Antibot Products Analysis

Techniques

- Reading (code, documentation, github issues)
- Googling
- Traffic analysis (burp, devtools)
- Manual testing

Tools

- Burp Suite
- Browsers + DevTools
- CLI HTTP tools: curl
- JS deobfuscators
- JS Beautifiers

Notes

- hypothesis
- test cases
- notes
- attacks
- commands (curl, queries)
- links

Datadome

https://datadome.co/

Документы

Описание продукта:
https://datadome.co/bot-management-protection/bot-protection-single-page-app-api/

Описание способов подключения продукта в веб-приложение (client-side):
https://docs.datadome.co/docs/protect-singlepage-app

Доступные облачные региональные эндпоинты:
https://docs.datadome.co/docs/api-server
https://status.datadome.co/

Приложение

Приложение, защищенное антиботом: https://datadome.co/

Анализ

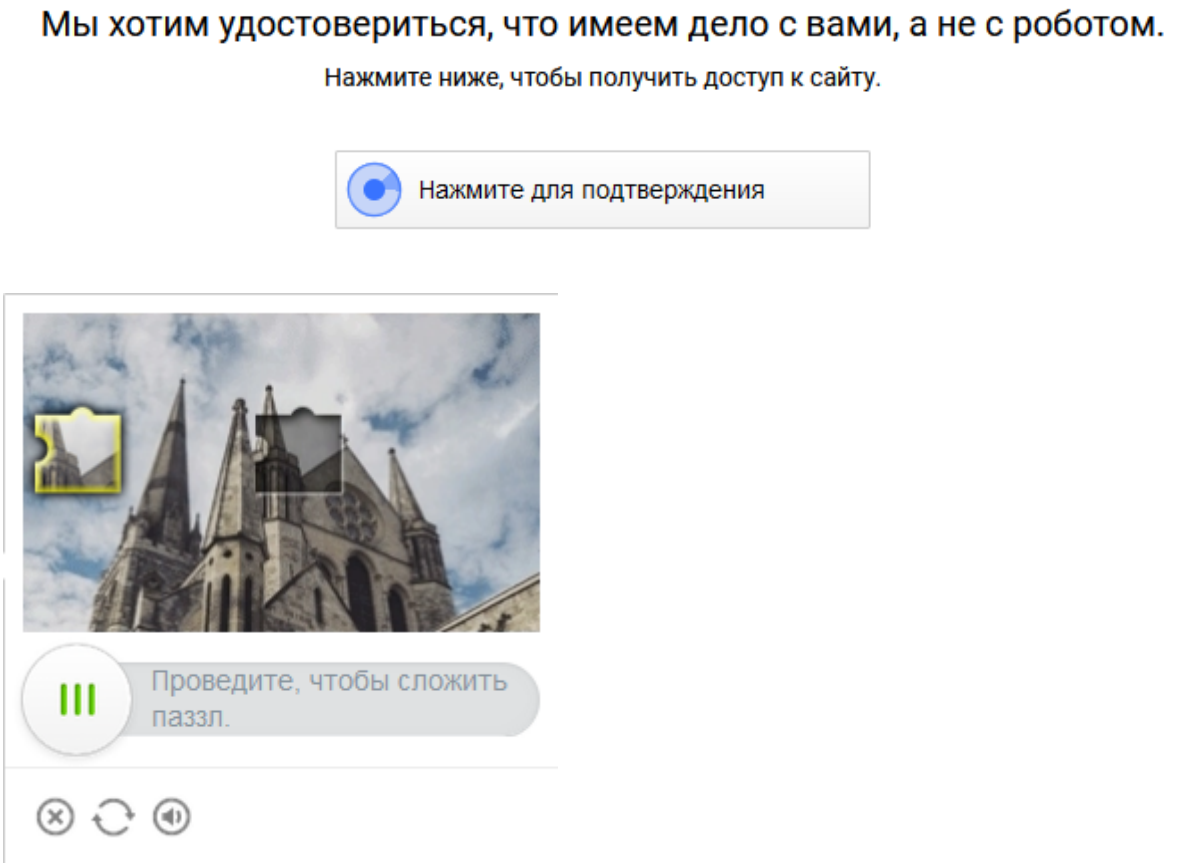
Обзор

Большинство антибот приложений, при работе на одностраничных сайтах не предоставляют подозрительному клиенту капчу, а сразу блокирует их.
Поскольку при блокировке запроса, API не отправляет ответ одностраничному приложению и приложение не может правильно интерпретировать ответ.
Из чего следует вероятность заблокировать легитимного пользователя.

Разработчики DataDome предлагают следующее решение проблемы: используя тег JavaScript, отслеживать вызовы AJAX на любой URL-адрес и отображать капчу поверх одностраничного приложения при блокировке API модулем DataDome.

Приложения, защищенные DD

Некоторые сайты, которые используют защиту от ботов от DataDome, просят пройти проверку, что пользователь не робот.



Были проведен тестн для сайта https://www.carsales.com.au/ . Мы получили исходную страницу сайта.В рамках тестов на сайт отправлялись запросы при помощи Selenium.

```
options = webdriver.ChromeOptions()
options.add_argument("start-maximized")
options.add_experimental_option("excludeSwitches", ["enable-automation"])
options.add_experimental_option('useAutomationExtension', False)
driver = webdriver.Chrome(options=options, executable_path=r'C:\WebDrivers\chromedriver.exe')
driver.get('https://www.carsales.com.au/')
print(driver.page_source)
```

В результате сработал механизм защиты и приложение потребовало решить капчу: (получили такой консольный выход)

```
<html><head><title>carsales.com.au</title><style>#cmsg{animation: A 1.5s;}@keyframes A{0%{opacity:0;}99%{opacity:0;}100%{opacity:1;}}</style><meta
 name="viewport" content="width=device-width, initial-scale=1.0"></head><body style="margin:0"><script>var dd={'cid':'AHrlqAAAAAMA8e3ZRgBIkcoABYAGgQ==',
 'hsh':'C0705ACD75EBF650A07FF8291D3528','t':'fe','s':6522,'host':'geo.captcha-delivery.com'}</script><script src="https://ct.captcha-delivery.com/c
 .js"></script><script>if("string"==typeof navigator.userAgent&&navigator.userAgent.indexOf("Firefox")>-1){var isIframeLoaded=!1,maxTimeoutMs=5e3;function
 iframeOnload(e){isIframeLoaded=!0;var a=document.getElementById("noiframe");a&&a.parentNode.removeChild(a)}var initialTime=(new Date).getTime();setTimeout
 (function(){isIframeLoaded||(new Date).getTime()-initialTime>maxTimeoutMs&&(document.body.innerHTML='<div id="noiframe">Please enable JS and disable any ad
  blocker</div>'+document.body.innerHTML)},maxTimeoutMs)}else function iframeOnload(){}</script><iframe src="https://geo.captcha-delivery
 .com/captcha/?initialCid=AHrlqAAAAAMA8e3ZRgBIkcoABYAGgQ%3D%3D&amp;hash=C0705ACD75EBF650A07FF8291D3528&amp;cid=71ho9XUL~YGgj0F6h0NDe8yImyA_05~xjn5z
 .W8OCqvjxGeHX3KFO7-bXR9vtQPWJQ5YhvkCtPUjoogUTChqjGx-MV5GAx2qqjqQ-6VG1S&amp;t=fe&amp;referer=https%3A%2F%2Fwww.carsales.com.au%2F&amp;s=6522" width="100%"
 height="100%" style="height:100vh;" frameborder="0" border="0" scrolling="yes" onload="iframeOnload()"></iframe>
</body></html>
```

html страница с проверкой.

PerimeterX

https://www.perimeterx.com/

Документы

https://www.perimeterx.com/resources/blog/2020/api-bot-attacks-the-hidden-threat-to-application-security/ - общий принцип работы

https://www.perimeterx.com/downloads/product-briefs/PerimeterX-Product-Brief-PerimeterX-Bot-Defender-Architected-for-Low-Latency.pdf - общее описание работы

Приложение

Приложения, защищенные антиботом:

- https://www.build.com/
- https://www.booking.com/
- https://www.bkstr.com/
- https://www.elfcosmetics.com/

Анализ

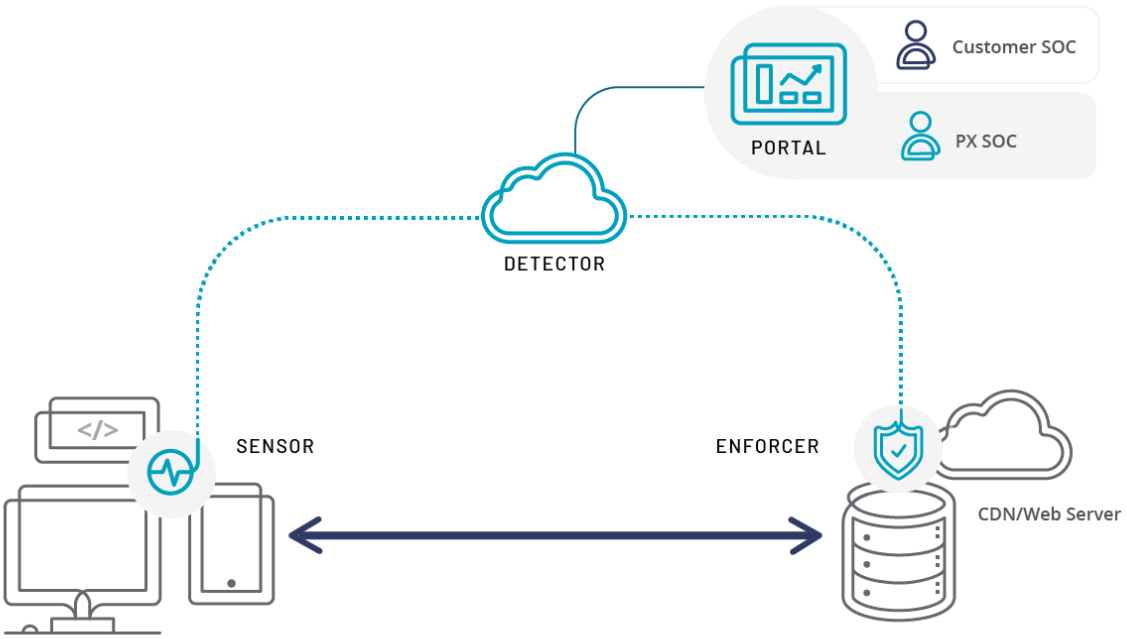
Overview

PerimeterX Bot Defender combines intelligent fingerprinting, behavioral signals and predictive analysis to detect bots on your web and mobile applications and API endpoints.

Bot Defender offers a range of enforcement actions to mitigate the impact of unwanted bots on your business. These actions range from blocking the bots completely, rate-limiting them, or redirecting them to decoy sites. Innovative capabilities like Human Challenge enable you to stay ahead of CAPTCHA-solving bots, improve human solve times and reduce abandonment

Bot Defender offers rich analytics through the Bot Defender Portal, which includes pre-built as well as customizable dashboards.
(from https://www.perimeterx.com/products/bot-defender)

How it Works



- **Sensor**
  The Sensor collects and sends hundreds of client-side indicators and signal to the Detector. These signals are used for validation of human versus bot activity, identification of suspicious script activity and malicious browser extensions to create the device, browser and browser extension fingerprints and script baselines. The same Sensor collects the signals asynchronously for the entire PerimeterX portfolio.

- **Detector**
  The machine learning (ML) based Detector continuously learns the common characteristics of human interactions, correlates it with customer-defined policies and updates the Sensor with new intelligence. The Detector maintains a repository of known attacks, shared with all customers, so malicious actions can be blocked quickly. The Detector frequently updates its data set and augments its ML features based on internal and external data. The Detector processes billions of events every day to provide best-in-class detection.

- **Enforcer**
  The Enforcer is the gatekeeper for threat response policies generated by the Detector. It enriches and mitigates automated traffic according to business needs. The Enforcer also continuously learns and updates the Detector with relevant data. The Enforcer can be deployed inline into any existing web architecture. In the case where there is no requirement for enforcement on the server, the enforcement is done on the client-side.

- **Portal**
  The cloud-based management Portal displays all activity -in real time- including attacks, blocked requests, traffic trends and top threat reports. Manage the configuration and settings for your PerimeterX products from a single pane of glass that features an intuitive admin interface. With it, you can investigate attacks and create custom reports. The portal also supports Security Assertion Markup Language (SAML) authentication, including out-of-the-box integrations with Google and Okta.

Demo can be requested after personal meeting:
https://www.perimeterx.com/request-demo/?source=products/bot-defender

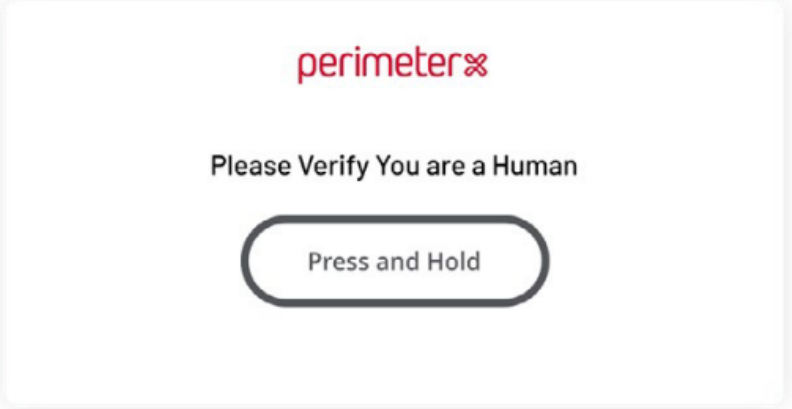
Human Challenge

(snips from product brief)

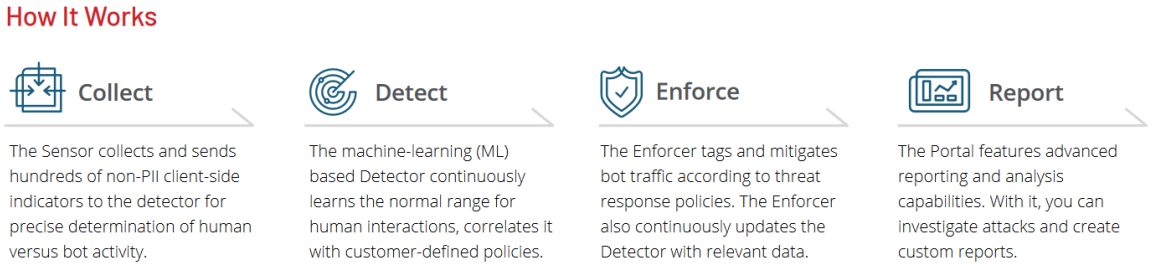
...Human Challenge leverages PerimeterX **machine-learning-based** technology…

By collecting additional in-depth behavioral data about the user, Human Challenge detects and blocks CAPTCHA-solving bots, whether they use automated or manual methods.

How captcha looks



Main principles



Steps of analysis

- The first step is to **collect behavioral**, **network** and other **fingerprints** from normal users as baseline to detect API bot behaviors in runtime. These can include **signals from how real users behave**, what their **Web API traffic** throws off, **cookie analysis** (and their absence), and **signals from mobile applications** such as mobile IDs and application tokens.

- For direct APIs, you need to look for signals in the network such as **network response times and patterns**, **network fingerprinting**, and **evidence of obfuscation** techniques such as using proxy networks.

- These signals should be combined with internal and external **reputation feeds** to evaluate the likelihood that a call is coming from a good user or a good bot rather than a malicious bot.

- Lastly, you must **include feedback loops that are application specific** - such as changes in conversion rates, log-in success rates, and traffic volumes to product pages, to name a few. All of this data can be used to build robust models of what is good, bad and unknown API traffic.

- With the model, you can detect malicious API bots by continuously processing signals given off by each API request. You will need to use **advanced machine learning and behavioral analytics** built to respond at web scale and in real time. The detection model will be constantly comparing behaviors and signals to those of real user signals, and assigning a risk score to each API request.

(from https://www.perimeterx.com/downloads/product-briefs/PerimeterX-Product-Brief-Bot-Defender.pdf)

User processing variants

- Hard block - block the user and terminate their session
- Allowlist - determine the user or the bot is legitimate and allow them in
- Rate limit - protect application performance by setting limits of how many time a partner bot or good bot can access an API
- Redirect - send an API request to a specific URL for further instructions or further user actions
- More actions can be applied like slow and delay the user (instead of blocking) and monitor the activity

(from https://www.perimeterx.com/resources/blog/2020/api-bot-attacks-the-hidden-threat-to-application-security/)

Bot Defender

(snips from product brief)

PerimeterX platform integrates at the **client (PX Sensor)** and at the **server (PX Enforcer)** level.

The **client-side component is a lightweight JavaScript** for web (PX Web Sensor), and **mobile SDK** for native apps (PX Mobile Sensor for iOS and Android), where the **PX token sent is a cookie** for the web user and a **header** for the mobile user.

The PerimeterX Web Sensor is lightweight (~30 kb), cacheable and operates asynchronously to ensure minimal latency for the page load times. The PX Sensor executes after the DOM-ready event, and most of the PX Sensor actions are run only once per session to limit the performance impact, so the user experience on the website is not affected.

(v start v) Interesting links

https://github.com/incizzle/perimeterx-tools - PX payload decoder/encoder
https://github.com/fiverr/perimeterx-axios-interceptor

https://www.g2.com/products/perimeterx-bot-defender/reviews#survey-response-4543044 - PX reviews
https://github.com/Greg-Pricemole/perimeterx - PX antibot reverse notes

Looking at traffic

Let's find any scripts that PX loads when visiting protected websites.

Opening https://build.com in Burp:

Script from https://www.build.com/2Ztkihy4/init.js is loaded with this comment at the beginning:

```
// @license Copyright (C) 2014-2021 PerimeterX, Inc
(www.perimeterx.com).  Content of this file can not be copied and/or
distributed.
```

(Does every script from PX has this copyright comment?)
(Couldn't find another .js with this comment)

After this POST request is sent (actually many times) to https://www.build.com/2Ztkihy4/xhr/api/v2/collector with body like (one example):

payload=aUkQRhAIEGJqABAeEFYQCEkQYmoLBBAIEFpGRkJBCB0dRUVFHFBHW15WHFFd
Xx0QHhBiagQBEAgQfltcR0oSSgoEbQQGEB4QYmoDCwMQCAIeEGJqCgcCEAgCHhBiagoH
AxAIAAcLBAYeEGJqAwICChAIAQQCAh4QYmoDAgcHEAgDBAAEAAcCAwcEAwoBHhBiagMC
BwQQCAMEAAQABwIDBwQDCwQeEGJqAwIBChAIEFEKAVdQVgECH1cGBVMfAwNXUB9QAlRQ
HwEHV1cKBgcDUVAEBxAeEGJqAQUDEAhGQEdXT09v&appId=PX2Ztkihy4&tag=v6.7.9
&**uuid**=c83ebd30-e47a-11eb-b0fb-35ee8451cb65&ft=221&seq=0&en=NTA&pc=60
68351429910118&**pxhd**=ZKDU0DIPjEYA9wn2/DsoDV85UEAKjTccsuH9rDuDF270sMea
3F/ZBpa7kISYlpF9UfDo3dpuUMeBM651L1VkXA==:GrMZ2f8HO72KkwVUu/JsCOeH9pH
EvHtFml74b/dOPoWsjLboxbhFoPFP/OYEtPbmywbe5VQTL3016OjdL0hTdNxi6sbS0Aq
/e-Fr5qKnQ64=&rsc=1

And response body is:

{"do":["bake|_px2|330|eyJ1IjoiYzgzZWJkMzAtZTQ3YS0xMWViLWIwZmItMzVlZT
g0NTFjYjY1IiwidiI6ImJiOGQ2MTQxLWU0N2EtMTFlYi1iZDJlLTc5YWJmYmQ2Y2FkYS
IsInQiOjE2MjYyNTA5MjUwMDksImgiOiJmODY3NDYxOTgzOGQ5MGNkNTdlNGFkYWQ4Zj
ZlMDM5MTQ1Y2Q2MWJjMTk3ZjdjMDkwYzkyNWRlYjg5MGExZTY1In0=|true|300","te
|score|0|binary","sid|dfb05e00-e47b-11eb-b21a-efd5b423d443","cls|380
19761381479454133","sts|1626250625009","wcs|c3n9r01gbal79tclp050","d
rc|2178","cts|dfb0ac20-e47b-11eb-b21a-efd5b423d443|true","cs|1c10863
bff60ae86117c622a726c8cd9e74e204a48f6b6f96b718716b25cc693","vid|bb8d
6141-e47a-11eb-bd2e-79abfbd6cada|31536000|true","sff|cc|60|U2FtZVNpd

GU9TGF4Ow==","en|_pxde|330|101b39c4b0ec84f11a628d57c5c6e5a3cd8c65e1d047248504e63c035034056b:eyJ0aW1lc3RhbXAiOjE2MjYyNTA2MjUwMDksImluY19pZCI6WyJhMjVmYWUyNjdmZGJhYzc3NmM5NGZlNmFiNjY5OGZhZSJdfQ==|true|300"]}

(What does this body do/mean?)

To other endpoint: https://www.build.com/2Ztkihy4/xhr/api/v2/collector/beacon same looking payloads are sent.

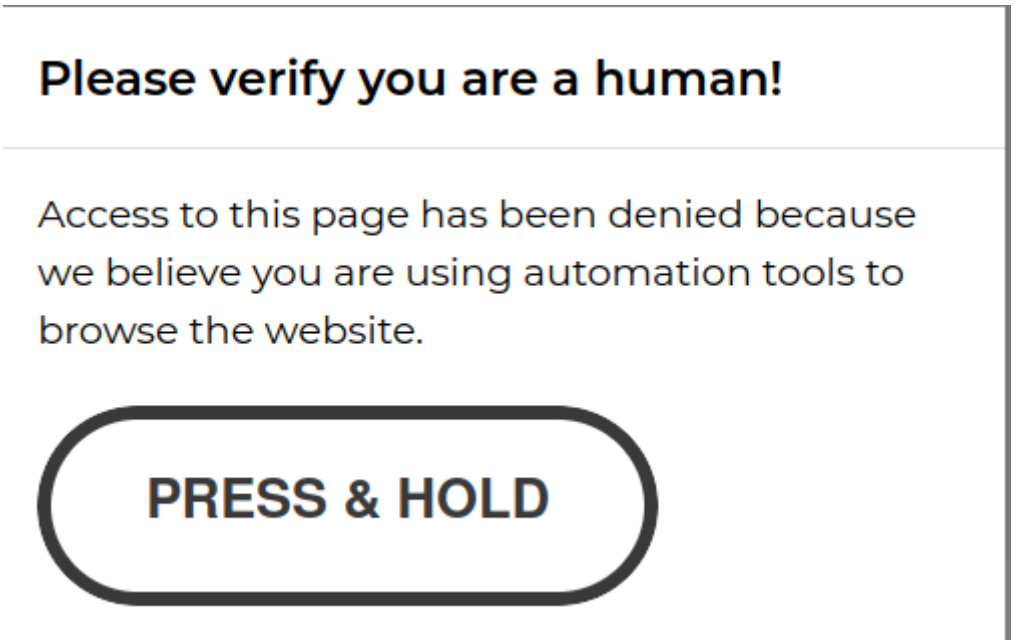
Payloads can be decoded using this tool: `https://px.incizzle.dev/decode`

In decoded payload some things can be easily identified, like: mouse position (movements?), which keyboard keys where pressed, what elements on page was clicked, OS, page visited

I ommited cookies (they are large; need to understand which were set by PX) in requests for now.

Let's analyze http traffic when visiting other protected website. To look for similar behaviour or identical cookies.

Let's visit https://www.bkstr.com/
This time I instantly got captcha:



On page update captcha remains. Captcha is on another page (redirect) so it can't be just igonred.
But on some other websites protected by PX it can just be bypassed by page update.
So it probably depends on configuration.

(Check what is sent on solving or failing captcha)

On this website name of PX .js file is not obfuscated (called main.min.js) and file is loaded from external domain. If browser or ad blocking add-on blocks script as fingerprinting tool, then website is trying to load it from another domains.

- Cookies that could be related to PX:

_pxhd=ZKDU0DIPjEYA9wn2/DsoDV85UEAKjTccsuH9rDuDF270sMea3F/ZBpa7kISYlpF9UfDo3dpuUMeBM651L1VkXA==:GrMZ2f8HO72KkwVUu/JsCOeH9pHEvHtFml74b/dOPoWsjLboxbhFoPFP/OYEtPbmywbe5VQTL3016OjdL0hTdNxi6sbS0Aq/e-Fr5qKnQ64=;

_px_f394gi7Fvmc43dfg_user_id=Y2IzZjUzMDAtZTQ3YS0xMWViLTljYWQtM2Q4NjcxOTlhNWVh; AMCVS_F5FA1253512D2B590A490D45%40AdobeOrg=1;

s_cc=true; _ga_THD421G9QG=

pxcts=61d498e0-e488-11eb-936c-19f570250ad1;

_pxvid=61d4457f-e488-11eb-be7f-0242ac12000a; usprivacy=1YYN; __pxvid=6213b013-e488-11eb-adcb-0242ac110002;

fsbotchecked=true;

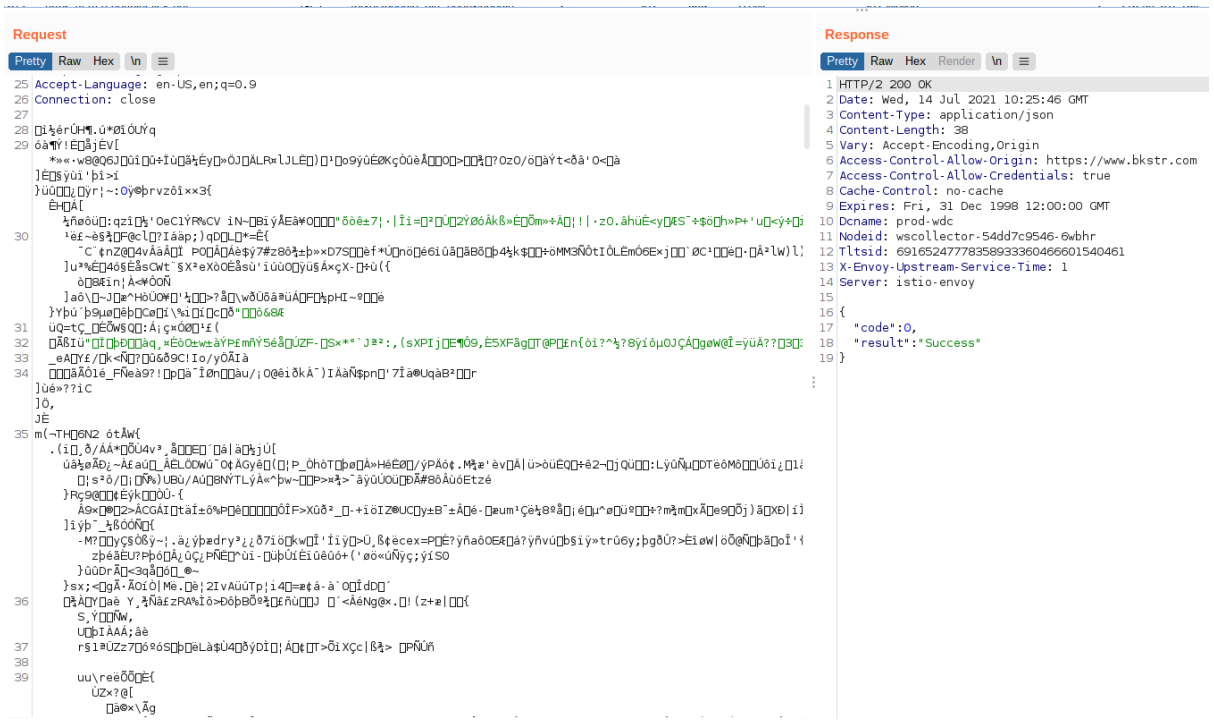
- Common for both websites cookies:

s_cc

---

strange post request: looks like encrypted

https://b.px-cdn.net/api/v1/PX6tnJz910/d/p



req:

```
------------------bx6gptnxe8xfdouw
Content-Disposition: form-data; name="m"

c=sx
------------------bx6gptnxe8xfdouw
Content-Disposition: form-data; name="p"

§\¯ë$}2}e$}615}e+-*:s}>//;}e}i+1%fno}s}/'0-6861}e}}s}+>8}e}nqgqj}s}37-}e}7++/,epp(((q=4,+-q<02p:9033:++,
+0-:p702:}s}<<,}e}m<9i;imi<lo=;jg=ii:=;fjoh9:9nj<<<<f>:9i9}s}>*+0+,}e}}s}**6^Àÿin:>i>;or:kggrnn:=rf;hmr=
=l9mlnkfoo=}s},6NrWfn:Ýr_w:.}eos}=;)6VhG;kkjh9rF;÷=:h9romkm><nmooo>}"s}/}e$}9o'jf<hil<:}enos^N{hmlkikfi}
e^Q{lfg=n=g<}s^Qglhm=k9}e}61*''giik}VCgnm:n=}e}nogoenfm~Ù^={n:i:h<9}e^PKf<:9mmVkCh;mgifh9VQ{io:::9k<}Zch
jg<m<=}e+-*:""
------------------bx6gptnxe8xfdouw--
```

resp:

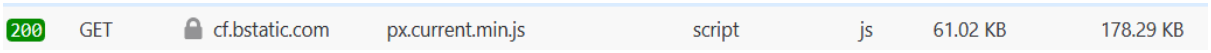
{"d":[{"c":"cs","a":["e7b1fcfab649d82092e112dbb8be2084dac279fab5df1bbbaa894912065a6807"]},{"c":"vid","a":["6213b013-e488-11eb-adcb-0242ac110002"]},{"c":"w","a":["0"]}]}

Analysis of work

Parameters to be collected:

Short description: gets plugins, navigator, screen data, touchpoints, platform, language, product sub, app version, geolocation, mimetypes, build ID, color depth, battery, hardware concurrency, device pixel ratio, local storage.

Full list of user characteristics processed by antibot service: we can get this information in “px.current.min.js” . In this JS we can see that the parameters being collected are not hidden. We can also restore JS logic. (https://www.booking.com/)



navigator.userAgent navigator.maxTouchPoints navigator.msMaxTouchPoints navigator.sendBeacon navigator.language navigator.languages navigator.platform, navigator.doNotTrack	navigator.geolocation navigator.mimeTypes navigator.product navigator.productSub, navigator.appVersion navigator.appName navigator.appCodeName navigator.buildID navigator.permissions	navigator.onLine navigator.cookieEnabled navigator.battery navigator.getBattery navigator.mimeType navigator.plugins navigator.plugins.length navigator.hardwareConcurrency navigator.cpuClass

Triggers:

- The antibot system **doesn’t block** requests from a **simple crawler bot** (our test assignment, Golang). Requests (~4000) were sent to a single thread within 60 min.
- **Crawling** with **Selenium** is also **available**, also in headless mode.
- Imitation of user actions (go to the site, click on the button) (Selenium) is also not blocked by antibot (so far) (¯_(⊙_⊙)_/¯)

Features of the antibot system:

- We didn't find JS, crypto challenge, cookie challenge, etc.
- You can disable JS and continue working with the site.