登录

WebAssembly黑暗的一面（上）

mss**** / 2018-10-20 20:01:38 / 浏览数 2587 技术文章 技术文章 顶(2) 踩(0)

原文：https://www.virusbulletin.com/virusbulletin/2018/10/dark-side-webassembly/

摘要

WebAssembly（Wasm）格式是一种在Web浏览器中运行C/C++等本机语言编译而成的代码的方法。与其他诸如JavaScript之类的编程语言编译得到的字节码相比，由本机
Edge在内的所有流行浏览器的最新版本都已经对WebAssembly提供了相应的支持。

虽然Wasm已经问世多年，但直到最近被用于浏览器环境下的加密货币挖掘，它才开始崭露头角。同时，这也打开了Wasm恶意应用的潘多拉盒子。

在本文中，我们将研究Wasm恶意应用的一些具体例子，例如：

- 技术支持诈骗：随着漏洞利用工具包的减少，技术支持骗局正在通过各种渠道喷薄而出，包括被攻陷的网站、恶意广告等。由于这些诈骗活动大量使用JavaScript代码，
- 浏览器漏洞利用：对于用JavaScript编写的浏览器漏洞利用代码，我们可以对其进行修改，让它们使用Wasm来利用浏览器漏洞，并完成后续的恶意软件下载。
- 基于脚本的键盘记录程序：Wasm还可用于窃取输入到Web表单中的信息。目前，这种信息窃取行为都是通过JavaScript代码完成的。

就目前来说，针对Wasm格式的恶意代码的检测还是非常困难的，因为这是一种编译型文件，使得之前基于字符串的检测方式几乎全部失效。下面，我们将讨论适用于上述方

简介

JavaScript

JavaScript[1]是一种通用编程语言。JavaScript是一种非常简单的语言，却有着庞大的生态系统，并且与Web密不可分。除非将现有的Web应用程序全部推倒重来，否则根

目前JavaScript的速度非常快，但是JavaScript引擎中的某些机制对其速度仍有所限制[2]:

- 装箱：引擎会对浮点数进行装箱处理，为其建立相应的封装器，使得它们可以与其他值（如对象）共存。
- 即时（JIT）编译和运行时类型检查：对于大多数JavaScript引擎来说，代码的编译工作是分两步完成的。首先，使用一种编译速度很快，但是运行速度很慢的格式进行编
- 自动垃圾收集：这会导致代码的运行速度大为降低。
- 灵活的内存布局：JavaScript的数据结构非常灵活，但也使得内存管理速度变慢。

Asm.js

Asm.js[3]是JavaScript语言的一个子集，定义该子集的目的是易于优化，主要用作C和C++等语言的编译器目标。Asm.js代码可以生成没有上述缺点的可执行文件。它们可以

由于Web不受任何单一供应商的控制，因此每一次改进大家都必须群策群力。asm.js出自于Mozilla的一群核心开发人员之手。同时，Google开发人员则致力于Native Client（NaCl）和Portable Native
Client（PNaCl），这是一种基于LLVM编译器项目的Web二进制格式。虽然这些解决方案都在某种程度上发挥了作用，但它们并没有为上述所有问题提供令人满意的答案。

WebAssembly是asm.js不断进化的结果
[4]。WebAssembly的目标是填补一个迄今为止JavaScript被迫扮演的角色:一种可以作为编译器目标的低级代码表示形式。

WebAssembly为C和C++等语言提供了统一的编译目标，而这些语言很难映射到JavaScript语言[5]。

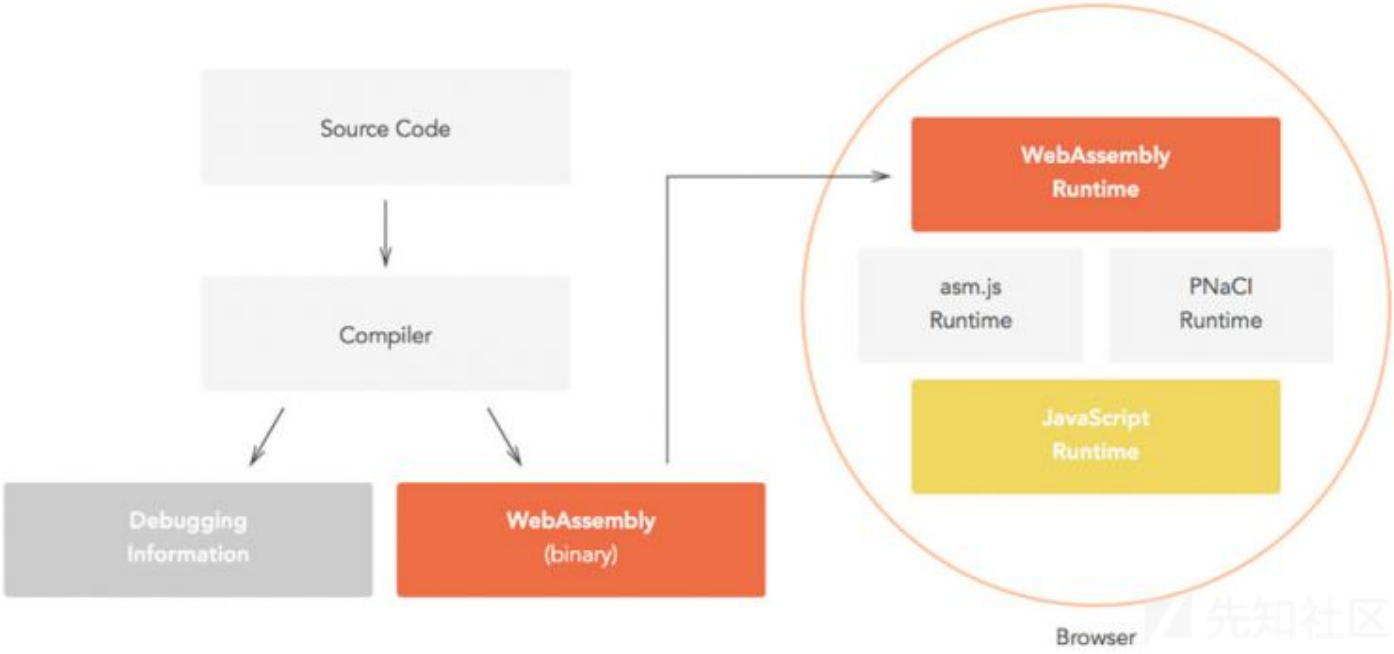
WebAssembly

WebAssembly
(Wasm)是一种可以在现代web浏览器上运行的新型代码，它不仅提供了新的特性，同时，性能也有了显著的提升。它被认为是web的一种新的二进制格式[6,7]。通常，对性

当然，创建Wasm的初衷并非为了替代JavaScript，而是为了实现两者之间的补充和配合。随着WebAssembly的引入，现代web浏览器的虚拟机将同时运行JavaScript和Wa

目前，所有主流浏览器都已经支持Wasm。WebAssembly代码的优点包括:

- 快速、高效和可移植性:WebAssembly的代码可以在不同的平台上以接近本机的速度执行
- 可读性和可调试性:WebAssembly是一种低级汇编语言，但它具有人类可读的文本格式
- 安全性: WebAssembly被指定在安全的沙箱执行环境中运行。



WebAssembly:旨在提供跨浏览器编译器目标的共同协作。

如何生成WebAssembly代码？

像Emscripten [8,9]这样的工具可以用来将用C/C++编写的代码编译成WebAssembly代码：

- 复制下面简单的C示例代码，并将其保存到本地驱动器的新目录下名为“hello.c”的文件中：

Sample Hello World

```c
#include <stdio.h>
int main(int argc, char ** argv) {
  printf("Hello World\n");
}
```

图2：将上面的C示例代码的副本保存到本地驱动器的新目录下名为“hello.c”的文件中。

- 导航到hello.c文件所在目录，然后运行以下命令：

```
emcc hello.c -s WASM=1 -o hello.html
```

对于上述命令中的各个选项，解释如下：

-s WASM=1 ——规定输出格式为Wasm。如果我们不指定的话，Emscripten将只输出asm.js，就像默认情况下那样。

-o hello.html
——通知Emscripten生成一个HTML页面，用于存放要运行的代码（以及要使用的文件名），以及相应的Wasm模块和JavaScript“胶水”代码（用来编译和实例化Wasm，使

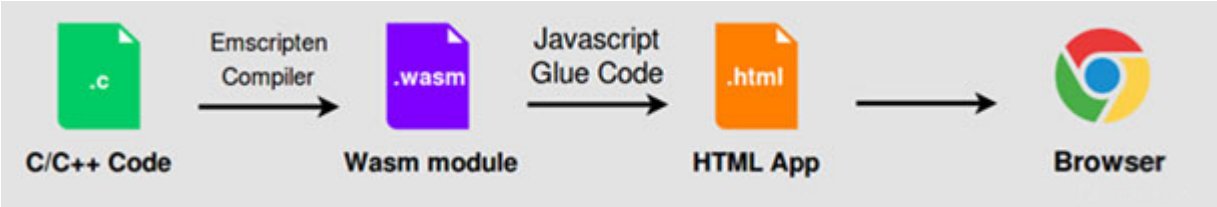


图3：将代码编译为WebAssembly格式。

我们将来的计划是摆脱上述JavaScript胶水代码，能够以类似`JavaScripts (<script type='module'>)`这样的形式来加载WebAssembly模块。

WebAssembly与恶意软件

我们知道，WebAssembly不仅提供了出色的性能，还提供了丰富的功能，这些特性早晚会引起恶意软件作者的注意。WebAssembly在基于浏览器的挖矿方面非常具有竞争

挖矿流程如图4所示。

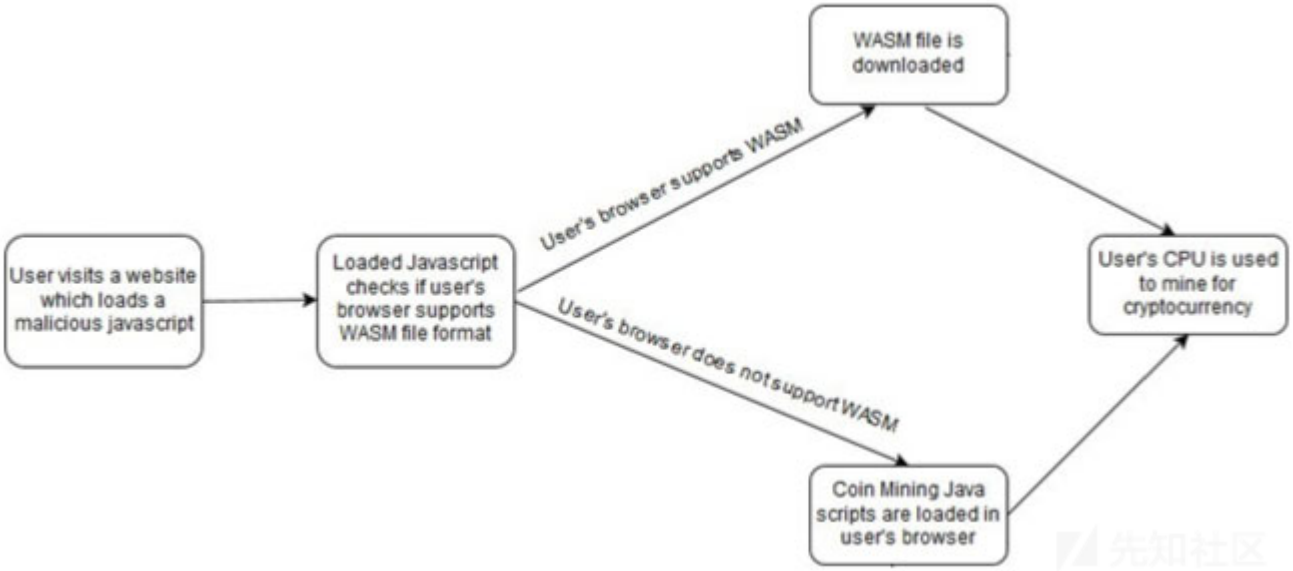


图4：挖矿流程

了解了上述技术后，下面将讨论恶意使用WebAssembly的其他方法。

小结

在本文中，我们为读者详细介绍了WebAssembly的基础知识，在下一篇文章中，我们将为读者介绍它在恶意软件方面的用途。

上一篇：区块链安全—白话parity多签名... 下一篇：Thinkphp-聚合查询漏洞

1. 0 条回复
   - 动动手指，沙发就是你的了！