最近老师想研究研究挖矿网站的工作，于是自己试着看了看相关的网站，记录部分结果如下：

首先了解一番挖矿究竟是啥，这里首先做个简单的比喻加深了解：

钱包——汇款地址：记录个人电子货币资产、操作交易、收发货币的一个hash地址；

矿池——工作单位：相关电子货币根据区块链统一分配“生产”任务的发布平台，也是算力效验的对接平台；

挖矿——努力工作：利用设备算力按特定电子货币生产计算方法和矿池分配方式产生的经济价值。

算力：计算机设备的工作能力。

共识机制——绩效评定方法：区块链系统中实现不同节点之间建立信任、获取权益的数学算法；

本文前端挖矿就是说“浏览器内实现、通过用户访问实施挖矿”的一种方式。

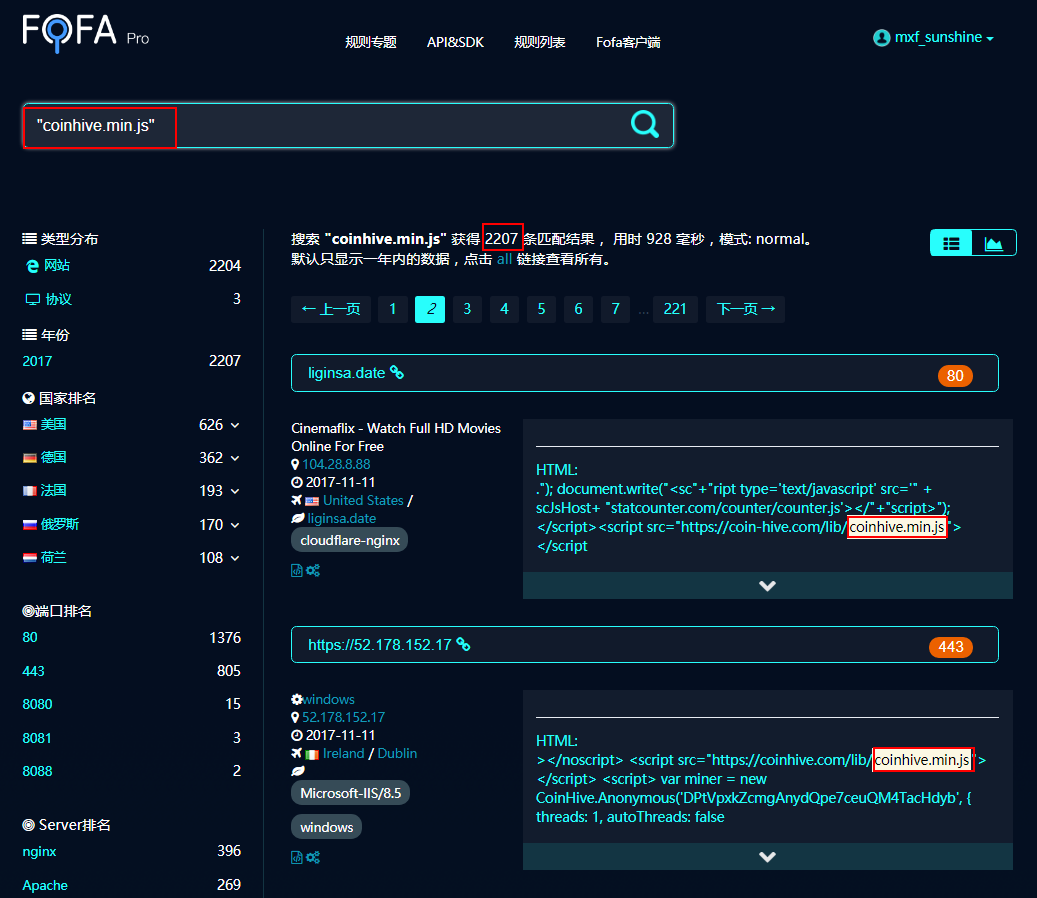
一般挖矿：用户以主动方式直接使用C或者其他语言构造的miner客户端进行CPU或GPU计算得到算力价值。

前端挖矿：用户以被动或主动方式在不知情或知情情况下由浏览者产生的CPU或GPU计算得到算力价值。

一般挖矿：利用CPU或GPU本身的高速浮点计算能力进行挖矿工作。而一般挖矿选取有更优秀浮点计算能力的GPU进行操作。从行为和实现方式看是一种主动在自己设备完成的挖矿行为，排除非法控制他人设备情况。c实现的工具直接使用OS的底层驱动和接口，无论操作内存、CPU或GPU流程损耗低计算速度更快。

前端挖矿：利用asm.js或webAssembly前端解析器中介在浏览器端被动使用用户的CPU完成挖矿或者利用Html5新规范WebGL利用浏览器完成GPU挖矿操作。从行为和实现方式看可以是一种以被动方式，在其他用户浏览到某网页情况下，在非己设备上发生的挖矿行为。通过将算力任务分担给其他浏览用户来实现“分布式”计算，其实更应该叫利用多设备优势提高单一时间内单目标多任务并发能力。

coinhive专门提供一个用来挖矿的JS引擎，在被攻击网站上的网页内嵌一段JS代码，只要有人访问被攻击网站，挖矿程序就会在网民的电脑上工作，占用大量的系统资源，导致CPU利用率突然提升。但是此JS只能用于挖取门罗币，在FOFA上查询此JS代码部署情况的结果如下：

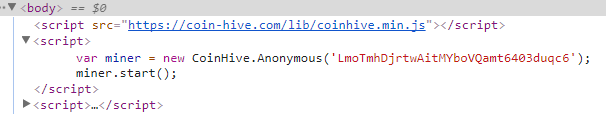


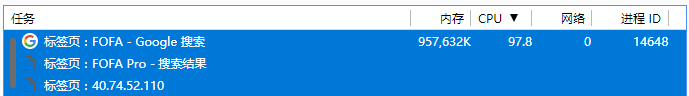
查看具体网站：

HTML中嵌入脚本代码，引用了<https://coinhive.com/lib/coinhive.min.js>

并嵌入script脚本启动挖矿代码。

网址：<http://40.74.52.110/>

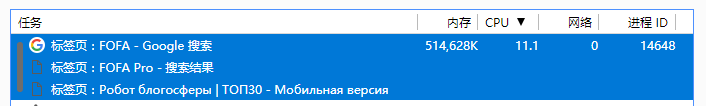




无限制时几乎占据所有CPU资源。

网址：<http://ya.topbloger.ru/page/t30p-mobile-application.aspx>



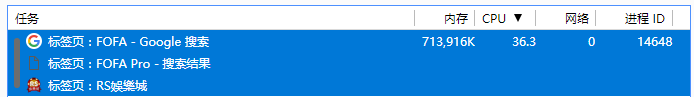


（此处之所以CPU占比不是90%左右，是因为线程数是1，本计算机CPU核数是8核，均分后CPU占比10%左右）

网址：<http://52.192.18.99:443/>

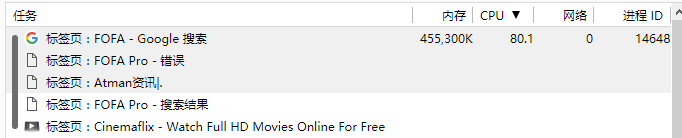
body中



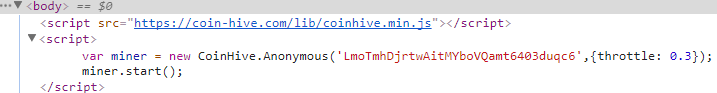


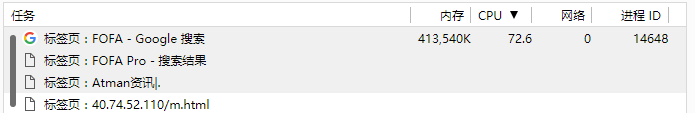
网址：http://cinemaflix.net/





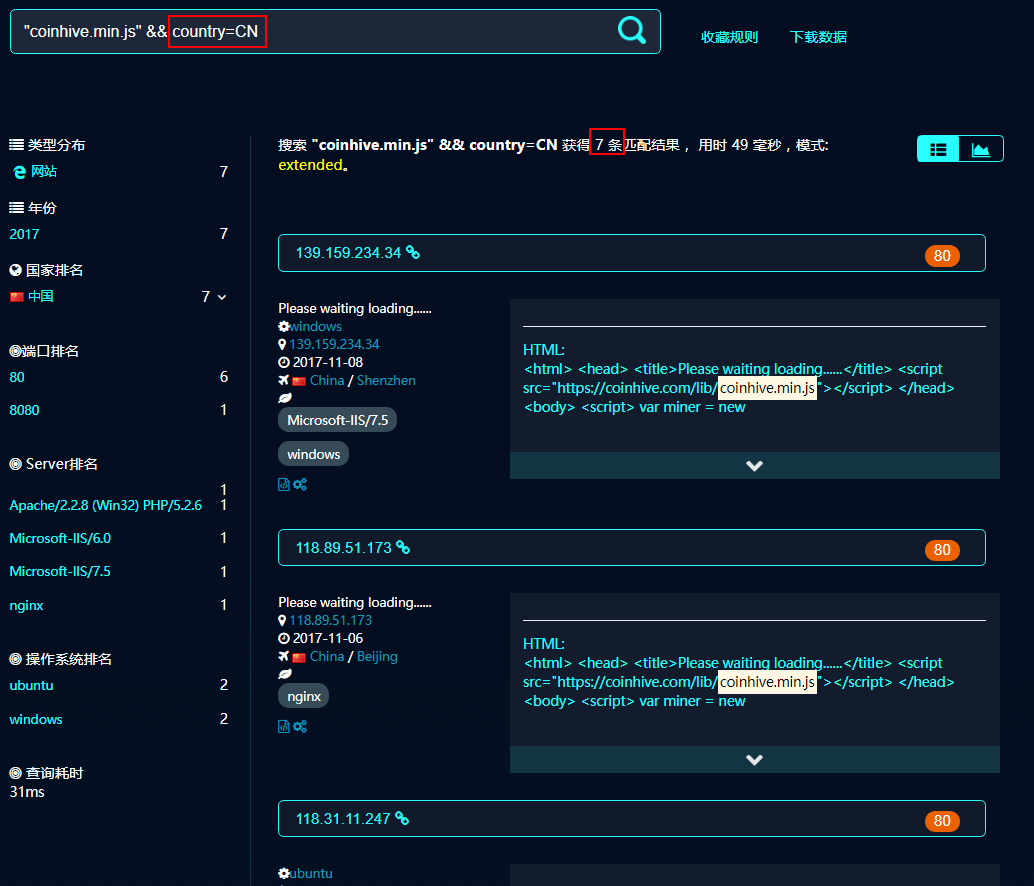
网址：http://40.74.52.110/m.html



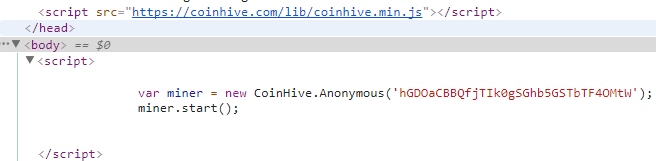


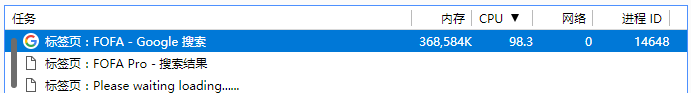
搜索中国存在的挖矿网站：

目前:



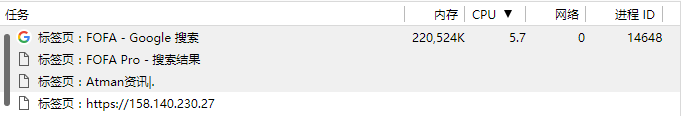
网址：http://139.159.234.34/



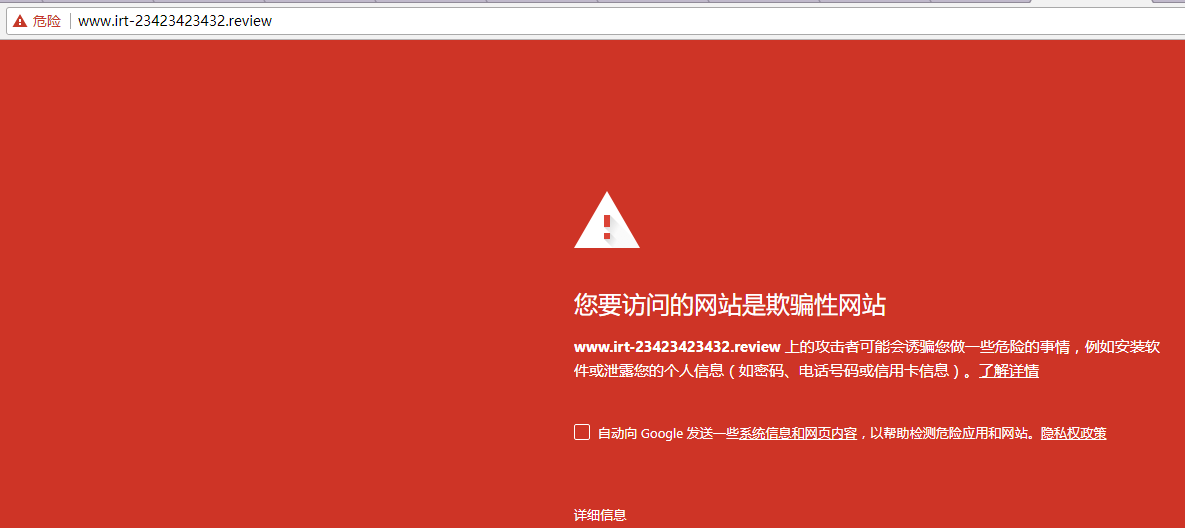


网址：https://158.140.230.27/





被谷歌识别出来的恶意网站：



## script代码分析

3个构造函数：

CoinHive.Anonymous(siteKey [, options]) //创建挖矿流程，但是未绑定到token或用户名上。

CoinHive.User(siteKey, userName [, options]) //开启并绑定所以的hash到用户上。

CoinHive.Token(siteKey, targetHashes [, options]) //开启挖矿流程，一旦指定hash找到则停止挖矿。

SiteKey注册网站给每个用户一个唯一的key作为标识，其是门罗币收款的账户。

Username：用户账户在网站上唯一标识符。

Options：

Threads：miner开启的线程数，默认是navigator.hardwareConcurrency,即CPU核数；

AutoThreads：是否自动调整线程数获得最佳性能，实验性功能，默认false；

Throttle：线程应该空闲的时间比例，默认为0，查看miner.setThrottle()设定；

forceASMJS：true表示miner使用asm.js的hsash算法实现，false表示使用更快的ebAssembly版本（如果支持），否则回退到asm.js，默认false；

theme：AuthedMine中屏幕可以选的颜色主题，light和dark，默认light；

language ：AuthedMine中屏幕可选的语言，默认的auto表示用户浏览器设定中的语言。

代码如：

var miner = new CoinHive.User('YOUR\_SITE\_KEY', 'john-doe', {

threads: 4,

autoThreads: false,

throttle: 0.8,

forceASMJS: false,

theme: 'dark',

language: 'auto'

});

.start([mode])方法：

连接到pool并启动mining。

可选mode指明当有一个miner在另外tab页运行时，此miner应该如何工作，默认是：CoinHive.IF\_EXCLUSIVE\_TAB；

**CoinHive.IF\_EXCLUSIVE\_TAB**：miner只有在其他tab页没有挖矿的情况下启动，其他tab页miner停止或关闭后开始工作；

**CoinHive.FORCE\_EXCLUSIVE\_TAB**：miner强制开始并杀死其他tab页中没有指定CoinHive.FORCE\_MULTI\_TAB 的miner；

**CoinHive.FORCE\_MULTI\_TAB**：miner总是开始，不会kill其他miner，也不会被其他miner杀死，但那时只能使用captcha（验证码）和shortlinks使用。

## Coinhive.com的库文件

https://coinhive.com/lib/

|—— coinhive.min.js // wss矿池通信分配worker交互实现

| :.. (CoinHive.CONFIG.WEBSOCKET\_SHARDS) // WebSocket地址连接pool进行任务发布等功能

| :.. (CoinHive.CRYPTONIGHT\_WORKER\_BLOB) // 以CreateURL形式动态生成Blob链接对worker内容实现

|—— cryptonight.wasm // WebAssembly实现预计源cryptonight.c & keccak.c 等

|—— cryptonight-asmjs.min.js // 兼容方案asmjs实现内容worker除外从L4851算法开始

|\_\_ cryptonight-asmjs.min.js.mem // 浏览器中asmjs所需模拟内存实现

cryptonight.wasm & cryptonight-asmjs.min.js主要就是cryptonight算法的实现；

CryptoNight算法：

是一个使用物理内存的高强度hash算法函数。 它的设计适用于GPUFPGA和ASIC架构上的有效性算力。步骤：

1. 初始化大型暂存器与伪随机数据；
2. 算法对暂存器中包含伪随机地址的大量的读/写计算操作；
3. 将整个暂存器的hash值进行hash效验验证本次计算产生的价值。

初始化流程：

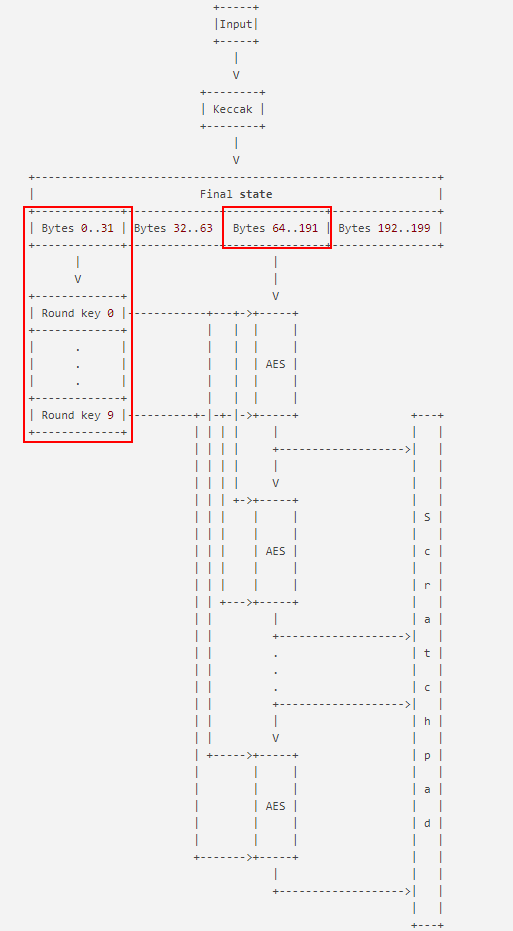
1）使用参数 b = 1600 和 c = 512 .对输入内容进行Keccak计算 [ KECCAK ]。计算结果的0..31字节用作AES-256密钥[AES]并扩展为10个循环密钥申请一个分配了2097152字节2 MB空间的暂存器；从计算结果的64..191字节处提取出来数据并分割成8个块每个块16字节。使用以下步骤对每个块进行加密



aes\_round() 函数执行一轮AES加密对本块执行 SubBytes ShiftRows 和 MixColumns 步骤其结果与round\_key进行异或运算。但这不同于AES加密算法第一轮计算和最后一轮计算没什么不同。

2）一轮下来得到的计算结果写入暂存器的前128个字节然后这些结果再次代入加密循环再把这次循环结果写入暂存器的第二个128字节里。这里每次往暂存器里写入下一个128字节都代表对先前写入的128字节内容在新一轮加密的结果。流程一直循环直到暂存器写满。至此一次算法的初始化就完成了。

该图表示暂存器初始化：



更多信息见详细网站：

Coinhive挖矿脚本分析与Pool改造自建(一)：

<http://www.freebuf.com/column/151316.html>

【不就是浏览器挖矿嘛】Coinhive挖矿脚本分析与Pool改造自建(二)：

http://www.freebuf.com/column/151376.html