RSA Conference2016
San Francisco | February 29 – March 4 | Moscone Center

Connect to Protect

SESSION ID: HTA-W04

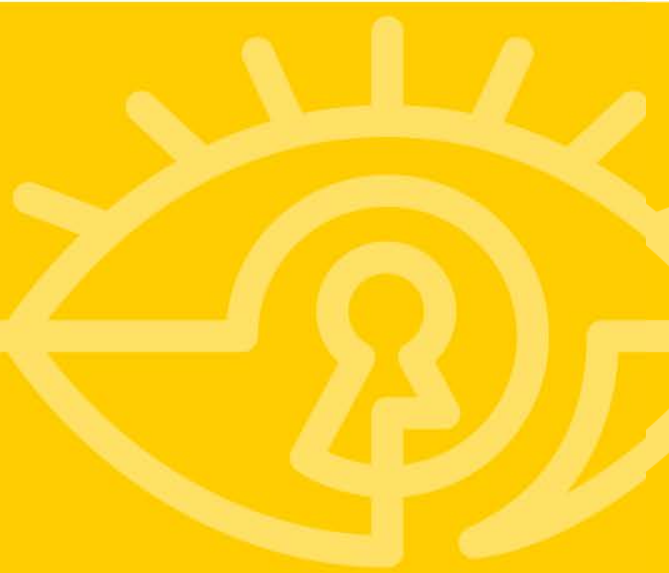
Understanding the Attack Surface and Attack Resilience of EdgeHTML

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

#RSAC

Agenda

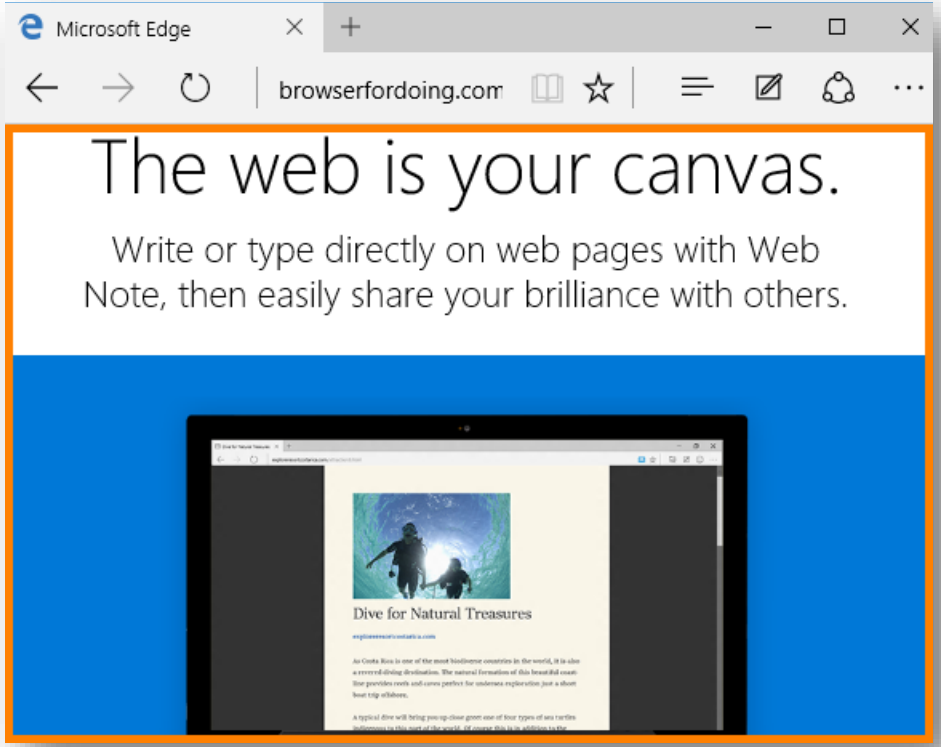
- Introduction
- Initial Recon
- Attack Surface
- Exploit Mitigations
- Conclusion



Introduction

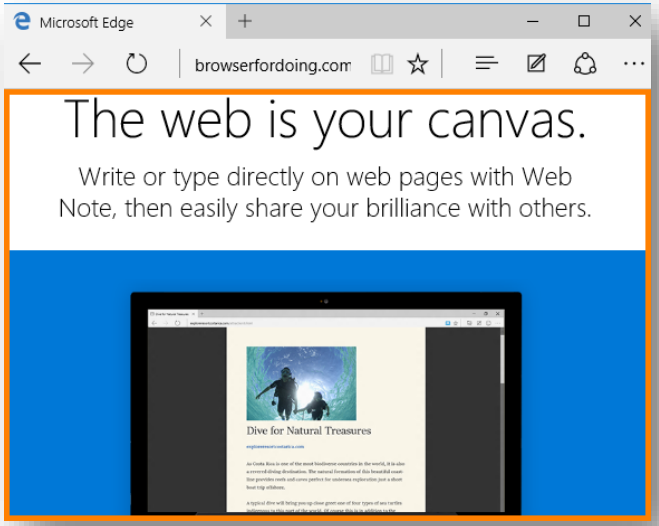
Understanding the Attack Surface and Attack Resilience of EdgeHTML

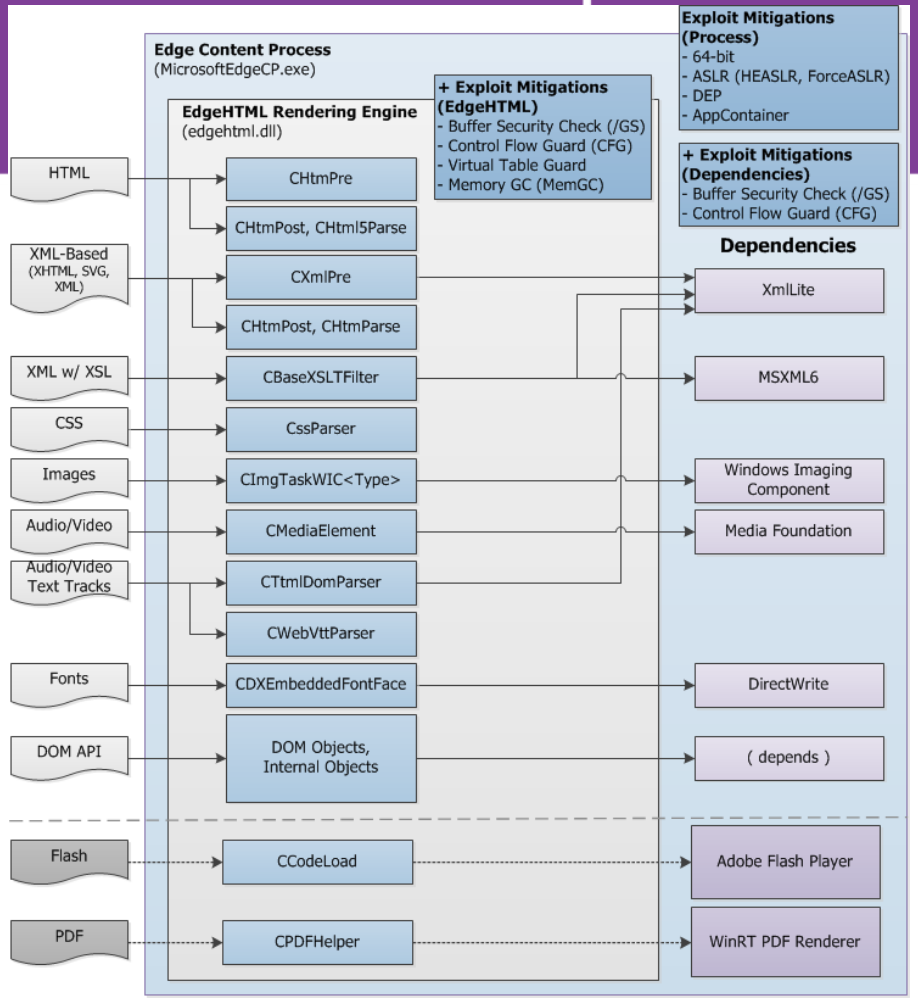
EdgeHTML Rendering Engine

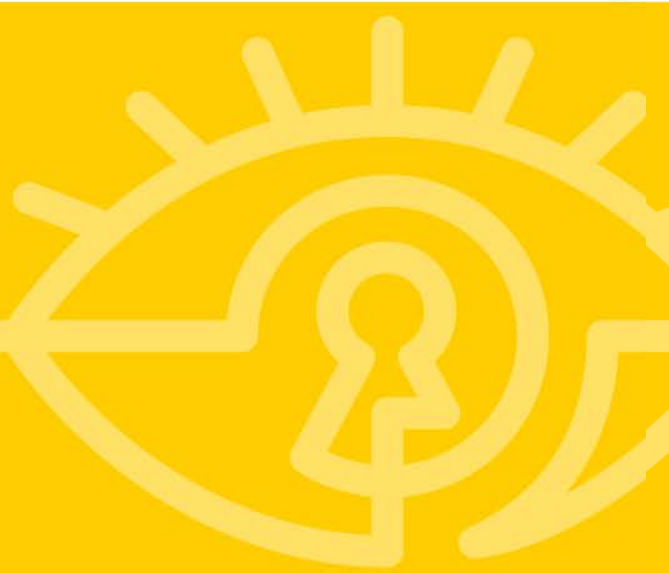


Overview

EdgeHTML Attack Surface Map & Exploit Mitigations







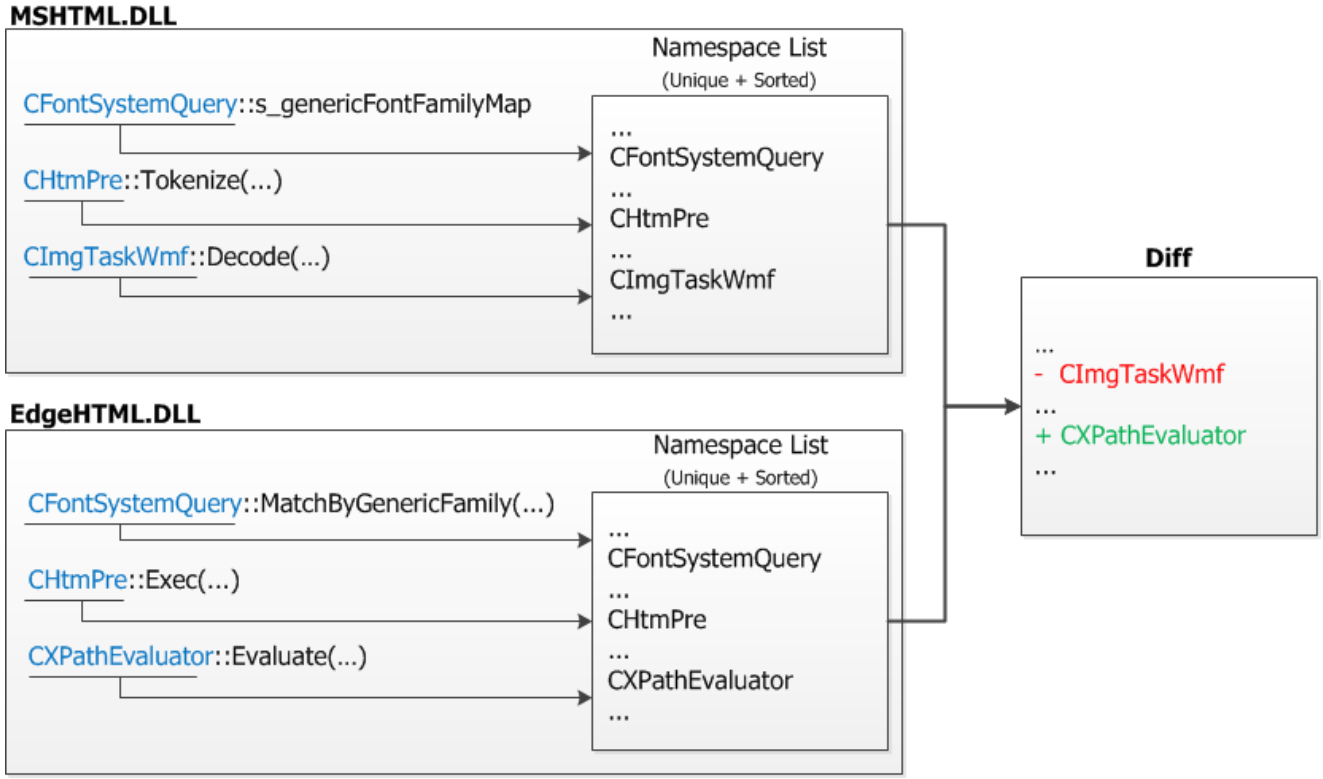
Initial Recon

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



Diffing MSHTML and EdgeHTML (Examples)

- Suggests change in image support:

```
-CImgTaskEmf
-CImgTaskWmf
```

- Suggests new DOM object types:

```
+CFastDOM::{…more…}
+CFastDOM::CXPathEvaluator
+CFastDOM::CXPathExpression
+CFastDOM::CXPathNSResolver
+CFastDOM::CXPathResult
+CFastDOM::CXSLTProcessor
```

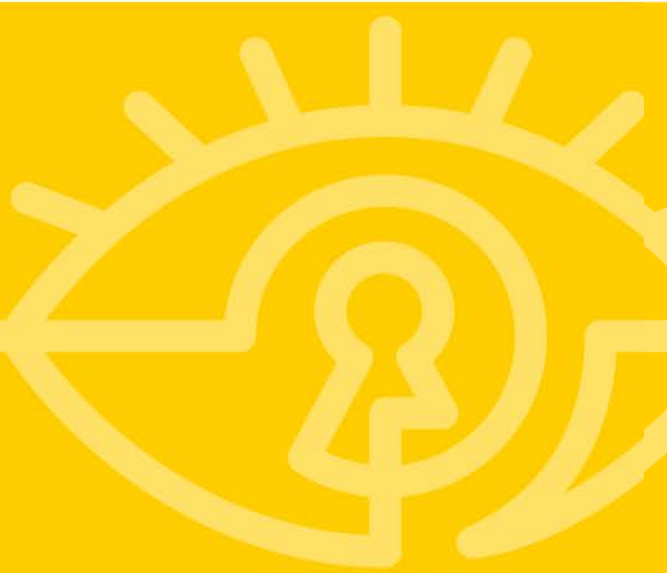
Diffing MSHTML and EdgeHTML (Examples)

- Suggests ported code from another rendering engine (Blink) for Web Audio support:

```
+blink::WebThread
+WebCore::AnalyserNode
+WebCore::AudioArray<float>
+WebCore::AudioBasicInspectorNode
+WebCore::Audio{…more…}
```

Diffing MSHTML and EdgeHTML (Notes)

- Further analysis needed
  - Renamed class/namespace results into a new namespace plus a deleted namespace
- Requires availability of symbols
  - Bindiffing is another option
- Same rudimentary diffing method can be applied to:
  - Function and method names
  - Strings
  - Imports and exports



Attack Surface

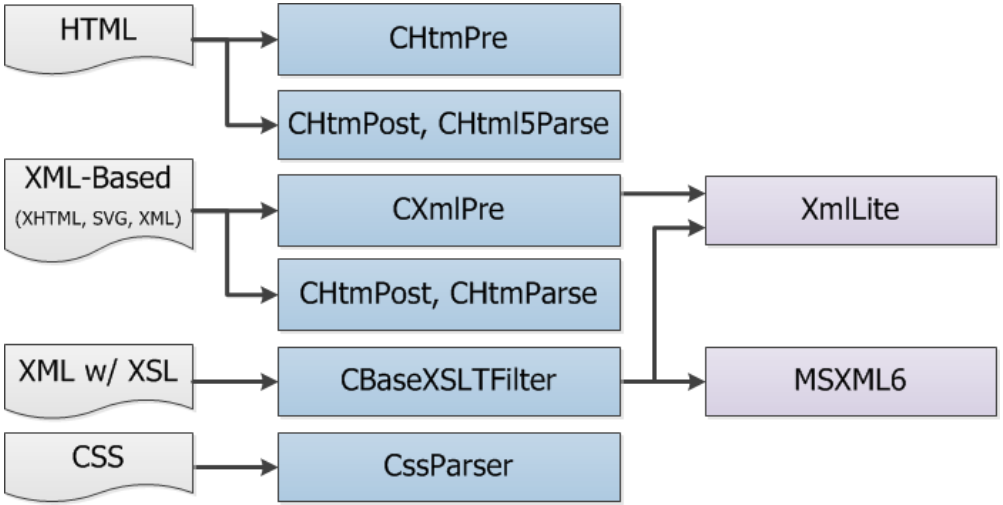
Understanding the Attack Surface and Attack Resilience of EdgeHTML

Attack Surface

- 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::**

Markup/Style Parsing



- 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

Markup/Style Parsing: XmlLite

- Lightweight XML parser
- Built-in Windows component
- *IXmlReader* interface is used by EdgeHTML for reading nodes from XML-based markups

Markup/Style Parsing: MSXML6

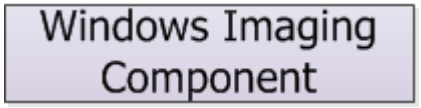
MSXML6

- Comprehensive XML parser
- Built-in Windows component
- *IXMLDOMDocument* interface is used by EdgeHTML for transforming XML that references an XSL stylesheet

Image Decoding

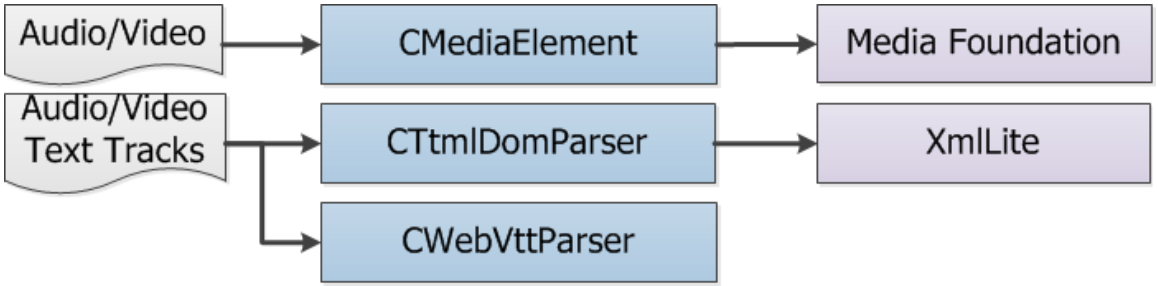
- Reachable via: direct link, *<img>*, *<embed>*
- Supported image formats: *g_rgMimeInfoImg*
- PNG, JPG, GIF, DDS, TIFF, BMP, HDP, ICO decoding via Windows Imaging Component (WIC)
- WMF and EMF support via GDI was removed in EdgeHTML

Image Decoding:
Windows Imaging Component (WIC)



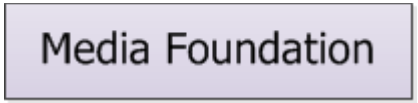
- Image decoder/encoder for multiple image formats
- Built-in Windows component
- *IWICImagingFactory::CreateDecoder()* is used by EdgeHTML to instantiate the decoder for a particular image format

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

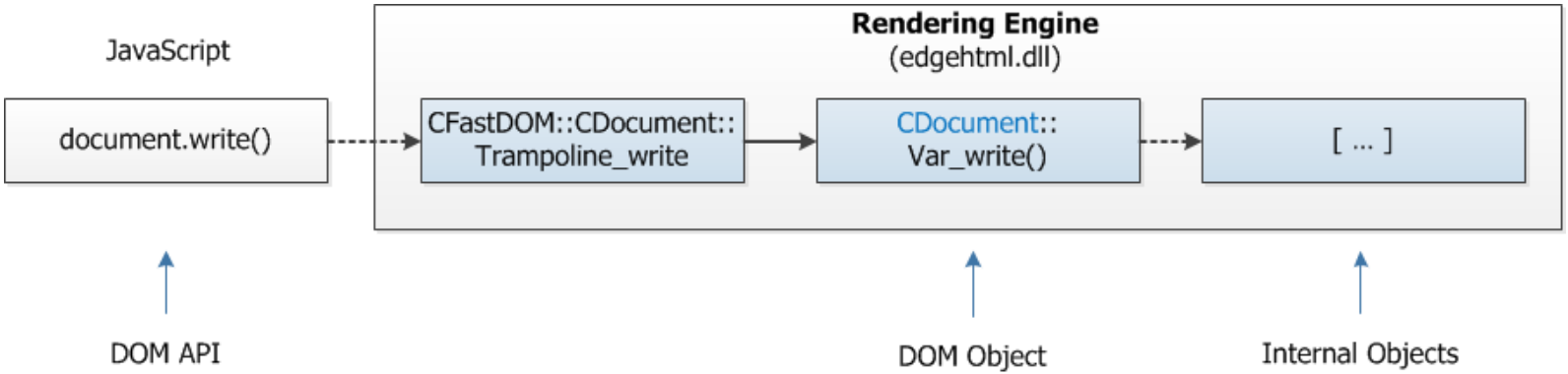
DirectWrite

- 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

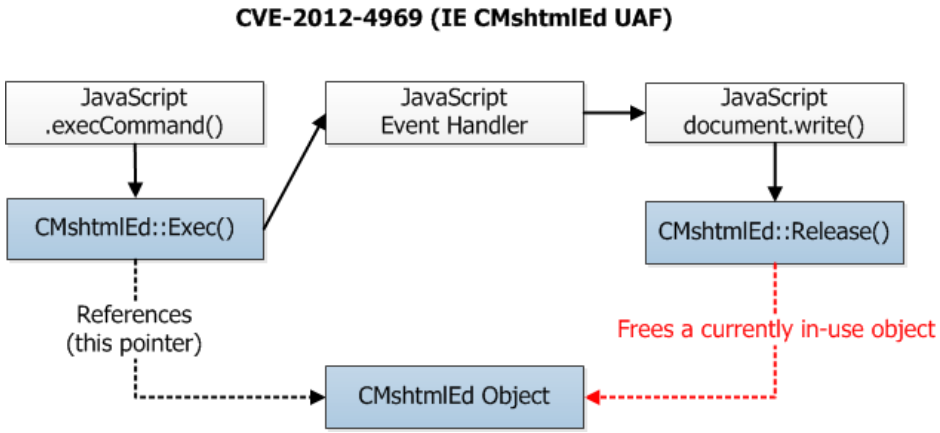
- Reachable via: JavaScript
- Large attack surface that:
  - Interacts directly with EdgeHTML DOM objects
  - Interacts indirectly with internal EdgeHTML objects and libraries (depends)

DOM API: Interaction with EdgeHTML DOM Objects and Internal Objects



- DOM API calls can change the state of the DOM tree, DOM objects and other internal EdgeHTML objects

DOM API: Vulnerabilities



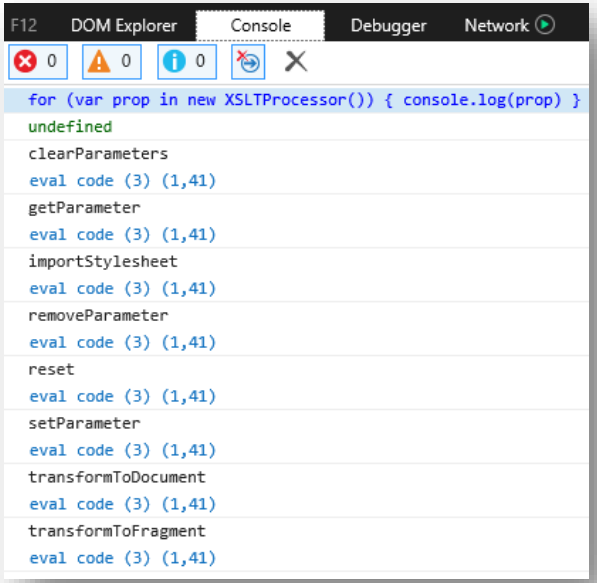
- Unexpected input, unexpected state changes or incorrect state when a DOM API is called can result to memory corruption such as: use-after-frees (above), heap overflows, invalid pointer access, etc.

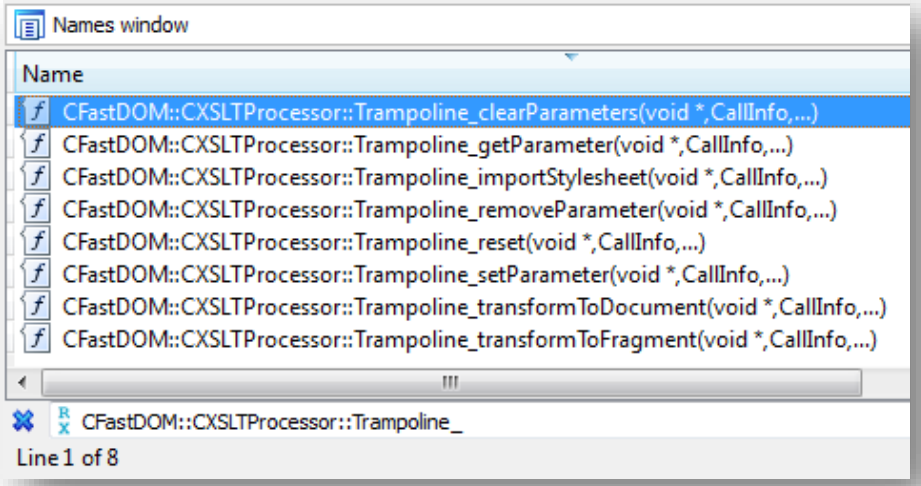
DOM API: New DOM Object Types

```
+CFastDOM::{…more…}
+CFastDOM::CVideoTrack
+CFastDOM::CVideoTrackList
+CFastDOM::CWaveShaperNode
+CFastDOM::CXMLHttpRequestUpload
+CFastDOM::CXPathEvaluator
+CFastDOM::CXPathExpression
+CFastDOM::CXPathNSResolver
+CFastDOM::CXPathResult
+CFastDOM::CXSLTProcessor
```

- 80 new DOM object types were found in EdgeHTML (GA build)
  - New code or new code paths that are reachable

DOM API: DOM Object Properties/Methods Enumeration





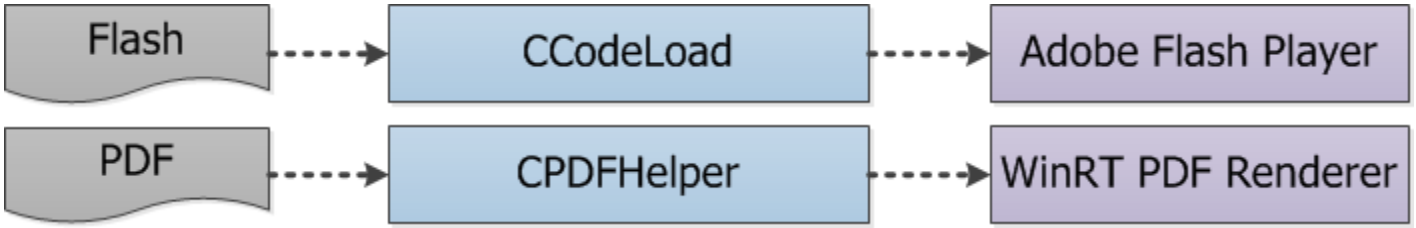
- Enumerating DOM object properties/methods via JavaScript and IDA…

DOM API: Diffing DOM Object Properties and Methods


```
 {…more…}
+document.evaluate
 document.execCommand
 document.execCommandShowHelp
+document.exitFullscreen
 document.fgColor
-document.fileCreatedDate
 {…more…}
```

- … 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

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

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

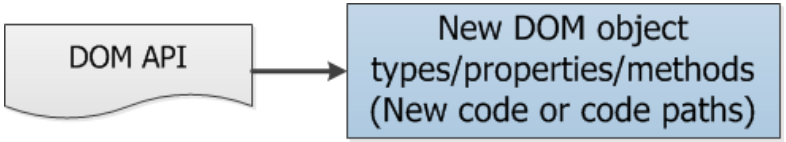
WinRT PDF Renderer

- 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

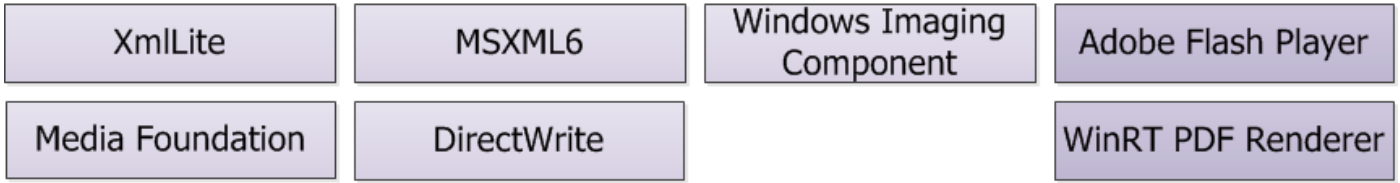
Attack Surface Summary

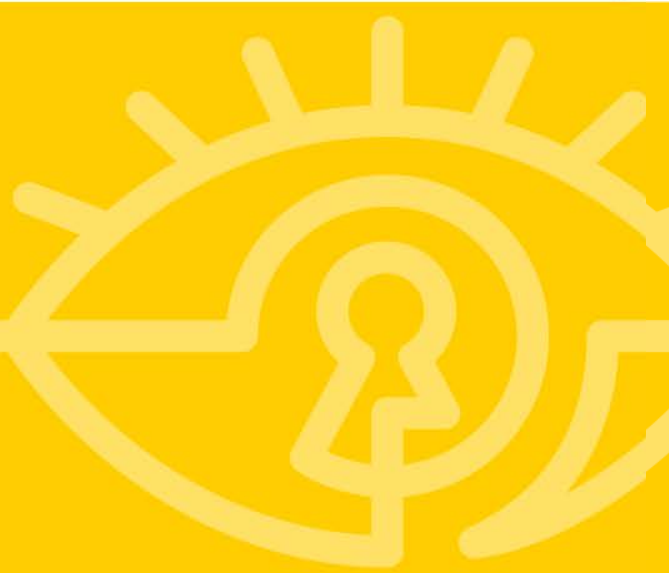
- Well-known attack vectors were removed

- New attack vectors were found in the DOM API



- Remotely-reachable libraries via EdgeHTML





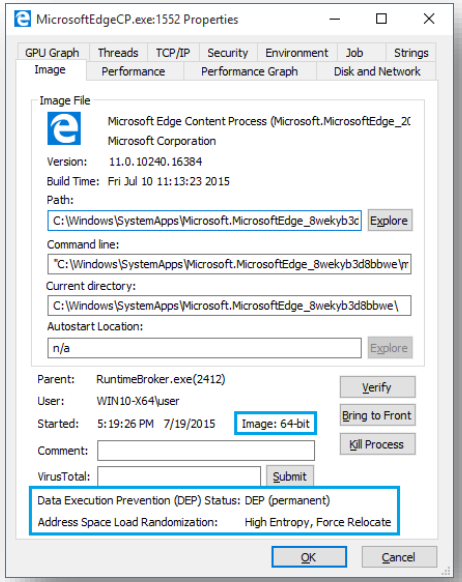
Exploit Mitigations

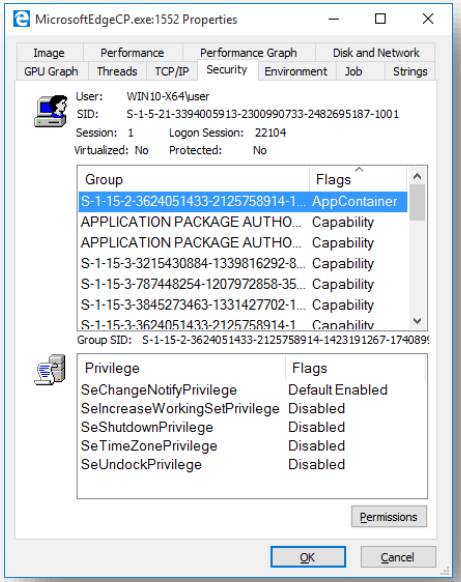
Understanding the Attack Surface and Attack Resilience of EdgeHTML

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]

Edge Content Process Mitigations





- MicrosoftEdgeCP.exe: 64-bit, ASLR (HEASLR, ForceASLR), DEP, and AppContainer

Edge Content Process Mitigations: Comparison with IE11 and ImmersiveIE

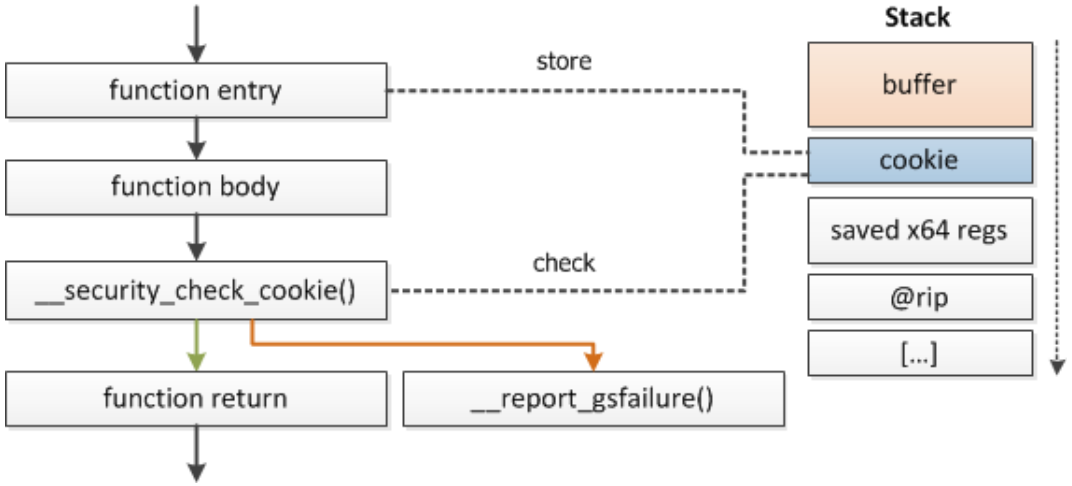
	Win10/ Edge	Win10/ IE11	Win8/ ImmersiveIE	Win8/ IE11	Win7/ IE11
64-bit	Yes	No	Yes	No	No
ASLR	Yes (HEASLR, ForceASLR)	Yes (ForceASLR)	Yes (HEASLR, ForceASLR)	Yes (ForceASLR)	Yes (ForceASLR)
DEP	Yes	Yes	Yes	Yes	Yes
Process Isolation	AppContainer	Low Integrity	AppContainer	Low Integrity	Low Integrity

- Comprehensive exploit mitigations are applied to the Edge content process (MicrosoftEdgeCP.exe) that hosts EdgeHTML (edgehtml.dll)

Edge Content Process Mitigations: Known Bypass/Weakness

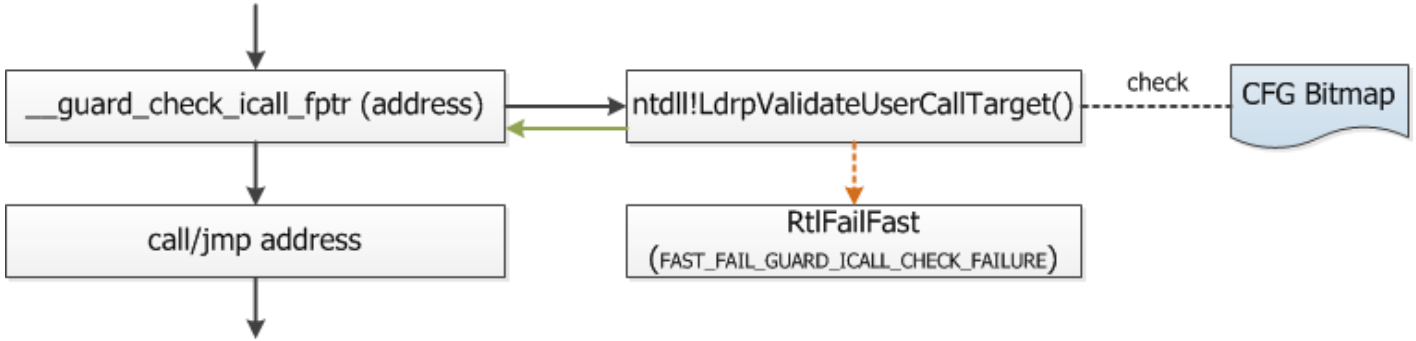
- 64-bit
  - Relative heap spraying (depends) [4,5]
- ASLR+DEP
  - Memory content disclosure (via vulnerabilities) [3,6]
- AppContainer
  - Kernel vulnerabilities [7,8]
  - Vulnerabilities in the broker or higher-privileged processes [9,10,11]
  - Leveraging writable resources [9]

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)

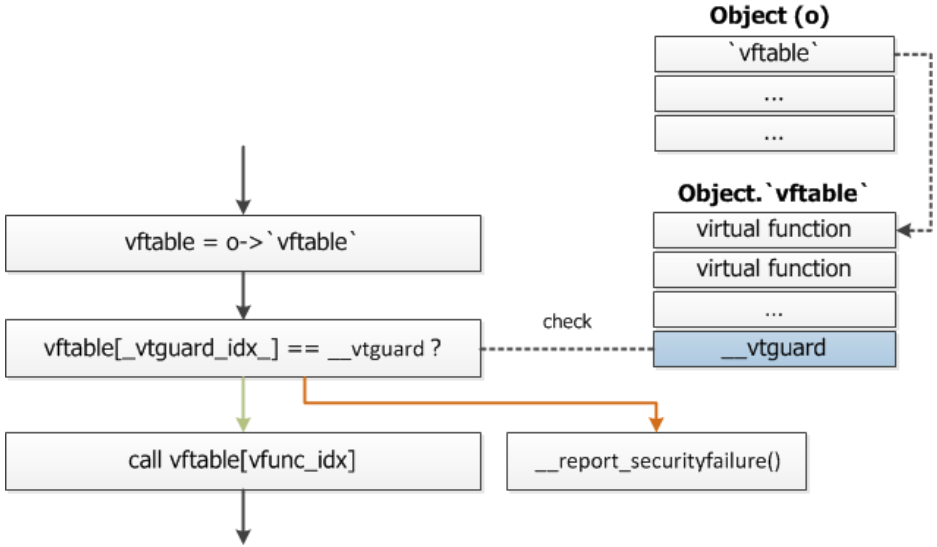


- Purpose: Detect and prevent abnormal control flow
- Recently introduced and well-researched [13,14]
- Several weaknesses and bypass techniques had been discovered (and mitigated) since its introduction

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)

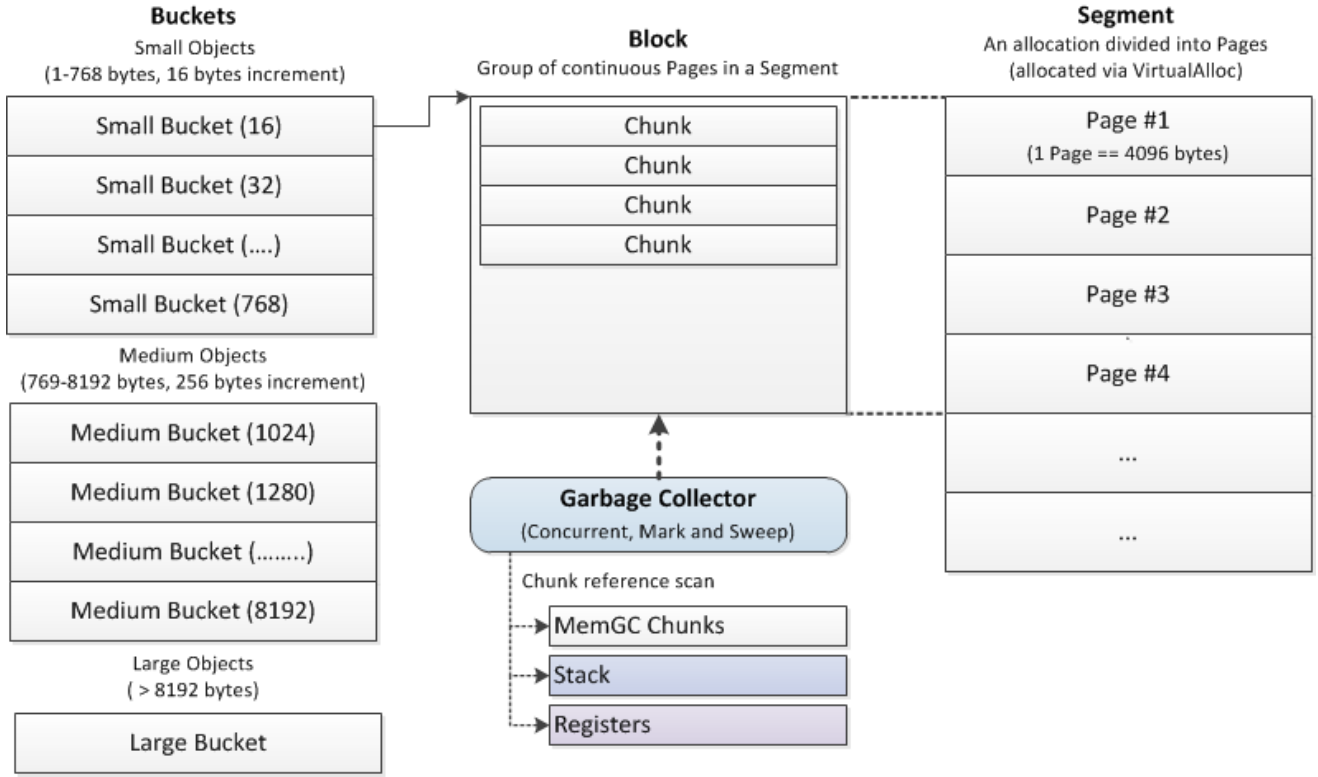


- Purpose: Detect an invalid virtual function table
- Known bypass/weakness: Applied only to select EdgeHTML classes and bypassed if address of *__vtguard* is leaked

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



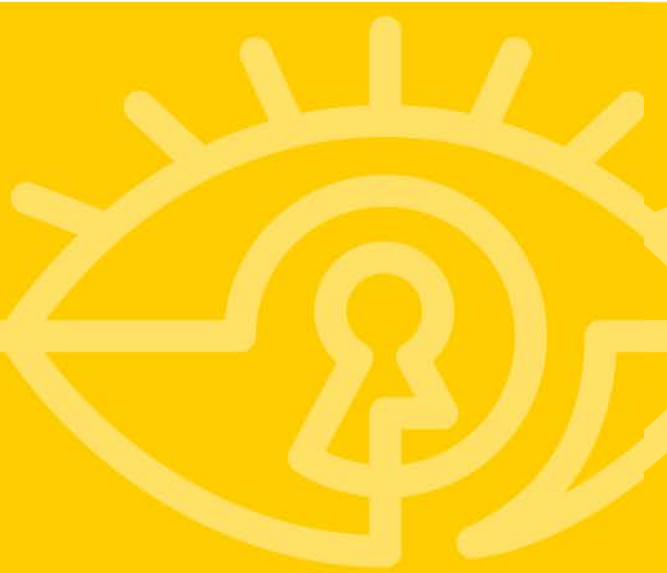
EdgeHTML Mitigations: Known MemGC Bypass/Weakness

- 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

Exploit Mitigations (Process)	+ Exploit Mitigations (EdgeHTML)	+ Exploit Mitigations (Dependencies)
- 64-bit - ASLR (HEASLR, ForceASLR) - DEP - AppContainer	- Buffer Security Check (/GS) - Control Flow Guard (CFG) - Virtual Table Guard (VTGuard) - Memory GC (MemGC)	- Buffer Security Check (/GS) - Control Flow Guard (CFG)

- 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

Conclusion

- 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):

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

Whitepaper

- Link of the detailed whitepaper is available at the end of the following blog post:
  - https://securityintelligence.com/memgc-use-after-free-exploit-mitigation-in-edge-and-ie-on-windows-10/
- 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)

- [1] M. Jurczyk, "**One font vulnerability to rule them all**," [Online]. Available: http://j00ru.vexillium.org/dump/recon2015.pdf
- [2] F. Falcón, "**Exploiting CVE-2015-0311, Part II: Bypassing Control Flow Guard on Windows 8.1 Update 3**," [Online]. Available: https://blog.coresecurity.com/2015/03/25/exploiting-cve-2015-0311-part-ii-bypassing-control-flow-guard-on-windows-8-1-update-3/
- [3] H. Li , "**Smashing the Heap with Vector: Advanced Exploitation Technique in Recent Flash Zero-day Attack**," [Online]. Available: https://sites.google.com/site/zerodayresearch/smashing_the_heap_with_vector_Li.pdf
- [4] I. Fratric, "**Exploiting Internet Explorer 11 64-bit on Windows 8.1 Preview**," [Online]. Available: http://ifsec.blogspot.com/2013/11/exploiting-internet-explorer-11-64-bit.html
- [5] Y. Chen, "**The Birth of a Complete IE11 Exploit Under the New Exploit Mitigations**"
- [6] F. Serna, "**The info leak era on software exploitation**," [Online]. Available: https://media.blackhat.com/bh-us-12/Briefings/Serna/BH_US_12_Serna_Leak_Era_Slides.pdf

References

(More are in the whitepaper)

- [7] T. Ormandy and J. Tinnes, "**There's a party at ring0 and you're invited**," [Online]. Available: https://www.cr0.org/paper/to-jt-party-at-ring0.pdf
- [8] Nils and J. Butler, "**MWR Labs Pwn2Own 2013 Write-up - Kernel Exploit**," [Online]. Available: https://labs.mwrinfosecurity.com/blog/mwr-labs-pwn2own-2013-write-up-kernel-exploit/
- [9] J. Forshaw, "**Digging for Sandbox Escapes - Finding sandbox breakouts in Internet Explorer**," [Online]. Available: https://www.blackhat.com/docs/us-14/materials/us-14-Forshaw-Digging-For_IE11-Sandbox-Escapes.pdf
- [10] M. V. Yason, "**Diving Into IE10's Enhanced Protected Mode Sandbox**," [Online]. Available: https://www.blackhat.com/docs/asia-14/materials/Yason/WP-Asia-14-Yason-Diving-Into-IE10s-Enhanced-Protected-Mode-Sandbox.pdf
- [11] P. Sabanal and M. V. Yason, "**Digging Deep Into The Flash Sandboxes**," [Online]. Available: https://media.blackhat.com/bh-us-12/Briefings/Sabanal/BH_US_12_Sabanal_Digging_Deep_WP.pdf
- [12] C. Evans, "**What is a "good" memory corruption vulnerability?**," [Online]. Available: http://googleprojectzero.blogspot.com/2015/06/what-is-good-memory-corruption.html

References

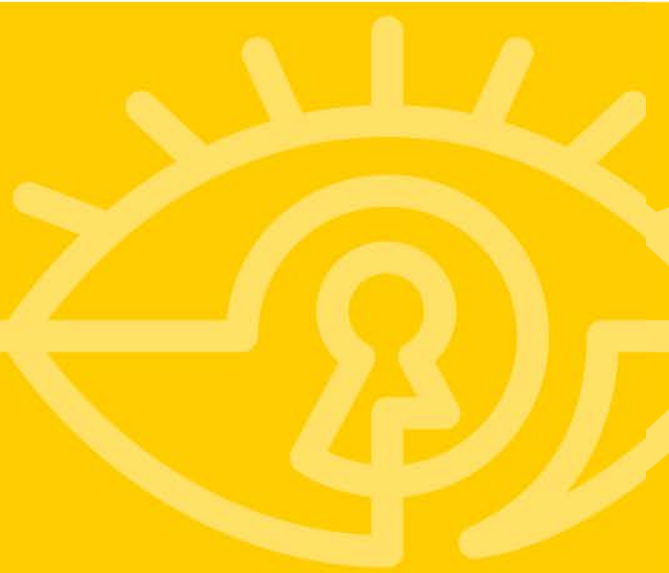
(More are in the whitepaper)

- [13] MJ0011, "**Windows 10 Control Flow Guard Internals**," [Online]. Available: http://powerofcommunity.net/poc2014/mj0011.pdf
- [14] J. Tang, "**Exploring Control Flow Guard in Windows 10**," [Online]. Available: http://sjc1-te-ftp.trendmicro.com/assets/wp/exploring-control-flow-guard-in-windows10.pdf
- [15] Y. Zhang, "**Bypass Control Flow Guard Comprehensively**," [Online]. Available: https://www.blackhat.com/docs/us-15/materials/us-15-Zhang-Bypass-Control-Flow-Guard-Comprehensively.pdf
- [16] tombkeeper, "**Bypass DEP and CFG using JIT compiler in Chakra engine**," [Online]. Available: http://xlab.tencent.com/en/2015/12/09/bypass-dep-and-cfg-using-jit-compiler-in-charkra-engine/
- [17] exp-sky, "**Use Chakra engine again to bypass CFG**," [Online]. Available: http://xlab.tencent.com/en/2016/01/04/use-chakra-engine-again-to-bypass-cfg/

References

(More are in the whitepaper)

- [18] S. Fleming and R. van Eeden, "**Triaging the exploitability of IE/EDGE crashes**," [Online]. Available: http://blogs.technet.com/b/srd/archive/2016/01/12/triaging-the-exploitability-of-ie-edge-crashes.aspx
- [19] M. V. Yason, "**Understanding IE's New Exploit Mitigations: The Memory Protector and the Isolated Heap**," [Online]. Available: http://securityintelligence.com/understanding-ies-new-exploit-mitigations-the-memory-protector-and-the-isolated-heap
- [20] H. Li, "**Microsoft Edge MemGC Internals**," [Online]. Available: https://github.com/zenhumany/hitcon2015



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