2 浏览器攻击面

作者：Ox9A82 最后修改：2018-3-15 发布时间：2018-3-15

浏览器攻击面

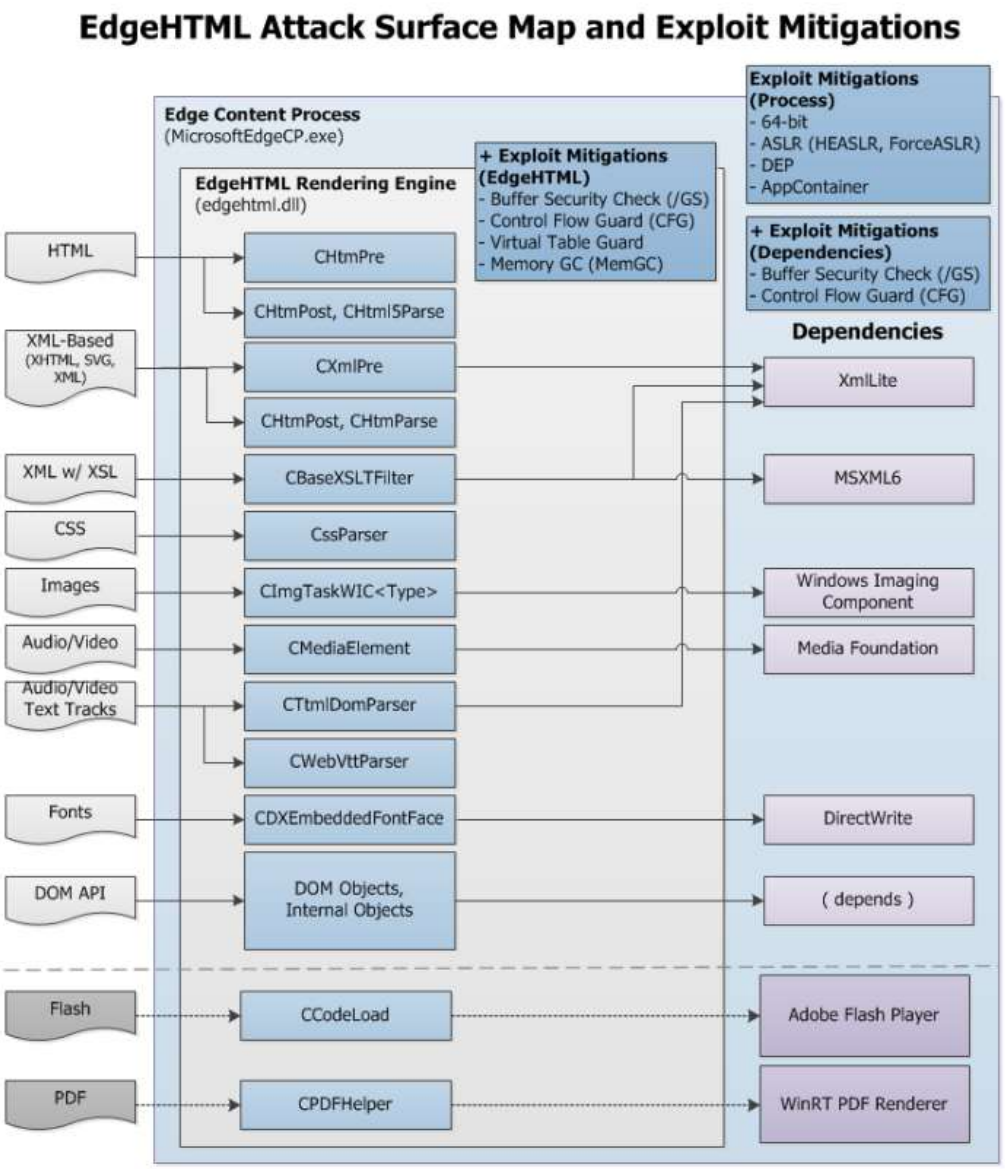
大家可能接触过一些关于漏洞利用的CTF题目或者亲自调试过一些网上搜集来的漏洞样本，但我认为做安全研究与在网上找一些漏洞样本拿来调试是有本质区别的。首先安全研究研究的是整个目标系统的安全问题，而不只是为了利用指定的某个漏洞点。所以当我们拿到一个未知的目标要进行安全研究之前，我认为首先应该要做的是搜集并整理出这个目标程序的攻击面。所以这里我选择在第二篇首先给大家介绍一下关于浏览器的攻击面。

提起浏览器我们可能下意识的觉得这并不是一个复杂的程序，因为浏览器对于计算机用户来说实在再熟悉不过了。我们每天都在使用浏览器浏览网页、观看视频等等，我们可能会认为浏览器只一个用来浏览器网页的应用程序，从而忽略了这个程序本身的复杂性。

但是如果我们仔细的进行分析，就会发现浏览器的功能其实相当复杂。比如我们日常浏览的页面不可能只是由简单的HTML构成的，尤其是随着Web前端技术的发展其中会充斥着大量的Javascript等脚本语言，那么浏览器就需要具有对脚本进行解析执行的能力因此需要具有脚本引擎。比如我们浏览的网页中不会只有纯文本，而是会存在着大量的图片，甚至这些图片的格式也各不相同，那么这些图片就需要正确的被解析以显示给用户。此外我们在使用浏览器观看视频时经常会用到的Adobe Flash说明浏览器还需要支持一些插件扩展，比如ActiveX等来满足用户的额外需求。

一般来说一个系统提供的功能越多，暴露的攻击面也就越大。在《白帽子讲浏览器安全》一书中，作者提出:浏览器安全其实是涵盖了客户端、web、server安全等的一个综合体，不过在本文中我们还是把重点放在能够造成内存破坏漏洞的攻击面上。这篇文章我参考了X41在去年发布的白皮书和Black hat15上一个讲述Edge浏览器攻击面的议题，也相当于是我个人做的学习笔记给大家分享。

这张图片展示了Edge浏览器支持的一些常见技术以及这些技术在Edge的渲染引擎中是被什么类进行处理的或者是依赖了什么外部库进行处理的。



可以看到像我们前面所说的那样，浏览器支持了相当多的技术，这里我摘录分类为六类攻击面。分别是渲染引擎主要负责的各类Web技术、在网页中不可获取的各类图片格式、曾被用于进行内核提权实现突破浏览器沙箱的自定义字体文件、以及已被实际CVE证明可行的PDF解析和最后两个大家最熟悉的攻击面DOM接口和脚本解析引擎。

Web技术

第一部分是浏览器支持的Web技术，如HTML、CSS、XML等，这些是渲染引擎主要负责的工作之一。

Markup/Style	EdgeHTML Class	Library (and Interface) Used
HTML	CHtmPre (Pre-parsing) CHtmPost, CHtml5Parse (Post-parsing)	
XML-Based (XHTML, SVG, XML)	CXmlPre (Pre-parsing) CHtmPost, CHtmParse (Post-parsing)	XmlLite (IXmlReader)
CSS	CssParser	
XML w/ XSL	CBaseXSLTFilter	XmlLite (IXmlReader) MSXML6 (IXMLDOMDocument)
VML	(Removed in EdgeHTML: No Binary Behaviors)	

在历史上HTML、CSS、XML都被实际爆出过内存破坏类漏洞。

比如HTML漏洞:CVE-2004-1050，这个漏洞今天看起来可能会有点"弱智"，只是在IFRAME标签的NAME属性里面赋一个很长的字符串就引发了溢出，虽然现在哪怕是最基础的CTF中也不屑于出现这种题目了，但是一份04年的exp中使用了现在大家依然喜闻乐见的堆喷技术，历史就是这么发展的不是吗:D
比如XML漏洞:CVE-2008-4844
比如CSS漏洞:CVE-2007-1750、CVE-2014-2764

Blackhat议题给出了Edge处理HTML的路径，Edge浏览器以CHTML*类对HTML进行解析
Edge会执行CHtmPre::Exec()->对标签进行预解析->进行初始解析->将解析的标签写入标签流->执行CHtmPost::Exec()->在标签流中取标签->进行进一步解析->创建DOM对象。
而XML在Edge中，由XmlLite和MSXML6这两个外部库负责处理。

Browser	CSS2.1	Frames	XSLT	XHTML 1.0	XHTML 1.1	Web Forms 2.0	SMIL	VML
Google Chrome	●	●	●	●	●	○	○	○
Microsoft Edge	●	●	●	●	●	●	○	○
Internet Explorer	●	●	●	●	●	○	●	●

Table 4.1: Web Technologies supported by Browsers (● - True, ○ - False, ◐ - Partly)

在目前主流浏览器支持的Web技术中可以看到只有IE浏览器还原生支持SMIL和VML。而在IE时代，VML解析也曾是一个漏洞高发地带，比如泉哥《漏洞战争》中出现过的CVE-2013-2551就是一个大家都有调试过的VML漏洞。

图像解析

如前所述Web页面中存在各种格式的图片文件，这些图片文件的解码也是要由浏览器负责的重要工作。

Image Format	EdgeHTML Class	Library (and Interface) Used
PNG	CImgTaskWICPng	WIC (IWICImagingFactory)
JPG	CImgTaskWICJpgTemplate	WIC (IWICImagingFactory)
GIF	CImgTaskWICGif	WIC (IWICImagingFactory)
DDS	CImgTaskWICDds	WIC (IWICImagingFactory)
TIFF	CImgTaskWICTiff	WIC (IWICImagingFactory)
BMP	CImgTaskWICBmp	WIC (IWICImagingFactory)
HDP	CImgTaskWICHdp	WIC (IWICImagingFactory)
ICO	CImgTaskWICIco	WIC (IWICImagingFactory)
WMF	(Removed in EdgeHTML: Processing Removed)	
EMF	(Removed in EdgeHTML: Processing Removed)	

而在Edge浏览器中，图片的解析是交给WIC来做的。WIC是微软提供的一套图像处理库，关于WIC的情况可以参见MSDN

Browser	JPEG	JPEG XR	WebP	GIF	PNG	APNG	TIFF	SVG	PDF	2D Canvas	XBM	BMP	ICO
Google Chrome	●	○	●	●	●	●	○	◐	○	●	●	●	●
Microsoft Edge	●	●	○	●	●	○	●	◐	○	●	○	●	●
Internet Explorer	●	●	○	●	●	○	●	◐	○	●	○	●	●

Table 4.5: Image Format Support (● - True, ○ - False, ◐ - Partly)

上图是浏览器对常见图片格式的支持情况。

字体解析

在浏览器中可以通过CSS来指定使用的字体，而负责处理字体的解析虚拟机一般都位于内核的win32k中。
在内核中执行代码这一点被攻击者看中，从而在历史中出现了通过在浏览器中触发字体漏洞获得内核权限实现绕过沙箱的利用。

Font Format	EdgeHTML Class	Library (and Interface) Used
TTF	CDXEmbeddedFontFace	DirectWrite (IDWriteFactory1, IDWriteFactory2)
OTF	CDXEmbeddedFontFace	DirectWrite (IDWriteFactory1, IDWriteFactory2)
WOFF	CDXEmbeddedFontFace	DirectWrite (IDWriteFactory1, IDWriteFactory2) (after extraction of TTF/OTF via CDXEmbeddedFontFace::UnpackFontFromWOFFData())
EOT	(Removed from EdgeHTML: Processing Removed)	

但是在Win10创意者更新之后，Windows的字体渲染已经全面使用DirectWrite。字体解析由内核移到用户层处理，通过字体漏洞实现穿透沙箱的日子可能一去不返了。

PDF

另外一点是关于PDF解析，我们在日常浏览网页时会发现PDF是可以直接在浏览器中打开的。因为Chrome、Edge等浏览器是原生支持PDF格式的，如果是IE浏览器则需要使用Adobe插件才能解析。
在Edge浏览器中渲染PDF的情况如下

Content Type	EdgeHTML Loader/Helper Class	Built-in/Pre-installed Renderer
PDF	CPDFHelper	Built-in WinRT PDF Renderer in Windows (%System32%\Windows.Data.Pdf.dll)

可以看到Edge是调用WinRT PDF Renderer的dll进行的PDF渲染，从而PDF也成为一个重要的攻击面。

比如CVE-2016-3203、CVE-2016-3370就是挖掘的WinRT PDF Renderer的漏洞，从而实现了从PDF影响Edge浏览器。

DOM接口

对于各种浏览器的历史漏洞来说，与DOM有关的漏洞是最多的并且直到今天也层出不穷。因此DOM接口一直是各大浏览器的一个最重要的攻击面，本系列文章也是围绕着DOM漏洞进行展开的。

当HTML文档被解析后，渲染引擎会把HTML标签实例化为DOM对象，但是DOM对象在实例化之后不是一成不变的，可以通过DOM API进行动态改动。通过js调用的DOM API最后会执行到相应的DOM函数进行各种操作。

Browser	JavaScript	ECMAScript 3	DOM 1	DOM 2	DOM 3	XPath	DHTML	XMLHttpRequest	Rich editing
Google Chrome	●	●	●	●	◐	●	●	●	●
Microsoft Edge	●	●	●	●	◐	●	●	●	●
Internet Explorer	●	●	●	●	●	●	●	●	●

Table 4.4: JavaScript Support (● - True, ○ - False, ◐ - Partly)

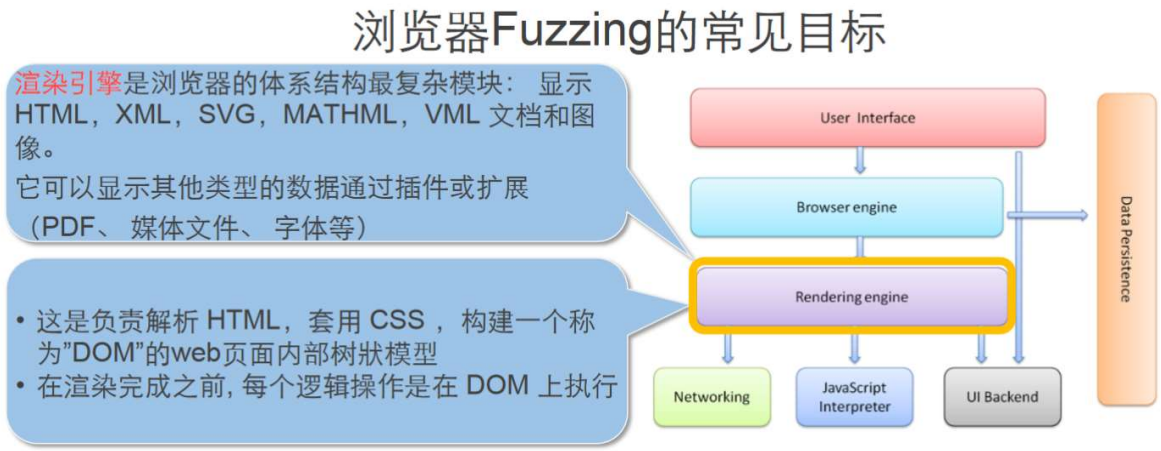
DOM标准由W3C维护，DOM并不是一个固定不变的标准，实际上目前有DOM1、DOM2、DOM3、DOM4四个标准，但目前只实现了DOM3，DOM4还是草稿状态。随着DOM Level3更多特性的实现，更多的攻击面也随之暴露出来。比如DOM LEVEL3中添加了更多的事件，这些事件在Fuzz过程中也被发现了更多的漏洞。

脚本引擎

浏览器支持Javascript等脚本来支撑页面的动态功能，所以浏览器需要内置js解析引擎来执行这些脚本代码。在早期的IE浏览器中，负责解析JS的库是jscript.dll，在IE9之后改为jscript9.dll，对于最新的Edge浏览器中这个库是Chakra。

内置Javascript本来不算什么太大的问题，但是Javascript本身语法的松散型导致脚本引擎虚拟机在解析执行Javascript时很容易出现各种问题，比如各种类型混淆或越界读写等。尤其是js本身也是一个不断更新不断添加新特性的语言，比如ECMASCRIPT6中的Proxy特性就导致了很多问题。
如今在浏览器厂商对DOM漏洞的防护日益加强的情况下(比如2014年加入的隔离堆、延迟释放等)，浏览器中的Javascript虚拟机已经成为了一个新的重要的漏洞攻击面，在本系列中js引擎是另一个展开描述的重点。

最后以一张图来总结，谢谢大家的阅读。



Reference

https://github.com/x41sec/browser-security-whitepaper-2017

https://www.blackhat.com/docs/us-15/materials/us-15-Yason-Understanding-The-Attack-Surface-And-Attack-Resilience-Of-Project-Spartans-New-EdgeHTML-Rendering-Engine-wp.pdf