namespace plus a deleted namespace

Requires availability of symbols
- Bindiffering is another option

Same rudimentary diffing method can be applied to:
- Function and method names
- Strings
- Imports and exports
Attack Surface

Understanding the Attack Surface and Attack Resilience of EdgeHTML
Legend for the next slides

- EdgeHTML class is the entry point for parsing/processing
  - Most use other EdgeHTML classes
  - Analysis can start by setting a breakpoint on the listed EdgeHTML class methods, i.e.:
    (WinDbg)> bm edgehtml!CXmlPre::*
- HTML & CSS parsing are done by EdgeHTML classes
- XmlLite is used for parsing XML-based markups, MSXML6 is used for XML transformation
- VML support (binary behaviors) was removed in EdgeHTML
Lightweight XML parser

Built-in Windows component

*IXmlReader* interface is used by EdgeHTML for reading nodes from XML-based markups
Comprehensive XML parser

Built-in Windows component

*IXMLDOMDocument* interface is used by EdgeHTML for transforming XML that references an XSL stylesheet
Image Decoding

- Reachable via: direct link, `<img>`, `<embed>`
- Supported image formats: `g_rgMimeInfolmg`
- PNG, JPG, GIF, DDS, TIFF, BMP, HDP, ICO decoding via Windows Imaging Component (WIC)
- WMF and EMF support via GDI was removed in EdgeHTML
Image Decoding: Windows Imaging Component (WIC)

- Image decoder/encoder for multiple image formats
- Built-in Windows component
- `IWICImagingFactory::CreateDecoder()` is used by EdgeHTML to instantiate the decoder for a particular image format
Audio/Video Decoding

- Reachable via: direct link, `<audio>`, `<video>`
- Supported audio/video containers: `g_rgMimeInfoAudio` and `g_rgMimeInfoVideo`
- MP4, MP3, WAV support via Media Foundation (MF)
- TTML & WebVTT support for timed text tracks (captioning) via `<track>`
Audio/Video Decoding: Media Foundation (MF)

- Framework for audio/video processing
- Built-in Windows component
- `IMFMediaEngine` is used by EdgeHTML to setup the media source and control playback
Font Rendering

- Reachable via: `@font-face` CSS rule
- TTF, OTF and WOFF (after TTF/OTF extraction) font support via DirectWrite
- EOT font support was removed in EdgeHTML
  - Removed dependence to T2EMBED and GDI for EOT font parsing
Font Rendering: DirectWrite

- DirectX Text Rendering API
- Built-in Windows component
- Parses the font in the user-mode process where it (DWrite.dll) is hosted
- `IDWriteFactory::CreateCustomFontFileReference()` is used by EdgeHTML to register a custom private font
- DirectWrite is discussed in the “One font vulnerability to rule them all” presentation [1]
DOM API

- Reachable via: JavaScript
- Large attack surface that:
  - Interacts directly with EdgeHTML DOM objects
  - Interacts indirectly with internal EdgeHTML objects and libraries (depends)
DOM API calls can change the state of the DOM tree, DOM objects and other internal EdgeHTML objects
Unexpected input, unexpected state changes or incorrect state when a DOM API is called can result to memory corruption such as: use-after-frees (above), heap overflows, invalid pointer access, etc.
80 new DOM object types were found in EdgeHTML (GA build)

- New code or new code paths that are reachable

+CFastDOM::{...more...}
+CFastDOM::CVideoTrack
+CFastDOM::CVideoTrackList
+CFastDOM::CWaveShaperNode
+CFastDOM::CXMLElement
+CFastDOM::CXMLElementList
+CFastDOM::CXMLHttpRequest
+CFastDOM::CXMLHttpRequestUpload
+CFastDOM::CXPathEvaluator
+CFastDOM::CXPathExpression
+CFastDOM::CXPathNSResolver
+CFastDOM::CXPathResult
+CFastDOM::CXSLTProcessor
Enumerating DOM object properties/methods via JavaScript and IDA...
DOM API: Diffing DOM Object Properties and Methods

- ... and then diffing them to find out new properties / methods in already-existing DOM object types
  - New code or new code paths that are reachable

```javascript
{...more...}
+document.evaluate
document.execCommand
document.execCommandShowHelp
+document.exitFullscreen
document.fgColor
-document.fileCreatedDate
{...more...}
```
Flash and PDF Renderers

- Built-in/pre-installed complex renderers that can be instantiated by default
  - Additional set of attack surface
  - PDF: Edge is also the default PDF viewer on Windows 10
- Functionalities can be repurposed for exploitation
  - CFG bypass (via Flash JIT - now mitigated) [2]
  - ASLR bypass (via Flash Vector - now mitigated) [3]
Flash and PDF Renderers: Adobe Flash Player

- Pre-installed 3rd party component since Windows 8
- Flash is used by attackers to compromise the browser process via:
  - Flash vulnerability + Flash functionality (e.g. Vector) for mitigation bypass (CVE-2015-0311 exploit)
  - Browser vulnerability + Flash functionality (e.g. Vector) for mitigation bypass (CVE-2014-0322 exploit)
Flash and PDF Renderers: WinRT PDF Renderer

- Built-in Windows component since Windows 8.1
  - Relatively new compared to the previously described Windows components
- Component is favorable to fuzzing
  - Directly accessible via the Windows Runtime API (Windows.Data.Pdf namespace)
  - Complicated file format parsing means more opportunities for bugs
**Attack Surface Summary**

- Well-known attack vectors were removed
  - VML (VGX)
  - EMF (GDI)
  - WMF (GDI)
  - EOT (T2EMBED, GDI)

- New attack vectors were found in the DOM API
  - New DOM object types/properties/methods (New code or code paths)

- Remotely-reachable libraries via EdgeHTML
  - XMLLite
  - MSXML6
  - Windows Imaging Component
  - Adobe Flash Player
  - Media Foundation
  - DirectWrite
  - WinRT PDF Renderer
Exploit Mitigations

Understanding the Attack Surface and Attack Resilience of EdgeHTML
Exploit Mitigations

- Discussion of exploit mitigations applied to:
  - Content process that hosts EdgeHTML
  - EdgeHTML and its dependencies
  - Specific to EdgeHTML

- Known/published bypass or weakness researched/discovered by various security researchers are discussed and [referenced]
MicrosoftEdgeCP.exe: 64-bit, ASLR (HEASLR, ForceASLR), DEP, and AppContainer
#RSAC

**Edge Content Process Mitigations:**
Comparison with IE11 and ImmersiveIE

<table>
<thead>
<tr>
<th></th>
<th>Win10/Edge</th>
<th>Win10/IE11</th>
<th>Win8/ImmersiveIE</th>
<th>Win8/IE11</th>
<th>Win7/IE11</th>
</tr>
</thead>
<tbody>
<tr>
<td>64-bit</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ASLR</td>
<td>Yes (HEASLR, ForceASLR)</td>
<td>Yes (ForceASLR)</td>
<td>Yes (HEASLR, ForceASLR)</td>
<td>Yes (ForceASLR)</td>
<td>Yes (ForceASLR)</td>
</tr>
<tr>
<td>DEP</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Process</td>
<td>AppContainer</td>
<td>Low</td>
<td>AppContainer</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Isolation</td>
<td>Integrity</td>
<td></td>
<td>Integrity</td>
<td>Integrity</td>
<td>Integrity</td>
</tr>
</tbody>
</table>

- Comprehensive exploit mitigations are applied to the Edge content process (MicrosoftEdgeCP.exe) that hosts EdgeHTML (edgehtml.dll)
Edge Content Process Mitigations: Known Bypass/Weakness

- 64-bit
  - Relative heap spraying (depends) [4,5]

- ASLR+DEP
  - Memory content disclosure (via vulnerabilities) [3,6]

- AppContainer
  - Kernel vulnerabilities [7,8]
  - Vulnerabilities in the broker or higher-privileged processes [9,10,11]
  - Leveraging writable resources [9]
EdgeHTML & Dependencies Mitigations: Buffer Security Check (/GS)

- **Purpose**: Detect stack buffer overflows
- **Known bypass/weakness**: Controllable stack buffer pointer/index [1,12]
EdgeHTML & Dependencies Mitigations: Control Flow Guard (CFG)

- **Purpose:** Detect and prevent abnormal control flow
- **Recently introduced and well-researched** [13,14]
- **Several weaknesses and bypass techniques had been discovered (and mitigated) since its introduction**
EdgeHTML & Dependencies Mitigations: Known CFG Bypass/Weakness

- Flash: JIT-generated code [2]
  - Now mitigated by JIT-generating a CFG check when generating CALLs
- Chakra JS engine: CFG check function pointer overwrite [15] and leveraging unchecked indirect jmps [16,17]
  - These are also mitigated but they illustrated additional CFG bypass techniques
- Jumping to a valid API address [5], stack data overwrite [13,5], more [5]...
EdgeHTML Mitigations: Virtual Table Guard (VTGuard)

- **Purpose**: Detect an invalid virtual function table

- **Known bypass/weakness**: Applied only to select EdgeHTML classes and bypassed if address of `__vtguard` is leaked
EdgeHTML Mitigations:
Memory GC (MemGC)

- **Purpose:** Mitigate exploitation of use-after-frees
- **First introduced in EdgeHTML (Edge) and MSHTML (IE11) on Win10**
  - Now in MSHTML (IE11) on earlier Windows versions [18]
- **Improvement and successor to Memory Protector [19]**
  - Checks MemGC chunks, registers and the stack for references
  - Uses a separate managed heap (MemGC heap) and a concurrent mark-and-sweep garbage collector
EdgeHTML Mitigations: Memory GC (MemGC) Heap in Edge x64
EdgeHTML Mitigations: Known MemGC Bypass/Weakness

- No known bypass for covered cases as of writing
- MemGC internals were documented and weaknesses (conservative GC, cross-heap pointers, etc.) were identified [20]
Exploit Mitigations Summary

- Comprehensive exploit mitigations are applied to the content process
  - Time-consuming/costly exploit development

- Additional exploit mitigations are applied to EdgeHTML and its dependencies
  - A number of vulnerabilities will be unexploitable or very difficult to exploit
Conclusion

Understanding the Attack Surface and Attack Resilience of EdgeHTML
New attack vectors in rendering engines will be introduced in the parsing of new markup/style specs and in the DOM API to support new web standards.

New attack vectors in EdgeHTML are balanced by the comprehensive exploit mitigations in place.

Interesting research topics related to EdgeHTML (internals, audit, fuzzing, bypass):

- XmlLite
- MSXML6
- Windows Imaging Component
- Media Foundation
- DirectWrite
- WinRT PDF Renderer
- MemGC
Apply What You Have Learned

- **Users**: Use the 64-bit version of Windows 10 (Edge will run 64-bit)

- **Users**: If Flash is not required, disable it in Edge via *Settings > Advanced settings > Use Adobe Flash Player = Off*

- **Software Developers**: Enable exploit mitigations in your software (DEP, ASLR, HEASLR, ForceASLR, /GS, CFG)

- **Software Developers**: Revisit your code, draw an attack surface map, and then remove unnecessary attack vectors

- **Security Researchers**: Look at the security posture of EdgeHTML and its dependencies by documenting their internals and performing audits/fuzzing
Link of the detailed whitepaper is available at the end of the following blog post:


All information is based on Microsoft Edge running on 64-bit Windows 10 build 10240 (GA build)

edgehtml.dll version 11.0.10240.16384
References
(More are in the whitepaper)


References
(More are in the whitepaper)


References
(More are in the whitepaper)


References
(More are in the whitepaper)


Thank You!

Mark Vincent Yason
Security Researcher, IBM X-Force
yasonm[at]ph[dot]ibm[dot]com
@MarkYason

Understanding the Attack Surface and Attack Resilience of EdgeHTML