University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Tomasz Kępa
Student no. 359746

Detecting Anti-Adblockers using Differential Execution Analysis

Master's thesis
in COMPUTER SCIENCE

Supervisor:
Dr Konrad Durnoga
Institute of Informatics

August 2019

Supervisor's statement

Hereby I confirm that the presented thesis was prepared under my supervision and that it fulfils the requirements for the degree of Master of Computer Science.

Date Supervisor's signature

Author's statement

Hereby I declare that the presented thesis was prepared by me and none of its contents was obtained by means that are against the law.

The thesis has never before been a subject of any procedure of obtaining an academic degree.

Moreover, I declare that the present version of the thesis is identical to the attached electronic version.

Date Author's signature

Abstract

Ads are the main source of income of numerous websites. However, some of them are fairly annoying which causes many users to use adblocking browser extensions. Some services, in turn, use specialized scripts to detect such plug-ins and silently report them or block some content as a punishment. The goal of this thesis is to build a pipeline for detecting such scripts based on a differential execution analysis, a method provided by other authors in 2018. Such a mechanism can be used later to analyze the prevalence of anti-adblockers on Polish websites or to build an extension capable of circumventing such scripts.

Keywords

dynamic analysis, differential execution analysis, javascript, anti-adblockers, ads

Thesis domain (Socrates-Erasmus subject area codes)

11.3 Informatics, Computer Science

Subject classification

Software and its engineering. Dynamic analysis

Tytuł pracy w języku polskim

Wykrywanie skryptów blokujących rozszerzenia typu AdBlock w przeglądarkach

Contents

Introduction

In the modern-era Internet the majority of web pages operate for profit. Most of them, however, choose to provide free content in exchange for displaying paid advertisements, or ads for short. Unfortunately, not all websites play fair. Some of them concentrate on displaying as many ads as possible, and generate traffic by using click-baits and other shady practices. Even web pages with valuable content can have an overwhelming amount of ads. This leads to grave dissatisfaction of some portion of users. To make their browsing experience better, they turn to ad-blocking extensions.

Operation of the most popular adblockers is based on user-curated lists. They maintain a collection of filters that detect and hide banners and other forms of advertisements. Such extensions have been around for such a long time that they became really effective. Not only they are able to block the vast majority of ads but also rearrange the remaining content to provide seamless experience. Moreover, they have an additional advantage of saving bandwidth, which results in shorter loading times and savings in data usage. It is such a common occurrence that even some browsers have ad-blocking functionality built-in.

Some big companies, e.g. Facebook, are able to circumvent ad-blocking extensions, but they have to put an ongoing effort to be able to catch up.

The amount of users utilizing adblockers cannot be ignored. In 2019 there was over 615 million devices worldwide with an adblocker installed [35]. This, in turn, leads to loss of revenue for many businesses. To combat this, some of them choose to deploy anti-adblockers, i.e., scripts that detect ad-blocking extensions and react in some way. They can generate a visible warnings or even block the website's content entirely.

The anti-adblockers come in many different fashions. Some of them are simple, custom-made scripts that set up an obvious bait and try to detect a reaction typical to ad-blocking extensions. They may check if some file was not loaded or if the advertisement banners have been tampered with. Some solutions perform more than one check. The most advanced ones also employ mechanisms that make the analysis and reverse-engineering harder. Examples include code minification or obfuscation involving *eval* function.

Some ad providers, publishers and extension creators recognize users' discontent and start initiatives like Acceptable Ads [1]. The idea is to have a white list of ads that meet certain criteria such as non-intrusiveness. Such list is embedded into ad-blocking extensions and enabled by default. Others just focus on making sure that the ads are displayed and continue to bring profit. The way to do that is by constantly improving anti-adblocking solutions.

The people behind adblockers have mixed approaches to anti-adblocking warnings. Some of them choose to let all of them be, others only if they are dismissable (and thus less effective from the point of view of the site owners). Naturally, there are also other players who develop extensions which block all such warnings, regardless of their intrusiveness. The whole conflict of interests, opinions and values leads to an arms race.

A regular study of anti-adblocking scripts can help both sides of the barricade. Knowing how such scripts behave can help create better methods of detecting them. That, in turn,

can lead to creation of better blocking tools. On the other hand, being able to automatically detect some scripts usually means that it is also possible to block them. As a consequence, studying different detection methods can lead to better anti-adblockers as well.

In 2018 Zhu et al. [47] proposed a sophisticated method of automated detection of such scripts. This method uses what is called a Differention Execution Analysis. The whole approach consists of several steps. First, at least two JavaScript execution traces have to be collected. An execution trace is a list of all events such as function enter or leave that occurred during program execution. The first trace is of website's code executed in an environment without any ad-blocking extensions. The second one is collected with such a plugin active. These two traces are later compared and checked if there are any differences between them. If there are, they can be attributed to anti-adblocking scripts.

There are lots of difficulties to get the whole mechanism working. First of all, trace collection has to be hand-made. There is no off-the-shelf solution that would meet all requirements of this method. Second, JavaScript event-based execution model makes it problematic to compare execution traces as the order and number of events can differ greatly. This issue is solved by processes called trace slicing (or untangling) and trace matching. The former is a method of gathering execution events into subtraces that correspond to different events. The latter is a process of pairing subtraces from two different execution traces. Third, a website can incorporate tons of noise. The web page can utilize some random functions, there can be network errors, the content can be generated or selected dynamically. The authors of the method give a glimpse of how to battle each of the difficulties, but leave out a lot of details in most cases.

The aim of this work was to implement a similar system based on the aforementioned idea. The contributions of the thesis are the following:

- An automated system analyzing web pages has been implemented. It takes a list of websites, collects the traces (with and without an adblocker), filters and analyzes them to find execution differences. Each step of the pipeline has been described in detail.
- A novel approach based on Stable Marriage Problem to solve trace matching problem has been introduced.
- A new approach to filtering execution noises has been introduced and tested.
- The entire pipeline has been evaluated and used to conduct a small-scale study on the most popular websites in Poland.
- A by-product of performed experiments is an overview of the anti-adblocking methods.

The first part of the system is Chromium browser with manually instrumented V8 engine [43]. Modified V8 produces JavaScript execution traces. The browser is run automatically a few times for each tested website. After each run, saved traces are filtered. When all traces are collected, the analysis is run to detect execution differences.

The implementation has been tested on 100 websites and the results verified manually. Half of the examples were positive ones (pages with anti-adblockers). The system proved to be a useful tool in detecting and analyzing anti-adblocking scripts. The analysis of those websites resulted in identification of the most popular mechanisms and solutions. The implementation was able to pinpoint relevant execution divergences in more than 80% cases.

The study conducted on 300 most popular web pages in Poland revealed that anti-adblocking scripts are far more widespread than what can be conjectured by just studying visible reactions. Visible reactions were found in a little more that 10% of websites, while if we consider all types, such as silent reporting, they seem to be more than 5 times as frequent.

Chapter 1

Basic concepts

In this chapter, we introduce the most essential concepts used throughout this thesis and provide the motivation for studying adblockers and anti-adblockers.

1.1. Definitions

- Execution event – each occurrence of some statement or expression being evaluated, e.g., entering a function or taking a "then" branch in the "if" statement.
- Execution trace – a series of execution events collected during program execution. It is dependent both on program structure and its input (also implicit, such as randomly generated numbers).
- Positive trace – an execution trace collected with an active ad-blocking extension.
- Negative trace – an execution trace collected without an ad-blocking extension.
- Execution index – any function that uniquely identifies execution points. In our case it is a statement source map information, i.e., file name and precise location of the statement, with the current function stack. This concept was formally introduced by Xin et al. [46].

1.2. Adblockers

Online advertising market is growing each year. The Interactive Advertising Bureau's report for 2018 [25] states that the Internet advertising revenues surpassed $100 billion annually in the US in 2018. Moreover, according the same report, online advertising is the biggest ad media, approximately 30% bigger than TV.

This means that there is going to be more and more ads. Some sources estimate that an average Internet user is served 11250 ads per month [13].

Unsurprisingly, ads are the main source of income for numerous websites. By displaying ads, authors are able to provide valuable content free of charge.

However, there are multiple reasons why users may not want to see online advertisements [26, 35]:

- There are websites which sole purpose is to earn money by displaying as many ads as possible, without providing any interesting or original content. They often generate traffic using click-baits and similar shady practices.

- Ads often lengthen pages loading times. The slower the connection, the more annoying it becomes.
- Ads increase web page payload size which incurs higher cost on mobile connections.
- Some ads track users which raises privacy concerns.
- Ads can be used to spread malware (in this case it is called Adware) [3].
- Users may find some of the ads annoying. The most notorious examples include pop-ups and video ads. Both are intrusive and negatively affect the comfort of using the website.
- Many advertisements are irrelevant to the content of the web page.
- They occupy the screen space, leaving less area for the content, which can be especially frustrating on devices with small screens.

One of the solutions is to use special browser extensions, called ad blockers (or adblockers), that prevent ads from being displayed. Their work is based on community-curated list of filters. These filters are used to identify some portions of the website as ads and then to remove them.

The removal depends on the type of an ad. Whole page overlays can simply be hidden without disrupting the website UI. On the other hand, after banners removal, there remains some blank space which is usually fixed by repositioning adjacent elements. In some cases, particularly when ads are served from domain of some ad provider, the requests that download ads can be blocked, thus saving the bandwidth and reducing data usage.

There is some controversy concerning morality of use of such extensions. One of the most popular ad-blocking extension, uBlock Origin states in its manifesto, that it is the users' right to have control over what content should be accepted in their browser [39]. As if other arguments for using adblockers were not enough of a justification.

Some ad-serving companies and extension authors recognize users' frustration and create initiatives like Acceptable Ads [1]. In this program, advertisements meeting certain criteria are whitelisted by a special committee. The whitelist is active by default in all adblocking extensions that join the initiative. Some bigger ad providers are also charged for being whitelisted.

1.3. Anti-adblockers

In 2017 PageFair prepared a report on adblockers usage [35]. Worldwide, 11% users use adblocking solutions. This may seem small, but it is not not uncommon for some markets to reach 20% or higher penetration (e.g. USA – 18%, Germany – 29%, Indonesia – 58%).

Such high percentage of users not seeing ads means lost income for many businesses. To recover lost revenue, many websites started deploying anti-adblocking scripts. Their goal is simple – when they detect that content blocking extension is present, they take some action, potentially mitigating the problem. The action can vary from simply just reporting the use of extension to the backend to blocking the content entirely.

The aforementioned PageFair report studied so-called "adblock walls", i.e., a mechanism preventing users from seeing the website content when they have an adblocker enabled. The study shows that 90% of ablockers users have encountered an adblock wall. More interestingly, 74% of them leave the website when confronted with such a wall.

Anti-adblockers come in many variants. There are simple, custom scripts written specifically for one service, but there are also some sophisticated scripts provided by third parties, designed to be easily integrated on any site.

One rather simple example is presented by a company offering the "Adblock Analytics" service [2]. In their example they add a file named *ads.js* with a short JavaScript code that adds a hidden div block with a unique id. Ad-blocking extensions usually block files named like that. All that remains to detect the extension is to check whether the div element was indeed added to the DOM tree. The lack of it indicates that an adblocker is active.

Another example is the "BlockAdBlock" module [37]. Similarly to the first first example, it first sets up a bait, i.e., a component that should look like an ad to the adblocker, but then the detection phase is a little bit more thorough. It runs the check several times and it inspects several properties of the bait to make sure it has not been tampered with. If something changed – an adblocker is detected. It also allows to register callbacks for both outcomes of the analysis.

Even people that base their websites on content managements systems have an easy access to such scripts. Most notably, around 1 in 3 out of the top 10 million most popular website runs on WordPress [45], and in this case there are multiple extensions available, varying in price and capabilities [5].

Naturally, there are multiple users that do not want to disable their adblocker to see the website's content. As a reaction, there are several solutions developed to combat anti-adblockers. Usually, they focus on providing new filtering lists, that can be used by the existing ad-blocking extensions [32, 36]

Unfortunately, these solutions have rather poor effectiveness, as reported by Zhu et al. [47]. This inability to effectively defuse anti-blockers was their most important reason to develop a method of detecting such scripts.

1.4. Differential Execution Analysis

Differential Execution Analysis is a dynamic program analysis method. Its goal is to pinpoint exact differences in two executions of the same program with different inputs or other conditions (e.g. network or memory errors). A good overview of the method is presented by Johnson et al. [27]

The analysis is usually performed by collecting an execution trace for each run and then comparing them using trace alignment. Trace alignment is a process of identifying which fragments of execution traces are common, where they diverge, and when they converge again. The exact algorithm is discussed in Section 4.3.

The results can be useful in various scenarios, e.g., debugging a program or during security analysis.

1.5. Detecting anti-adblockers

Zhu et al. [47] introduced a novel method of detecting anti-adblockers using Differential Execution Analysis. The work presented here is based on this method. The differences and new ideas are explained in the later sections.

The approach is quite simple. They collect execution traces of JavaScript code on a given website, first without any content blocking and then with an adblocker turned on. Afterwards, they analyze such traces using Differential Execution Analysis. Any differences in execution are attributed to the anti-adblocking activities of the website.

While the idea is pretty straightforward, the are multiple challenges here. First, trace collection is not a trivial task, especially when the interest exceeds just function entry and exit events. The authors instrument the V8 JavaScript engine [43] to achieve the task but do not provide much details, apart from briefly stating that their instrumentation is embedded into the native code generation process.

Second, due to JavaScript execution model, described in detail in Section 1.6, execution traces of different events are interleaved. To battle this issue, traces have to be sliced into subtraces and analyzed pairwise. In a language with a simpler execution model this step would be unnecessary. Also, the authors do not explain their method of pairing the subtraces. They just mention that all $m \times n$ pairs have to be analyzed.

Lastly, the biggest challenge is to combat execution noises, e.g., page randomness or variable content. The authors solve the issue by loading the same page three times with an adblocker and three times without it, and use redundant traces to generate a blacklist of execution differences.

1.6. JavaScript execution model

JavaScript concurrency model is based on an "event loop" [33]. The engine is essentially single-threaded[1] and concurrency is achieved by utilizing a message queue. This queue processes events one by one, till completion, i.e., a function corresponding to the message starts with a new, empty stack and its processing is done when the stack becomes empty.

The easiest way to add new events to the queue is by calling *setTimeout* or *setInterval*. Furthermore, all callbacks attached to DOM events (e.g. *onClick*) are executed by adding an event to the queue.

It is worth noting that execution of functions can be intertwined, e.g., when generators are used. This is the reason why trace slicing is needed.

Each iframe and browser tab has its own execution environment that include a separate message loop. We elaborate on this in Section 3.2.

[1] With the exception of Web Workers introduced in HTML5. Since they are not a common occurrence, we consider them to be out of the scope of this thesis

Chapter 2

Trace collection

In this chapter we discuss available methods of collecting execution traces of JavaScript code.

There are a few distinct ways to obtain an execution trace. The simplest (and the most limited) methods use only mechanisms present in the language. Others inject special tracing code to the analyzed script. The most profound ones modify the engine to produce the desired data.

2.1. Dynamic in-JavaScript code injection

JavaScript is a very dynamic language. For this reason, it is relatively easy to write code that will modify each function present in the environment to log each entry and exit, possibly along with all the arguments and the returned value [38].

Listing 2.1 presents a code example that instruments all functions in a selected object to log their name and arguments when they are called. Function *instrument* goes through all properties of an object and replaces each member function with a new function that first logs the function name and all provided arguments and then calls the original function. This function can be easily extended to also log the returned value and to instrument all subobjects recursively.

However, this is not enough for the needs of Differential Execution Analysis. The most obvious limitation is that it is not possible to instrument control statements. Another major shortcoming is that functions can be instrumented only after they are defined. It is quite an obstacle, because JavaScript allows to define function practically anywhere and it is very common to use anonymous functions as callbacks. It is not possible to instrument such callbacks without modifying the instrumented code which brings us to the static injection methods.

```
function instrument(obj, withFn) {
    for (const name of Object.keys(obj)) {
        const fn = obj[name];
        if (typeof fn === 'function') {
            obj[name] = (function() {
                return function(...args) {
                    withFn(name, ...args);
                    return fn(...args);
                }
            })();
        }
    }
}
```

```
14
15 const o = {
16     f(w) {
17         /* function body */
18     }
19 };
20
21 instrument(o, console.log);
22
23 o.f("hello");
```

Listing 2.1: Dynamic instrumentation in JavaScript

2.2. Static code injection

A less limited technique is to statically rewrite the instrumented code and inject tracing code wherever it is needed. The upside of such an approach is that there are several ready-to-use frameworks. The downsides are pointed out below, for each solution separately.

2.2.1. Web Tracing Framework

One notable system is the Web Tracing Framework developed by Google [23]. The main use of this framework is to profile web applications to find performance bottlenecks. The functionality is similar to that of the *Performance* tab in Chromium developer tools.

Notwithstanding, one of its advanced features is closer to our needs. It allows the user to instrument JavaScript sources to collect execution traces and inspect them in a special application.

Having to instrument all source code is cumbersome, especially when we try to analyze the code on some arbitrary website. For this reason, the Web Tracing Framework also offers a browser extension and a proxy server that cooperate together to instrument all JavaScript code online, when it is loaded into the browser.

Unfortunately, this solution has a few deal breaking downsides:

- It logs only function entry and exit events.
- The logging format is not public.
- It is a bit dated, new JavaScript features may not be traced properly.
- Functions defined using *eval* or *Function* constructor are not traced.

2.2.2. Iroh

Iroh [20] is the most complete solution based on static code injection. Just like the Web Tracing Framework, Iroh also needs to patch the code first but its capabilities go well beyond what the previous solution offers. It allows the user to register arbitrary callbacks to practically any element of JavaScript's Abstract Syntax Tree (AST). It means that this tool is able to instrument *if* statements. This use case is even included in the official examples.

Unfortunately, the framework does not offer a proxy server that could instrument code loaded into browser on the fly. There is also another, more general concern – the performance of such solution may not be acceptable.

2.3. Engine instrumentation

The last option is to modify the JavaScript engine itself to produce execution traces. The most striking benefit is that the engine already has all required info and the solution does not require the analyzed code to be modified. Another advantage is the performance. Implementing tracing code directly in the engine means that there is less overhead. The code does not need to be interpreted by the engine, it is native code that is called from JavaScript.

Unfortunately, such an instrumentation has to be written almost from scratch. Nevertheless, due to its flexibility and performance advantages, this solution has been chosen for this implementation.

The same choice has been made by Zhu et al. [47] for their implementation, but they did not share their code.

More details on how to instrument the engine can be found in Chapter 3.

Chapter 3

Trace collection by V8 instrumentation

In this chapter we give an overview of the architecture of the V8 engine and explain how we modify it to obtain execution traces.

3.1. V8 architecture

Most modern browsers do not implement a JavaScript interpreter directly. Rather, they utilize a more specialized program called a JavaScript engine, which they usually embed.

V8 is an engine used by the most popular browser, Chrome [43], which in June 2019 had over 80% market share [44].

Let us start with introducing some V8-specific glossary [41]:

- Isolate – an instance of V8. There can be more than one Isolate used b a single process of the embedding application.
- Context – a concept of a global variable scope in V8. Each Context has its own global variables and prototype chain. Each iframe has its own separate Context. There can be multiple Contexts in one Isolate but due to site isolation (see Section 3.2), each iframe is run in another process and has its own Isolate.

V8 processes JavaScript code in several steps. In this thesis we focus on steps directly related to trace collection implementation.

In short, JavaScript code is first parsed into an AST which contains source map information. In the next step, V8 traverses the entire AST and emits bytecode for each node. The bytecode is V8-specific and reflects the architecture of the V8's abstract machine. More on that can be found in Section 3.1.1.

It is worth noting that while user-defined functions are translated to bytecode, most built-in functions are implemented in a different way. We take a closer look at them in Section 3.1.2.

Only after the code is translated into bytecode, it is finally executed. At this stage there are two kinds of functions. First – those defined in JavaScript, represented in bytecode, second – built-ins defined in other ways and already compiled into native code. This distinction is not important to the user as these functions do not differ in JavaScript, i.e., there is are no syntactic differences and they are called in exactly the same way. It becomes important, though, when we try to add instrumentation code.

At some point during the execution, functions that are called very often with the same argument types, can be compiled into native code by its TurboFan Just-In-Time compiler [11]. If later the same function is called with some argument of some other type, it gets deoptimized into bytecode.

The engine's architecture is focused on achieving superior performance, while conforming to all standards without jeopardizing security.

3.1.1. JavaScript bytecode

The V8's interpreter, Ignition, is a register machine with an accumulator register [21]. While all other registers have to be specified when used as arguments, accumulating register is implicit, it is not specified by bytecode fragments that use it.

This section is supposed to be only a shallow dive into V8's bytecode. We take a look at one simple example to be able to understand what is going on in Section 3.5.

Let us have a look at a simple JavaScript function and see the bytecode produced by Ignition [1]. Listing 3.1 shows a naive implementation of a function calculating the factorial. It includes a function call (line 5) because the interpreter is lazy and otherwise the function would not be compiled. Bytecode generated by the V8's interpreter for this function is presented in Listing 3.2.

```
1 function factorial(n) {
2     return n <= 1 ? 1 : n * factorial(n - 1);
3 }
4
5 console.log(factorial(3));
```

Listing 3.1: Calculating factorial in JavaScript

```
1  [generated bytecode for function: factorial]
2  Parameter count 2
3  Register count 3
4  Frame size 24
5     18 E> 0x167d37c1eb2a @    0 : a5                StackCheck
6     28 S> 0x167d37c1eb2b @    1 : 0c 01             LdaSmi [1]
7     37 E> 0x167d37c1eb2d @    3 : 6b 02 00          TestLessThanOrEqual a0,
      [0]
8           0x167d37c1eb30 @    6 : 99 06             JumpIfFalse [6] (0
      x167d37c1eb36 @ 12)
9           0x167d37c1eb32 @    8 : 0c 01             LdaSmi [1]
10          0x167d37c1eb34 @   10 : 8b 15             Jump [21] (0x167d37c1eb49
      @ 31)
11    48 E> 0x167d37c1eb36 @   12 : 13 00 02          LdaGlobal [0], [2]
12          0x167d37c1eb39 @   15 : 26 fa             Star r1
13          0x167d37c1eb3b @   17 : 25 02             Ldar a0
14    64 E> 0x167d37c1eb3d @   19 : 41 01 04          SubSmi [1], [4]
15          0x167d37c1eb40 @   22 : 26 f9             Star r2
16    52 E> 0x167d37c1eb42 @   24 : 5d fa f9 05       CallUndefinedReceiver1 r1,
       r2, [5]
17    50 E> 0x167d37c1eb46 @   28 : 36 02 01          Mul a0, [1]
18    69 S> 0x167d37c1eb49 @   31 : a9                Return
19 Constant pool (size = 1)
20 0x167d37c1ead9: [FixedArray] in OldSpace
21  - map: 0x0a0b39f807b1 <Map>
22  - length: 1
23            0: 0x167d37c1e741 <String[#9]: factorial>
```

[1] V8 prints out bytecode when the *--print-bytecode* flag is enabled

```
24 Handler Table (size = 0)
```

Listing 3.2: The Ignition bytecode for a *factorial* function

The listing starts with the parameter count, followed by the register count and the frame size. The first one may be baffling at first since *factorial* takes only one number as an argument. We have to remember, though, that all JavaScript functions also take an implicit argument – *this*.

After the header, the actual bytecode is listed. The first number and the letter, e.g., "18 E" in line 5, identifies an expression (E) or a statement (S) from the source file. The number is an offset in characters. It is followed by the code's address in memory, the offset (in bytes) from the function start and the bytecode itself in the hexadecimal form. All of them are of secondary importance. The most useful parts are the codes in a human-readable form at the end of each line.

Once we recall that most codes use the accumulating register, the code becomes relatively self-explanatory. To make things easier, the use of the accumulating register is reflected in the code's name, e.g., `LdaConstant [0]` loads constant numbered 0 to the accumulating register.

We should now be ready to interpret each line of the *factorial*'s bytecode. Upon the function entry (line 5) the validity of the stack is checked. Later, integer 1 is loaded into the accumulating register. Next, the argument numbered 0 (symbol `a0`), which happens to be n, is tested to be less than or equal to a value stored in the accumulating register (1). If not, the jump to line 11 is performed. If the conditional was true, integer 1 is loaded into the accumulating register, then the control jumps to the last instruction which returns from the function. The return value is always stored in the accumulating register, so 1 is returned. If the conditional was true, and we are at line 11, constant 0 is loaded, which happens to be the name of our function. Later, that name is stored in the `r1` register, the argument 0 is loaded into the accumulating register, 1 is subtracted, and the result is stored in the `r2` register. Next, a function with a name stored in `r1` (*factorial*) is called with a value stored in the `r2` register ($n-1$). The result of the call, stored in the accumulating register, is then multiplied by the first argument (n). The result of the multiplication is already in the accumulating register and the function can now return.

3.1.2. JavaScript built-in functions

According to a V8's documentation post [40], JavaScript built-in functions can be implemented in three different ways. They can be written in JavaScript directly, implemented in C++ (runtime functions), or defined using an abstraction called *CodeStubAssembler* (CSA). The post, however, is slighlty dated. Since then, a new abstraction, Torque [42], has been introduced.

It is not crucial to know the details of CSA or Torque. The important takeaway is that the majority of the built-in functions are implemented in a different way than the user-defined ones.

3.2. V8 usage in Chromium

Today's usage of V8 in Chromium is determined in large part by security concerns [10]. For us, the most important decision was made after a discovery of Spectre [29] and Meltdown [30] vulnerabilities. To increase the protection against attacks based on these two vulnerabilities, Chrome's team decided to make Site Isolation enabled by default. [16]

Each website can have multiple iframes – the default one and some embedded ones. All of them are run in separate processes and, as a consequence, have their own instances (Isolates) of V8. The only case when there are multiple Isolates per process is when we use Web Workers. Each worker has its own Isolate, but they are all run within the same process.

3.3. Chrome's extensions architecture

In the current model, Chrome's extensions may consist of the following components [15]:

- Manifest – a file describing an extension, listing all its files and capabilities.
- UI Elements – code adding an extension's user interface.
- Options Page – a page allowing the user to customize the extension.
- Background script – a file with callbacks for browser events. It is executed only when an event with a registered callback occurs. It runs in its own Context, in a separate process.
- Content script – extension's code that is run in the page's Context. This code can read and modify the DOM of the website and communicate with the parent extension via messages or storage. It can also directly access a limited subset of Chrome's APIs, mostly the ones needed to communicate with the parent extension [14].

3.4. V8's *--trace* flag

Usually, the easiest way to implement some new functionality is to find code that provides a similar functionality and extend it. In case of tracing, such a base is provided by the V8's *--trace* flag.

First, we inspect what this flag can do. The reuse the example from Section 3.1.1 (Listing 3.1). Listing 3.3 shows the console output of V8 with the *--trace* flag enabled. The output is well-formatted and self-explanatory. Unfortunately, it has some shortcomings. First, there is no source map information. Second, due to the nature of JavaScript, function traces can get intertwined (as mentioned in Section 1.6). Lastly, since stack information is limited to just the stack depth, it may be impossible to untangle events in some cases.

```
1   1: ~(this=0x1bd8e6501521 <JSGlobal Object>) {
2   2:  ~factorial(this=0x1bd8e6501521 <JSGlobal Object>, 3) {
3   3:   ~factorial(this=0x1bd8e6501521 <JSGlobal Object>, 2) {
4   4:    ~factorial(this=0x1bd8e6501521 <JSGlobal Object>, 1) {
5   4:    } -> 1
6   3:   } -> 2
7   2:  } -> 6
8 6
9   1: } -> 0x1bc7923804d1 <undefined>
```

Listing 3.3: V8's output for a *factorial* function with the *--trace* flag enabled

Nevertheless, this flag is a good starting point. Let us have a deeper dive into how it works.

We have already seen the bytecode for *factorial* in Listing 3.2. Listing 3.4 shows the bytecode for the same function when the *--trace* flag is enabled. There are two differences compared to the bytecode produced without the tracing flag. The first – there is a call to

a runtime function *TraceEnter*, before *StackCheck*, right after the function is entered. The second – just before returning, the result is saved to the `r0` register and a runtime function *TraceExit* is called with the return value as its argument. *TraceExit* stores its argument back in the accumulating register so there is no change in semantics of the inspected code.

```
[generated bytecode for function: factorial]
Parameter count 2
Register count 3
Frame size 24
         0x315c6429eb32 @    0 : 61 a7 01 fb 00    CallRuntime [TraceEnter], r0-r0
   18 E> 0x315c6429eb37 @    5 : a5                StackCheck
   28 S> 0x315c6429eb38 @    6 : 0c 01             LdaSmi [1]
   37 E> 0x315c6429eb3a @    8 : 6b 02 00          TestLessThanOrEqual a0, [0]
         0x315c6429eb3d @   11 : 99 06             JumpIfFalse [6] (0x315c6429eb43 @ 17)
         0x315c6429eb3f @   13 : 0c 01             LdaSmi [1]
         0x315c6429eb41 @   15 : 8b 15             Jump [21] (0x315c6429eb56 @ 36)
   48 E> 0x315c6429eb43 @   17 : 13 00 02          LdaGlobal [0], [2]
         0x315c6429eb46 @   20 : 26 fa             Star r1
         0x315c6429eb48 @   22 : 25 02             Ldar a0
   64 E> 0x315c6429eb4a @   24 : 41 01 04          SubSmi [1], [4]
         0x315c6429eb4d @   27 : 26 f9             Star r2
   52 E> 0x315c6429eb4f @   29 : 5d fa f9 05       CallUndefinedReceiver1 r1, r2, [5]
   50 E> 0x315c6429eb53 @   33 : 36 02 01          Mul a0, [1]
         0x315c6429eb56 @   36 : 26 fb             Star r0
         0x315c6429eb58 @   38 : 61 a8 01 fb 01    CallRuntime [TraceExit], r0-r0
   69 S> 0x315c6429eb5d @   43 : a9                Return
Constant pool (size = 1)
0x315c6429eae1: [FixedArray] in OldSpace
 - map: 0x3806c0d807b1 <Map>
 - length: 1
           0: 0x315c6429e741 <String[#9]: factorial>
Handler Table (size = 0)
```

Listing 3.4: The Ignition bytecode for a *factorial* function with the *--trace* flag enabled

3.5. Bytecode injection

Finally, we have come to the gist of the current chapter – tracing implementation. Once we have seen how the *--trace* flag works, we can improve it to our needs.

The entire implementation requires a few steps:

1. Adding two new flags – one for turning on our tracing (*--trace-dea*), and another one for specifying the file with tracing information.
2. Preparing a function that prints out the entire stack with the source map information.
3. Preparing a set of new runtime functions that log the runtime events with their locations and call stacks (using the function from Step 2). Each type of event we want to log has its own function. These functions are called directly from the bytecode (Step 4).
4. Injecting calls to the runtime functions from Step 3 in the appropriate places.

We will not delve too deeply into how to implement each part. The first and third points are pretty straightforward. Both of them just require adding declarations to special header files.

The second point seems the hardest but luckily there is a similar function in the Chromium codebase. It is worth noting that it is possible to recover source map information during the runtime and have the entries contain the precise line and column information. However, it turned out to be too slow for use in Chromium, as there was a noticeable slowdown. Almost no web page could finish loading in reasonable time. The reason for such poor performance is that those locations are not stored directly in the AST. Rather, only offset in characters is stored. To recover line and column information it is necessary to first go through a few levels of abstraction to get the source code and then calculate the coordinates by traversing it. As a consequence, only character offset is logged. It is always possible to recover the more friendly line, column location later, so it is not that big of a deal.

The challenge of the fourth part is to have the right offsets of statements/expressions. It is the easiest with functions. Objects representing functions during runtime hold their location. In case of statements it is harder as their locations are available only during the parsing and the bytecode generation stages. To have these offsets accessible during the runtime, we store them as code constants and pass as arguments to the runtime functions that log the code events.

Listing 3.5 shows bytecode generated by Ignition with our tracing flag enabled. Similarly to the original tracing, there are calls to the runtime functions right after entering (line 5) and just before returning (lines 23-24). The new part is that also the information which branch was taken is logged. In lines 7-8 the offset information is stored in the `r0` register which is later used by the runtime functions that perform the actual logging (lines 12, 15).

```
1  [generated bytecode for function: factorial]
2  Parameter count 2
3  Register count 4
4  Frame size 32
5          0x3dd338d9eb3a @    0 : 61 a9 01 fb 00    CallRuntime [TraceDeaEnter
       ], r0-r0
6     18 E> 0x3dd338d9eb3f @    5 : a5                StackCheck
7     28 S> 0x3dd338d9eb40 @    6 : 12 00             LdaConstant [0]
8          0x3dd338d9eb42 @    8 : 26 fb             Star r0
9          0x3dd338d9eb44 @   10 : 0c 01             LdaSmi [1]
10    37 E> 0x3dd338d9eb46 @   12 : 6b 02 00          TestLessThanOrEqual a0,
       [0]
11         0x3dd338d9eb49 @   15 : 99 0b             JumpIfFalse [11] (0
       x3dd338d9eb54 @ 26)
12         0x3dd338d9eb4b @   17 : 61 af 01 fb 01    CallRuntime [
       TraceDeaIfStmtThen], r0-r0
13         0x3dd338d9eb50 @   22 : 0c 01             LdaSmi [1]
14         0x3dd338d9eb52 @   24 : 8b 1a             Jump [26] (0x3dd338d9eb6c
       @ 50)
15         0x3dd338d9eb54 @   26 : 61 b0 01 fb 01    CallRuntime [
       TraceDeaIfStmtElse], r0-r0
16    48 E> 0x3dd338d9eb59 @   31 : 13 01 02          LdaGlobal [1], [2]
17         0x3dd338d9eb5c @   34 : 26 f9             Star r2
18         0x3dd338d9eb5e @   36 : 25 02             Ldar a0
19    64 E> 0x3dd338d9eb60 @   38 : 41 01 04          SubSmi [1], [4]
20         0x3dd338d9eb63 @   41 : 26 f8             Star r3
21    52 E> 0x3dd338d9eb65 @   43 : 5d f9 f8 05       CallUndefinedReceiver1 r2,
        r3, [5]
22    50 E> 0x3dd338d9eb69 @   47 : 36 02 01          Mul a0, [1]
23         0x3dd338d9eb6c @   50 : 26 fb             Star r0
```

```
         0x3dd338d9eb6e @   52 : 61 aa 01 fb 01    CallRuntime [TraceDeaExit
   ], r0-r0
   69 S> 0x3dd338d9eb73 @   57 : a9                 Return
Constant pool (size = 2)
0x3dd338d9eae1: [FixedArray] in OldSpace
 - map: 0x0f45366007b1 <Map>
 - length: 2
           0: 35
           1: 0x3dd338d9e741 <String[#9]: factorial>
Handler Table (size = 0)
```

Listing 3.5: The Ignition bytecode for the *factorial* function with the *--trace-dea* flag enabled

Listing 3.6 shows the output produced with the new flag enabled. Each execution event entry consists of an event type, location (optional, depends on the event type) and a full call stack. Events returning a value also log that value, but it is not used later in the analysis. The call stack is represented as a list of locations. Each location consists of a function name, a file of origin, and an offset in the file (in characters). The integer -1 that appears after the offset is a remnant of the implementation that recovers the full position information (line, column). The next part of the pipeline expects two numbers here. This way it is easy to turn on full position recovery and have the pipeline still working.

In the current implementation the following execution events are logged:

- Function enter and exit.
- Generator enter, suspend, yield (exit is indistinguishable from a normal function exit).
- "then"/"else" branches in conditional statements.
- Truthy/falsy values in ternary expressions (the same types of events as in the "if" statement branches).

It is easy to extend the implementation to also log execution events associated with loops, but there is a trade-off between the coverage and the size of the log files created, so it was left out.

```
FUNCTION ENTER
   0: ~ at factorial.js @@ 0,-1 ()
=
--
FUNCTION ENTER
   0: ~factorial at factorial.js @@ 18,-1 ()
   1: ~ at factorial.js @@ 0,-1 ()
=
--
IF STMT - ELSE
   2: ~factorial at factorial.js @@ 35,-1
   0: ~factorial at factorial.js @@ 18,-1 ()
   1: ~ at factorial.js @@ 0,-1 ()
=
--
FUNCTION ENTER
   0: ~factorial at factorial.js @@ 18,-1 ()
   1: ~factorial at factorial.js @@ 18,-1 ()
   2: ~ at factorial.js @@ 0,-1 ()
=
--
IF STMT - ELSE
```

```
   3: ~factorial at factorial.js @@ 35,-1
   0: ~factorial at factorial.js @@ 18,-1 ()
   1: ~factorial at factorial.js @@ 18,-1 ()
   2: ~ at factorial.js @@ 0,-1 ()
=
--
FUNCTION ENTER
   0: ~factorial at factorial.js @@ 18,-1 ()
   1: ~factorial at factorial.js @@ 18,-1 ()
   2: ~factorial at factorial.js @@ 18,-1 ()
   3: ~ at factorial.js @@ 0,-1 ()
=
--
IF STMT - THEN
   4: ~factorial at factorial.js @@ 35,-1
   0: ~factorial at factorial.js @@ 18,-1 ()
   1: ~factorial at factorial.js @@ 18,-1 ()
   2: ~factorial at factorial.js @@ 18,-1 ()
   3: ~ at factorial.js @@ 0,-1 ()
=
--
FUNCTION EXIT
   0: ~factorial at factorial.js @@ 18,-1 ()
   1: ~factorial at factorial.js @@ 18,-1 ()
   2: ~factorial at factorial.js @@ 18,-1 ()
   3: ~ at factorial.js @@ 0,-1 ()
-> 1
=
--
FUNCTION EXIT
   0: ~factorial at factorial.js @@ 18,-1 ()
   1: ~factorial at factorial.js @@ 18,-1 ()
   2: ~ at factorial.js @@ 0,-1 ()
-> 2
=
--
FUNCTION EXIT
   0: ~factorial at factorial.js @@ 18,-1 ()
   1: ~ at factorial.js @@ 0,-1 ()
-> 6
=
--
6
FUNCTION EXIT
   0: ~ at factorial.js @@ 0,-1 ()
-> 0x0da3956804d1 <undefined>
=
--
```

Listing 3.6: V8's output for a *factorial* function with the *--trace-dea* flag enabled

A careful reader might have noticed that in Listing 3.1 in the last line there is a call to *factorial* and the result is passed to *console.log*, but only the former call is logged. The reason for this is that the implemented solution works only for functions defined in JavaScript (it is also true for the default *--trace* flag). Functions defined in other ways (see Section 3.1.2) are not logged. It is certainly possible to add logging to each one of them by modifying their code or by modifying CSA and Torque compilers but it has not been done here. The amount of work required to instrument also these functions seems disproportionate to the potential improvements. Another justification is that they are black boxes anyway, there is

no JavaScript code corresponding to them and we cannot see branch divergences happening inside them.

The last obstacle worth noting is the Chrome's process separation. As explained in Section 3.2, there are multiple instances of V8 running at the same time, in different processes. When some flag is passed to Chrome, all the instances see the same value. Therefore, if some trace output file is passed, all processes will write to the same file, possibly resulting in interlacing and an unusable output. An easy workaround is to create a new file for each Isolate and later just select only the interesting one (the one that corresponds to the analyzed website). There is always at most one such file (theoretically there could be none as a website can contain no JavaScript code) and it is easy to select it by filtering (using the "grep" tool) by source location. All other files can be deleted.

A V8 Isolate is generally pretty oblivious to what code it runs. Website code is the same to V8 as extension code. After all, the environment (Web API, DOM, etc.) is provided by the browser. Nevertheless, it is preferable not to log adblocker events. Such extensions have really long lists of filters and their initialization takes 1-2 seconds when the browser is not producing execution traces. When it does, the initialization takes more than half a minute on a slow computer and the trace itself can easily grow to several gigabytes. Fortunately, we can resort to a trick: inspect the logged event and stop tracing in a given Isolate when some extension-specific file has been encountered. As a result, extension code is not traced and it has no noticeable performance impact.

3.6. Controlling Chrome programmatically

Ad-blocking extensions need a second or two to initialize their code. Even in a stock browser, opening a website immediately after the browser starts can result in ads not being blocked. As a consequence, simply passing the website address through console is not enough when we want to collect traces automatically. The extension will simply not be fully initialized and advertisements will not be blocked. A more sophisticated solution is needed.

Luckily, there is a framework for building end-to-end tests – Selenium [9]. It has binding in all major languages, in our case we use Python.

Selenium is capable of performing the same interactions with the website that user can do. It communicates directly with a specialized WebDriver, different for each browser. In case of Chrome it is called ChromeDriver. A WebDriver issues commands to the browser through the debugging interface which is a special port with a well-defined communication protocol. That is all we need to know about it, as we are not using any of its sophisticated features, just connecting to the browser, opening a website and closing it after some specified time.

Chapter 4

Trace analysis

In this chapter we describe how we process the execution traces to find execution differences and, in turn, detect anti-adblockers.

4.1. Parsing

Although the format presented in Chapter 3 is really simple, parsing it can prove challenging. The sole reason is the size of the files. It is really common for websites to produce files of size in the range of a few gigabytes. Some can even output as much as 48 GB in about 2 minutes!

The analyzing program was written in Haskell [24]. Haskell is a purely functional, lazy, statically typed language. It has been chosen because it is relatively easy to write parsers in a language like this. Further, static typing with pattern matching proves convenient when manipulating well-structured data like log entries. Last but not least, automatically derived instances [1] help avoid writing tedious code and focus on implementing non-trivial parts.

All that being said, Haskell is a bit like C++ – inexperienced user can easily make mistakes that render the code slow and memory-greedy.

The logging format is described in Section 3.5. Each entry consist of an event type and several locations. Each location comprises a function name, a source file and a position within the file.

At first, an internal format for the event was exactly the same. One object consisted of two strings and two numbers (it could be one number, but this way line and column info logging can be turned on any time).

The first mistake, specific to Haskell, was the use of the `String` type to represent the function and the file names. The use of the default `String` to represent textual data makes the code inefficient because it is a linked lists of `Char`s, i.e. to store one character, 9 bytes of memory are used. This representation is so inefficient that it is not even worth trying to profile it.

That error was fixed by changing the representation to the `ByteString` type from the *bytesting* package [19]. `ByteString`s are represented internally as real byte arrays, not linked lists. This change was accompanied by a rewrite of the parsing code to *attoparsec* [12] which can operate on input `ByteString`s.

The code was much less memory-consuming and faster but it still seemed to consume too much memory. Profiling proved that suspicions were warranted. The profiling diagram is shown on Figure 4.1. All the profiling diagrams in this section were collected by running the

[1] Without going into details, type classes in Haskell are a bit like interfaces in object-oriented languages, e.g., the `Ord` class defines objects that can be ordered

analyzing program on the same set of 6 input files. The first three files occupy 48MB each, the last three – 4MB each. All the files combined occupy around 150MB of space, while the peak memory usage of the program was above 900MB of memory.

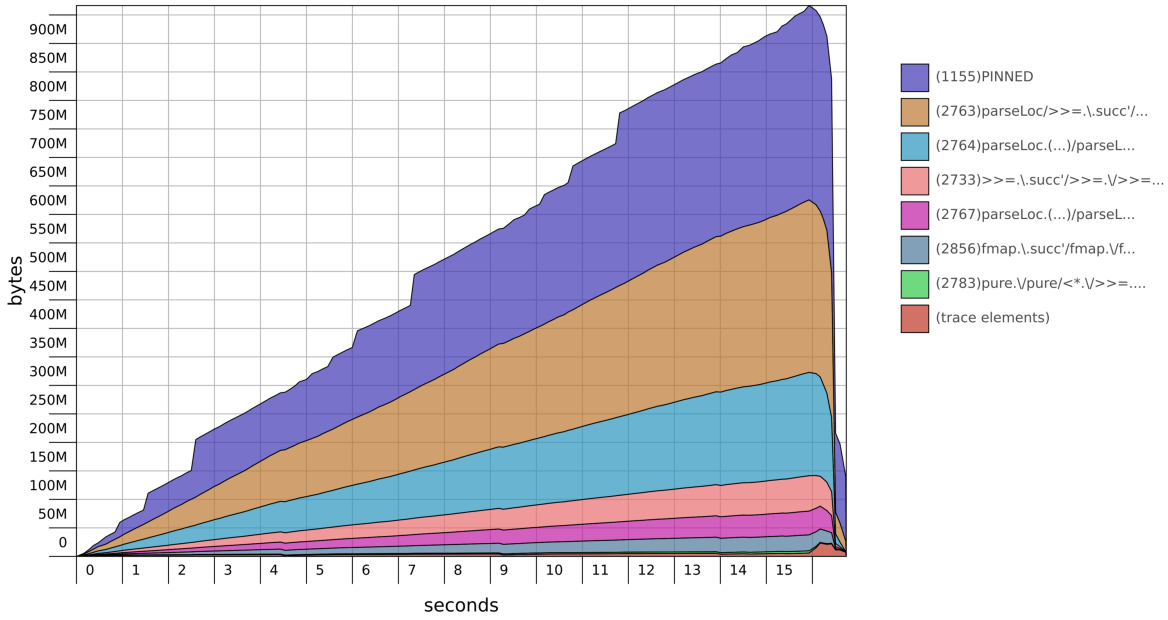

Figure 4.1: Memory usage of the first implementation using the `ByteString` type as the internal strings representation

Pinned memory seemed to be never freed during the execution. Also, traces seemed to reside in memory for too long and occupy too much space.

The first problem was a reflection of how Haskell keeps `ByteString`s. They are stored in pinned memory, which is kept on a heap but not managed by a garbage collector [31] and cannot be moved around. As a result, it leads to fragmentation of the heap and it cannot be effectively managed. The fix turned out to be simple. There is a more suitable storage format – `ShortByteString` (also part of the *bytestring* package). Objects of this type are managed by the GC and stored just like usual heap objects, which can be moved around and easily garbage collected.

The second problem was harder to track down. This time the cause was the laziness. Laziness can improve the performance when some objects are never used and the language never needs to fully compute them. However, when we know that all objects will eventually be used, it is more efficient to calculate them as soon as possible. The enforcement of eager evaluation seems to be an obscure feature, but it can significantly reduce memory consumption in certain situations. The improvement was visible (cf. Figure 4.2) but it was not the end, as the peak memory usage of almost 240MB was notably larger than the total trace files size.

It seemed suspicious that the entire file had to be kept in memory to be parsed (pinned memory on Figure 4.2 represents input files read into memory). After all, if the parser was written in C++, probably one line at a time would be read and processed. So why did the parser in Haskell consume so much memory?

This is what *attoparsec* documentation states about the incremental input:

> Note: incremental input does not imply that attoparsec will release portions of

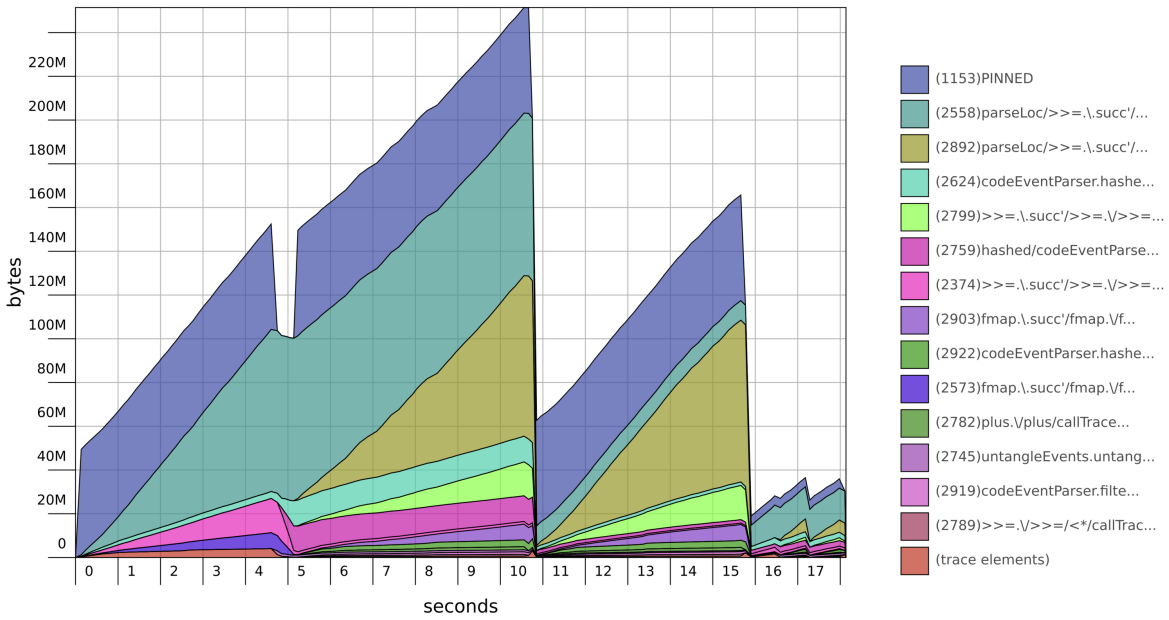

Figure 4.2: Memory usage after switching to the `ShortByteString` type and forcing the eager evaluation

> its internal state for garbage collection as it proceeds. Its internal representation is equivalent to a single ByteString: if you feed incremental input to a parser, it will require memory proportional to the amount of input you supply. (This is necessary to support arbitrary backtracking.)

So, our parser kept the whole file contents in memory to be able to backtrack. But it was not necessary with such a simple format. Fortunately, it was possible to alter this behaviour. At that time, the parser tried to parse the entire file into a list of events. To make sure that it can backtrack, it kept the whole input in memory. Instead, we asked the parser to parse only one event. It did it and returned a part of the input that was not parsed yet. We repeated that in a loop and the parser never retained more than just a few kilobytes of memory. Figure 4.3 shows the improvement.

Last but not least, the internal representation of the event could be greatly improved. Locations consist of file names and function names. But both sets are limited! Usually there are only a few sources containing just a few hundreds of unique functions. We can create a map of all source and function names and just keep the appropriate identifiers in location objects. Just 4 numbers instead of 2 strings and 2 numbers!

This representation also has another major benefit – comparisons of such objects are much faster.

The final memory consumption is presented on Figure 4.4.

4.2. Trace untangling

Due to the nature of the JavaScript execution model (see Section 1.6), execution events corresponding to different JavaScript events or functions can be intertwined.

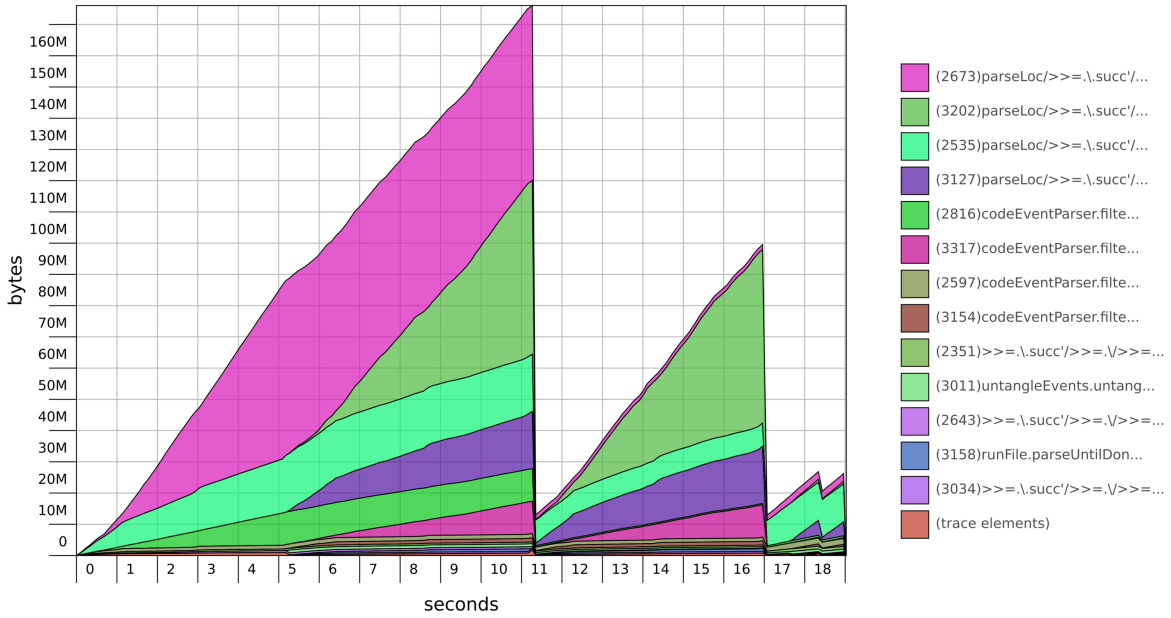

Figure 4.3: Memory usage after preventing the parser from keeping the entire input file in memory

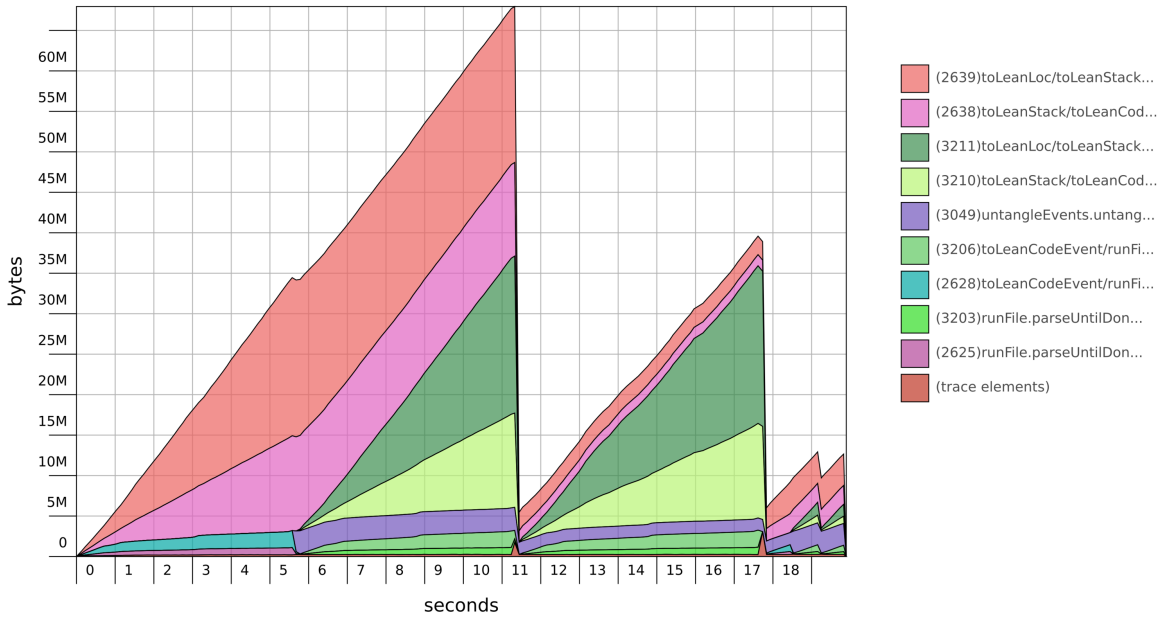

Figure 4.4: Final memory usage of the analyzing program

For this reason, all events forming a trace have to be untangled into subtraces, each corresponding to one script or callback.

Let us recall that all events are logged by our instrumented Chromium with their call stacks. Now, we have three cases:

- The event is a function entry with an empty call stack – it starts a new subtrace.

- The event is a function exit removing the last item from the stack – it ends a subtrace.
- Any other event – it is a continuation of a subtrace whose last event has the same call stack.

Naturally, some care has to be taken to ensure that the right stack is compared. Some events change the call stack, e.g., a function entry and exit, some do not, e.g. "if statement – then". Here, the stack from the moment just after the event occurred is saved and it is appropriately modified (the stack can grow or shrink by exactly one location or remain unchanged) when comparing with the past events' stacks.

Listing 4.1 presents the pseudocode of the algorithm.

```
def untangleEvents(events):
    closedTraces := ∅
    openTraces := ∅

    for event in events:
        if event.type == FunctionEntry and |event.stack| == 1:
            openTraces.insert([event])
        else:
            matchingSubtrace := findMatchingSubtrace(event, openTraces)
            openTraces.remove(matchingSubtrace)
            matchingSubtrace.append(event)

            if event.type == FunctionExit and |event.stack| == 0:
                closedTraces.insert(matchingSubtrace)
            else:
                openTraces.insert(matchingSubtrace)

    return closedTraces

def findMatchingSubtrace(ev, traces):
    st := case
        ev.type in [FunctionEnter, GeneratorEnter] → tail ev.stack
        ev.type in [FunctionExit, GeneratorSuspend] → (ev.loc : ev.stack)
        otherwise → ev.stack

    return trace whose last event's stack equals st
```

Listing 4.1: Pseudocode of the trace untangling algorithm

The above code is rather uncomplicated, but it is slow when implemented naively. The critical part is finding the trace with the right stack in *findMatchingSubtrace*. To make it fast, two things were done:

- The location objects are just 4 numbers instead of 2 strings and 2 numbers. This makes stack comparisons one or two orders of magnitude faster.
- The open traces are kept in a map indexed by a stack of the last event. The lookup cost becomes logarithmic instead of linear. This optimization is especially important when there are lots of intertwined subtraces. It should also be noted that this optimization alone would not help much if stacks comparisons were slow.

One last caveat – it is possible to enter a function in JavaScript and never return from it. When an exception is thrown and there is no catch block inside a function, the error will be propagated higher and possibly multiple functions can be left without any of them returning any value. The throw events are not logged, therefore this situation has to be taken care of

in the analyzing program. The solution is the following: when there is no open subtrace with a matching stack, we check if some stack matches if we remove some elements from its top. If so, then we assume that it is due to the try-catch block.

4.3. Trace alignment

Trace alignment is a technique of identifying execution differences between two runs of the same program. To be able to align two traces, events in both have to be assigned execution indices. In our case the execution index is the event type, its location, and the stack at the time of its occurrence. In this thesis the term "execution index" is often used interchangeably with the execution event as everything that is logged when an event occurs is the index.

The algorithm used in this implementation is based on the work by Johnson et al. [27] The result of the trace alignment is an execution trace diff. It is a list of events of three kinds:

- Common – an event that occurred in both traces.
- Left – an event that occurred only in the left-hand trace.
- Right – an event that occurred only in the right-hand trace.

Each diff event includes the associated execution event. Listing 4.2 present the pseudocode of the algorithm.

```
def alignTraces(leftTrace, rightTrace):
    results := []
    score := 0

    if leftTrace.size() == 0 || rightTrace.size() == 0
        || leftTrace[0] != rightTrace[0]:
        return (-1, [])

    while leftTrace.size() > 0 || rightTrace.size() > 0:
        if leftTrace.size() == 0:
            results.append(map(Right, rightTrace))
            rightTrace := []
        elif rightTrace.size() == 0:
            results.append(map(Left, leftTrace))
            leftTrace := []
        else:
            (leftEvent:leftTail) := leftTrace
            (rightEvent:rightTrail) := rightTrace

            if leftEvent == rightEvent:
                score := score + 1
                results.append(Common(leftEvent))
                leftTrace := leftTail
                rightTrace := rightTail
            elif leftEvent.stack.size() <= rightEvent.stack.size():
                result.append(Right(rightEvent))
                rightTrace := rightTail
            else:
                result.append(Left(leftEvent))
                leftTrace := leftTail

    return (score, results)
```

Listing 4.2: Pseudocode of the trace alignment algorithm

The algorithm output includes a score, which is a number of common events. This concept is present in the original article and its purpose is explained in Section 4.4.

The algorithm is rather simple. We assume that matching traces have to start with the same event. If they do not (lines 5-6), we just terminate and return a score of -1, which signifies that the traces do not match.

Apart from the case when one of the traces ends, there are only two cases:

- Both unprocessed traces start with the same event (this implies that stacks are equal) – line 20 – in this case we add an event of type Common and remove one event from both lists
- The top events do not match (lines 25 and 28) – in that case the event with larger stack is processed first. The reasoning is the following. The bottom of the larger stack is the same as the shorter stack. If we first process the shorter stack and if it shrinks, we lose some common events. It we process the larger stack first, the shorter one is "frozen". The larger stack may grow, but it doesn't matter since it is already divergent. When it shrinks to the size of the shorter stack, it is a re-convergence point.

4.4. Trace matching using the Stable Marriage Problem

We already know how to align two traces (Section 4.3), but we have not answered a different question – which pair of subtraces should we align?

This thesis presents a novel approach which uses the Stable Marriage Problem (SMP) to answer this question. First, we define what the SMP is and later explain how it is adapted to our needs.

The original statement of the problem is the following: given equally sized sets of men and women and their matrimonial preferences, find a stable matching, i.e. matching in which there exists no pair of a man and a woman in which they both would have better partner than the currently assigned one. The problem can be solved using the Gale-Shapley algorithm [22]. Listing 4.3 presents its pseudocode.

```
1  def stableMatching(men, women):
2      for m in men:
3          m.mathing = free
4      for w in women:
5          w.matching = free
6
7      while there exists a free man m who still has a women to propose to:
8          w := first women on m list that he has not yet proposed to
9          if w.matching == free:
10             m.matching := w
11             w.matching := m
12         else:
13             m' := w.matching // current provisional partner of w
14             if w.preferences[m] > w.preferences[m']:
15                 m'.matching = free
16                 m.matching = w
17                 w.matching = m
18             else:
19                 pass // w and m' remain engaged
```

Listing 4.3: Pseudocode of the Gale-Shapley (deferred acceptance) algorithm

This is how the SMP is used to solve the subtrace matching problem:

- In the first phase all pairs of subtraces are aligned and similarity scores (a number of common events) are calculated – the score is the marital preference.
- In the second phase, an adapted Gale-Shapley algorithm is run and only the matched subtraces are used. Their diff is retrieved from a special map as it was calculated and stored there in the first phase.

All men have a list of women sorted by their preferences (from highest to lowest). At the beginning everyone is free. Then, in each round each free man m proposes to the first woman w he has not yet proposed to. The woman can reply "maybe" if she is free or is engaged to m' and prefers m over m'. Otherwise, she rejects. So women only "trade up" and men propose whenever they are free to the best possible partner. In the end everybody is engaged and the marriages are stable.

The time complexity of this algorithm is $O(n^2)$, where n is the number of men or women. There is at most n rounds (some man proposes to every woman) and in each round there are at most n proposals made.

In the original problem statement the number of women is equal to the number of men and everybody gets engaged in the end. In our case the number of men and women does not have to the same. We assume that the more numerous group has the role of women. We allow some pairs of men and women to not match each other (the preference score equal to -1), therefore in the end some people may not be engaged.

Let us see what happens with these two adjustments and the same procedure as in the original version:

- The men should still propose in the order from the highest preference to the lowest. All men will still get the best possible match this way, because all women that rejected must have had some better options.
- The women should accept anybody when they do not have any tentative match and trade up later. If there is a man that a woman prefers over the current match and he did not propose, he must have had someone who he preferred more and who accepted.

Consequently, the algorithm is almost the same. The only exception is that men do not propose to women with score -1. Also, the result is different – in the end there usually will be many people without a pair. However, those who are paired form stable marriages.

Such a usage of the Deferred Acceptance algorithm allows us to form the best pairs of subtraces to align. There is no point in aligning the traces that do not have the same initial event (the aligning algorithm has to start from an anchor point), so we skip the pairs with the assigned score equal to -1. Further, the best pairs are selected by the number of common events. If two traces A and B are exactly the same, the only possibility that they will not be matched is when there is some longer trace C that contains the entire A and B. But, if there is also a longer trace D that contains the entire A and B, it is very likely that the extra part will be matched with an extra part of C and, in consequence, C and D will be matched. If there is no such a trace D and C is matched with either A or B, it is still fine as we have more calls to some function in one group and we cannot be sure which subtraces truly correspond to each other.

4.5. Artificial examples

This section presents how the above algorithm works on a few simple, hand-crafted examples run in a standalone V8.

The first example is based on our already-familiar example of the *factorial* function (Listing 3.1).

First, we call the function with $n = 3$, then with $n = 4$ and compare the traces. This demonstrates how "if-then-else" divergences are detected. Listing 4.4 shows the output of the program.

We see that after 3 *factorial* calls the traces diverge and one of them takes the "else" branch and makes one more recursive call (lines 19-30). The other takes the "then" branch (lines 31-33). After that one extra recursive call both traces immediately reconverge (line 34).

```
1                       COMMON:
2  Event: FunctionEnter
3  "~" at "factorial.js" @@ 0,-1
4                       COMMON:
5  Event: FunctionEnter
6  "~factorial" at "factorial.js" @@ 18,-1
7                       COMMON:
8  Event: IfStmtElse
9  "~factorial" at "factorial.js" @@ 35,-1
10                      COMMON:
11 Event: FunctionEnter
12 "~factorial" at "factorial.js" @@ 18,-1
13                      COMMON:
14 Event: IfStmtElse
15 "~factorial" at "factorial.js" @@ 35,-1
16                      COMMON:
17 Event: FunctionEnter
18 "~factorial" at "factorial.js" @@ 18,-1
19                                      RIGHT:
20 Event: IfStmtElse
21 "~factorial" at "factorial.js" @@ 35,-1
22                                      RIGHT:
23 Event: FunctionEnter
24 "~factorial" at "factorial.js" @@ 18,-1
25                                      RIGHT:
26 Event: IfStmtThen
27 "~factorial" at "factorial.js" @@ 35,-1
28                                      RIGHT:
29 Event: FunctionExit
30 "~factorial" at "factorial.js" @@ 18,-1
31     LEFT:
32 Event: IfStmtThen
33 "~factorial" at "factorial.js" @@ 35,-1
34                      COMMON:
35 Event: FunctionExit
36 "~factorial" at "factorial.js" @@ 18,-1
37                      COMMON:
38 Event: FunctionExit
39 "~factorial" at "factorial.js" @@ 18,-1
40                      COMMON:
41 Event: FunctionExit
42 "~factorial" at "factorial.js" @@ 18,-1
43                      COMMON:
44 Event: FunctionExit
45 "~" at "factorial.js" @@ 0,-1
```

Listing 4.4: A trace diff of the *factorial* function run with $n = 3$ and $n = 4$

The next example demonstrates that sometimes it is possible to find a divergence even when the built-in functions are involved. Listing 4.5 is a short example involving *Array.prototype.map*.

In the first run the built-in function is called with an array of length 3, in the second – the array contains one item less. Listing 4.6 shows the diff. We can see that in one run *multiplyByTwo* is entered 2 times and in the other – 3 times. Naturally, if the arrays were of the same length, but with different content, this difference would not be caught this way.

```
const multiplyByTwo = (x) => 2 * x;

const nat = [...Array(2).keys()];

const even = nats.map(multiplyByTwo);
```

Listing 4.5: A simple JavaScript example of the *Array.prototype.map* usage

```
                    COMMON:
Event: FunctionEnter
"~" at "map.js" @@ 0,-1
                    COMMON:
Event: FunctionEnter
"~multiplyByTwo" at "map.js" @@ 22,-1
                    COMMON:
Event: FunctionExit
"~multiplyByTwo" at "map.js" @@ 22,-1
                    COMMON:
Event: FunctionEnter
"~multiplyByTwo" at "map.js" @@ 22,-1
                    COMMON:
Event: FunctionExit
"~multiplyByTwo" at "map.js" @@ 22,-1
                                RIGHT:
Event: FunctionEnter
"~multiplyByTwo" at "map.js" @@ 22,-1
                                RIGHT:
Event: FunctionExit
"~multiplyByTwo" at "map.js" @@ 22,-1
                    COMMON:
Event: FunctionExit
"~" at "map.js" @@ 0,-1
```

Listing 4.6: A trace diff of the *Array.prototype.map* run with arrays of length 3 and 2

The last example demonstrates the generator-related events in action. Listing 4.7 shows an example of *factorial* rewritten to use generators. It is run two times, with $n = 2$ and $n = 3$. Listing 4.8 shows the resulting diff. We can see that when the generator is created, the `FunctionEnter` event is emitted, followed by the `GeneratorSuspend` (lines 7-12). Events `GeneratorEnter` and `GeneratorYield` are used only when the *Generator.prototype.next* is called. Last but not least, the `GeneratorYield` event is always followed by `GeneratorSuspend`.

```
function *factorial(n) {
  let acc = 1;
  while (n > 1) {
    acc = acc * n;
    n = n - 1;
    yield [acc, n];
  }
}

function print_factorial(num) {
  const gen = factorial(num);

  while (true) {
```

```
    const ret = gen.next();
    if (ret.done) break;
    const [acc, n] = ret.value;
    console.log(`Acc: ${acc}, n: ${n}`);
  }
}

print_factorial(2);
```

Listing 4.7: An example of JavaScript code using generators

```
                    COMMON:
Event: FunctionEnter
"~" at "generators.js" @@ 0,-1
                    COMMON:
Event: FunctionEnter
"~print_factorial" at "generators.js" @@ 143,-1
                    COMMON:
Event: FunctionEnter
"~factorial" at "generators.js" @@ 19,-1
                    COMMON:
Event: GeneratorSuspend
"~factorial" at "generators.js" @@ 19,-1
                    COMMON:
Event: GeneratorEnter
"~factorial" at "generators.js" @@ 19,-1
                    COMMON:
Event: GeneratorYield
"~factorial" at "generators.js" @@ 96,-1
                    COMMON:
Event: GeneratorSuspend
"~factorial" at "generators.js" @@ 19,-1
                    COMMON:
Event: GeneratorEnter
"~factorial" at "generators.js" @@ 19,-1
                                   RIGHT:
Event: GeneratorYield
"~factorial" at "generators.js" @@ 96,-1
                                   RIGHT:
Event: GeneratorSuspend
"~factorial" at "generators.js" @@ 19,-1
                                   RIGHT:
Event: GeneratorEnter
"~factorial" at "generators.js" @@ 19,-1
                    COMMON:
Event: FunctionExit
"~factorial" at "generators.js" @@ 19,-1
                    COMMON:
Event: IfStmtThen
"~print_factorial" at "generators.js" @@ 231,-1
                    COMMON:
Event: FunctionExit
"~print_factorial" at "generators.js" @@ 143,-1
                    COMMON:
Event: FunctionExit
"~" at "generators.js" @@ 0,-1
```

Listing 4.8: A trace diff of the generator-using *factorial* function run with $n = 2$ and $n = 3$

4.6. Noise filtering

One of the biggest challenges in the entire method is filtering the noise. The noise on the website may come in many different flavours:

- Some content can be based on Random Number Generators.
- The content may be dynamic and different with each refresh of the page, e.g., Facebook feed.
- Code that is not related to the website, e.g., trackers, analytics.

The difficult part is to filter out as much data as possible, without removing valuable information.

In the original paper authors collect redundant traces, diff together all positive traces (as defined in Section 1.1), then also all negative traces, and generate a blacklist of execution divergences that are guaranteed to be noise.

In this thesis, we chose a different approach. The positive traces are all put together and the common subtraces are extracted. The same thing is done with the negative traces. At the end, the whole SMP and alignment analysis (see Sections 4.3 and 4.4) is run only on these common subtraces.

The effectiveness and limitations of this approach are discussed in Chapter 5.

Chapter 5

Evaluation

In this chapter we give an overview of the implemented system, evaluate it, and present the study conducted with its help.

5.1. The pipeline

The goal of this work is to automate as many steps of the anti-adblocking analysis as possible. Ideally, the system should be able to read a list of websites, collect traces, analyze each of them, and produce meaningful results.

The entire system is brought together by a program written in Python. It is controlled by an extensive set of flags and logs each step to make troubleshooting easier. The pipeline comprises the following steps:

- A website (or list of websites) to analyze is read as an argument or from a file when a special flag is provided.
- If selected, 3 positive and 3 negative traces are collected for each website (before each browser run all cookies are deleted). After each run, all unwanted traces are deleted (they may come from extensions or unwanted iframes).
- If selected, the analysis is run and results are saved in a selected location.
- Optionally, the traces are deleted.

The system counts the number of traces with execution differences. If there is at least one such a pair of traces found, we assume that it is due to the anti-adblocking activities. The output includes the diff of these traces, as well as all unmatched ones.

5.2. Methodology

The methodology is the same as in the original paper [47]. First, we test the effectiveness of the method on sets of negative and positive examples. Then, we use the system to conduct a small scale study.

One important difference compared to the original approach is that, unless the page has a visible anti-adblocking reaction on the main page, we always test the websites on their subpages. The reason is that some news pages show the anti-adblocking warnings only in the articles.

The authors of the original work composed the negative examples set from 100 websites that do not display ads. The reasoning behind this is that they are guaranteed not to contain any anti-adblocking scripts. However, such a choice of examples as a benchmark seems flawed. One of the key challenges in this method is the noise filtering. If there are no ads, then there are practically no sources of execution differences between runs with and without an adblocker. A better choice of websites to evaluate an anti-adblock detection system on would be a mix of websites without ads and sites that display ads, but are known to contain no anti-adblockers. Unfortunately, finding such websites is not easy as it entails a detailed analysis of the entire code served by each site. Certainly, it is not a task for one person to be done in a reasonable timespan. For this reason, we also only consider pages that do not serve ads, exactly 50 of them, as the negative examples.

Zhu et al. [47] have an extensive list of websites with anti-adblockers collected during their previous studies (almost 700 positions). Unfortunately, they do not share this list. For this reason, a similar list of positive examples had to be created from scratch. It is a mix of websites encountered during daily browsing and sites appearing in the Polish anti-adblock filter list [34]. To avoid a time-consuming code analysis at the stage of composing the list, all selected websites contain visible anti-adblocking warnings. Another important note is that the list contains at most a handful of websites from Alexa 500 list [7] to keep the list usable for the small scale study.

The ad-blocking extensions have different policies when it comes to their official, default filters. AdBlock Plus behaves politely and respects publishers' right to ask users to show some support and disable the extension, whatever form such a request takes [4]. On the other hand, uBlock Origin employs a different rule. If the warning is not dismissable or is annoying, it is blocked [28].

Since usage of AdBlock Plus may expose a wider variety of adblock walls and warnings, it was chosen for the experiments.

5.3. Negative examples

The list of websites without ads, on which the system was tested is presented in Table 5.1. The number of execution differences found is listed in the last column.

Table 5.1: Results of a system evaluation on the negative examples – pages without ads

	URL	Diff. count
1	http://administracja.sgh.waw.pl/en/dsl/Pages/default.aspx	0
2	http://architektura.um.warszawa.pl	0
3	http://es6-features.org/	0
4	http://kulikowisko.pl/kursy-prawa-jazdy/	0
5	http://sejm.gov.pl/Sejm8.nsf/poslowie.xsp	0
6	http://www.klubfocus.pl/o-nas	0
7	http://www.mdkopoczno.pl/index.php?cat=historia-mdk	0
8	http://www.mit.edu/research/	0
9	http://www.niebokopernika.pl/repertuar/	0
10	http://www.nsa.gov.pl/sprawozdania-roczne.php	0
11	https://amu.edu.pl/wiadomosci/aktualnosci/doktoranci/rekrutacja-do-szkoly-doktorskiej-uam2	0
12	https://arxiv.org/archive/astro-ph	0
13	https://awa.zh.ch/internet/volkswirtschaftsdirektion/awa/en/standortfoerderung/lebensraum/steuern.html	0

14	https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop	0
15	https://docs.djangoproject.com/pl/2.2/intro/	0
16	https://docs.python.org/3/whatsnew/3.7.html	0
17	https://mansfeld.pl/webdesign/strony-one-page-zalety-wady-mity/	0
18	https://maps.qed.ai/map/browse?page=1&selected_facets=region%3AAfrica	0
19	https://phillipreeve.net/blog/comparison-sigma-art-35mm-1-2-35mm-1-4-and-40mm-1-4/	0
20	https://pl.wikipedia.org/wiki/Bokeh	0
21	https://theoatmeal.com/comics/how_walk_human	0
22	https://uni.wroc.pl/przewodnik-dla-pierwszorocznych/	0
23	https://v8.dev/docs/webassembly-opcode	0
24	https://wiki.archlinux.org/index.php/Zsh	0
25	https://www.chromium.org/for-testers/bug-reporting-guidelines	0
26	https://www.fbi.gov/news/stories/rcfls-follow-the-modern-evidence-trail-081219	0
27	https://www.gatesfoundation.org/What-We-Do/Global-Health/Malaria	0
28	https://www.haskell.org/community/	0
29	https://www.icrc.org/en/war-and-law/treaties-customary-law/geneva-conventions	0
30	https://www.il.pw.edu.pl/index.php/pl/studia/i-stopnia-inzynierskie/kierunkowe-efekty-ksztalcenia	0
31	https://www.kgof.edu.pl/rejestracja/	0
32	https://www.mimuw.edu.pl/rekrutacja	0
33	https://www.napaluchu.waw.pl/czekam_na_ciebie/zasady_adopcji/	0
34	https://www.nsa.gov/what-we-do/cybersecurity/	0
35	https://www.oi.edu.pl/l/35/	0
36	https://www.oie.int/en/for-the-media/onehealth/	0
37	https://www.planetarium.edu.pl/noc-spadajacych-gwiazd-w-parku-slaskim/	0
38	https://www.pwpw.pl/Strefa_wiedzy/Zabezpieczenia/zabezpieczenia-dokumentow.html	0
39	https://www.stanford.edu/research/	0
40	https://www.w3.org/TR/css-grid-1/	0
41	https://www.who.int/immunization/newsroom/new-measles-data-august-2019/en/	0
42	https://www.word.waw.pl/egzaminy/wolne-terminy	0
43	https://www.zoo.lodz.pl/zwiedzanie-zoo/godziny-otwarcia/	0
44	http://kinoelektronik.pl/mazowieckie-centrum-kultury-medialnej-kultura-edukacja-sztuka-filmowa/	1
45	https://ethz.ch/en/research.html	1
46	https://szkolaimpro.pl	1
47	https://www.paperswithcode.com/about	1
48	https://www.whitehouse.gov/briefings-statements/remarks-president-trump-mass-shootings-texas-ohio/	1
49	https://www.worldanimalprotection.org/our-work/protecting-animals-communities	1
50	https://zoo.waw.pl/edukacja/oferta-edukacyjna	1

Ideally, there should be no execution differences on any website listed here. Nonetheless, a few differences were identified (positions 44-50). We comment on them below.

kinoelektronik.pl

The origin of the difference is unclear. There does not seem to be any element blocked by AdBlock. Also, there are no blocked network requests. But, for some reason, there are differences in jQuery traversals.

ethz.ch

The cause of the difference is the tracking code.

szkolaimpro.pl

The difference is due to the progress bar.

www.paperswithcode.com

The execution divergence stems from Sentry – an error reporting solution.

www.whitehouse.gov

The origin of the difference is unclear.

www.worldanimalprotection.org

A similar case to kinoelektronik.pl. The difference is reflected in jQuery traversals but it is unknown why the DOM trees are different.

zoo.waw.pl

The same case as in kinoelektronik.pl and www.worldanimalprotection.org.

5.4. Positive examples

For the positive case evaluation, we run the system on 50 websites that contain visible anti-adblocking warnings. Afterwards, we manually inspected each page and verified if the differences found by the system were relevant.

The results are presented in Table 5.2. Each row consists of a URL, a number of traces with execution divergences, a relevance of the differences, and an anti-adblocker type.

In most cases the system is able to find execution differences, which is a very good sign. In the following sections we analyze when the system fails and discuss how the detected solutions work.

The analysis of 50 websites incorporating adblock walls allows us to create a simple classification of anti-adblocking mechanisms:

- Bait-based
  - A file as a bait
    - A variable value is changed
    - An element injected into the DOM
    - An error handler on a network request
  - A DOM element as a bait
- Real ads checkup
  - A visibility inspection
  - A load verification

We can also enumerate all encountered anti-adblocking solutions:

- Custom scripts
- Off-the-shelf modules:

- BlockAdBlock [37]
- Adblock Detect [18]
- Kill AdBlock [8]
- Admiral's Recover [6]
- *an_message_display*
- *Adb_Detector*

The most popular solutions are custom scripts and open-source BlockAdBlock (already mentioned in Section 1.3). Other open-source scripts include Adblock Detect and KillAdBlock. One website also used a paid solution – the Admiral platform. The origin of the last two solutions is unclear, but the code was clearly the same (the names listed are distinctive functions' names, not the real names of these modules).

Some custom scripts can be really widespread – some publishers own a number of domains and it makes sense to create one robust solution and use it in every website. One example is the Wirtualna Polska group (positions 25 and 27), which uses their own custom anti-adblocking module.

In most cases a significant part of the differences found originated from third-party ads or tracking code (Google Analytics, AdWords, Gemius trackers, Twitter buttons, Facebook buttons, etc.), but it was not inspected further.

Table 5.2: Results of a system evaluation on the positive examples – pages with adblock walls

	URL	Diff. count	Diff rel.?	Type
1	http://jakimasklad.pl/	0	N	BlockAdBlock + eval
2	http://www.srebrnestawy.pl/	0	Y	Custom, var set in a file
3	https://www.cyberciti.biz/faq/ram-size-linux/	1	N	Custom, var set in a file
4	http://zmiksowani.pl/	1	N	Custom, element-based bait
5	https://torrentcity.pl	1	N	Custom, element-based bait + error handler
6	http://www.nadajemy.pl/	1	Y	Custom, var set in a file
7	https://darkw.pl/	1	Y	Custom, ads visibility check
8	https://gragieldowa.pl/notowania/akcje/GR_WIG20	1	Y	Custom, ads visibility check
9	https://waw4free.pl/	1	Y	Adblock Detect
10	https://www.gry-planszowe.pl/forum/	1	Y	Custom, element-based bait
11	https://zw.pl	1	Y	Custom, element-based bait
12	http://www.audiklub.org/dane/a6/c6/color/100.html	2	Y	Custom, var set in a file
13	https://e-dokument.org/sadowe-zazalenia/zazalenie-powoda-na-postanowienie-sadu-odmawiajace-podjecia-zawieszonego-postepowania/	2	Y	BlockAdBlock + eval
14	https://kursbootstrap.pl/tutorial/	2	Y	Custom, element added in a file
15	https://napisy24.pl/	2	Y	BlockAdBlock
16	https://who-called.eu/pl/178533310	3	Y	Custom, ads visibility check
17	https://www.wgrane.pl/	3	Y	Custom, element added in a file
18	https://wzory-cv.com/przyklady.html	3	Y	BlockAdBlock + eval
19	https://megane.com.pl/	4	Y	Custom, ads visibility check
20	https://polscygracze.pl/	4	Y	BlockAdBlock + fallback
21	https://ogloszenia.plock.pl/	5	Y	Custom, var set in a file
22	https://take-kino.pl/	5	Y	Custom, ads visibility check
23	https://www.pajacyk.pl	7	Y	Custom, element added in a file

24	https://www.comperia.pl/	8	N	Custom, ads visibility check
25	https://komorkomania.pl/37730,dji-osmo-mobile-3-oficjalnie-jest-skladany-i-tanszy-od-poprzednika?src01=c801b	9	Y	Custom, ads visibility check
26	https://manjaro.pl/kilka-slow-o-nieoficjalnych-repozytoriach-arch-linux/	9	Y	Abd_Detector, element-based bait
27	https://maciejgalas.fotoblogia.pl/11821,leica-summicron-m-35-mm-f-2-asph	10	Y	Custom, ads visibility check
28	https://www.magazynbieganie.pl/fotostory/bieganie-dla-poczatkujacych-3-proste-sposoby-na-poprawe-formy/	10	Y	Abd_Detector, element-based bait
29	https://kulturalnemedia.pl/muzyka/lindsey-stirling-na-dwoch-koncertach-w-polsce/	11	Y	an_message_display, ads visibility check
30	https://www.filmweb.pl/film/John+Wick-2014-698144	11	Y	Custom, var set in a file
31	https://www.telepolis.pl/artykuly/testy-sprzetu/honor-20-pro-smartfon-telefon-test-recenzja	12	Y	Custom, var set in a file
32	http://lubliniec.info/artykul/12377/ostrzezenie-meteorologiczne-drugiego-stopnia-dla-powiatu-lublinieckiego	13	Y	Custom, var set in a file
33	http://www.krs-online.com.pl/	13	Y	Custom, var set in a file
34	http://www.serialosy.pl/zagraniczne-seriale9/agenci-tarczy/24685-agenci-tarczy-s06e11-napisy-pl.html	13	Y	BlockAdBlock + eval
35	https://kimcartoon.to/Cartoon/Squidbillies-Season-12	15	Y	Custom, scripts loading check
36	https://pomorska.pl/100-najpopularniejszych-stron-internetowych-w-polsce-lista/ar/7198080	16	Y	Custom, element added in a file
37	https://www.businessinsider.com/cheapest-michelin-star-lunch-new-york-city-2018-10	19	Y	BlockAdBlock
38	https://polskiedzieje.pl/polecane-ksiazki/po-dolary-i-wolnosc-dariusz-terlecki.html	23	Y	an_message_display, ads visibility check
39	https://jbzdy.net/	24	Y	Custom, var set in a file
40	http://psychologia-ogolna.wyklady.org/	25	Y	BlockAdBlock + eval
41	https://mmorpg.org.pl/	35	N	BlockAdBlock + fallback
42	https://applemobile.pl/recenzja-bose-sleepbuds-1200-zlotych-za-sluchawki-przez-ktore-nie-posluchasz-wlasnej-muzyki/	36	Y	an_message_display, ads visibility check
43	https://burzowo.info/	41	Y	Custom, ads visibility check
44	https://www.cinemablend.com/	46	Y	Admiral
45	http://chelmski.eu/bedzie-wiecej-polaczen-kolejowych/	49	Y	KillAdBlock
46	https://player.pl/seriale-online/na-wspolnej-1001-1500-odcinki,5134/odcinek-1163,S03E1163,149527	60	N	Custom, scripts loading check
47	https://www.polska-ie.com/	94	N	Custom, element-based bait
48	https://www.motocykl-online.pl/artykuly/Prawo-jazdy-na-motocykl-2019-kat-A-A1-A2-Jak-je-zrobic,12709,1	170	Y	BlockAdBlock
49	https://www.sonyalpharumors.com/the-first-sony-a7riv-versus-a7riii-size-comparison/	181	Y	an_message_display, ads visibility check
50	https://www.pasazer.com/	215	Y	Custom, var set in a file

5.4.1. Websites with unsatisfactory results

jakimasklad.pl

A brief analysis of the code shows that the site incorporates BlockAdBlock [37]. It turns out that, in this case the module's obfuscation is effective enough to circumvent our system. It is

interesting that a number of different websites use the same solution and the system is able to pinpoint the differences in these cases.

www.srebrnestawy.pl

In this case the code that activates the warning seems very simple and there is no reason why it should not be detected by the system. Listing 5.1 shows the code.

```
$(function(){
    ...
    if (advertPolicy === false && typeof window.advertChecker === "undefined") {
        browserAlert.append(
            '<div class="alert alert-warning alert-dismissable">' +
                '<div class="container">' +
                    '<a href="#" class="close" data-dismiss="alert" aria-label="close" title="close">x</a>' +
                    'Do prawidlowego dzialania i wyswietlania sie strony nalezy wylaczyc wtyczke adblocka z Twojej przegladarki!' +
                '</div>' +
            '</div>');
        ...
    }
    ...
}
```

Listing 5.1: An anti-adblocking script of www.srebrnestawy.pl

The reason why it is not detected is that the bait is based on the file *advert.js*, which is not loaded even when AdBlock is disabled, due to an HTTP 404 error. Consequently, the warning is displayed even when AdBlock is disabled and the system is right – there are no execution differences.

www.cyberciti.biz

In this case the difference found corresponds to a PageAd reaction, but the check that activates the adblock warnings is not found.

An anti-adblocker used on this page is simple, works in the same way as the one found on site www.audiklub.pl. This site, however, uses Cloudflare's Rocket Loader [17].

Rocket Loader is a powerful mechanism, developed by Cloudflare, that is able to cache JavaScript code and defer its execution until the entire web page is rendered. The way it works is the following:

- Before serving the page, the server inspects the website and changes all JS scripts to some type not identified by the browser as JavaScript code. The type also has to be unique so that these scripts are easy to find later.
- The server also encloses a special JavaScript module that activates when the entire web page is loaded.
- JavaScript module finds all scripts with changed type and executes them.

Since the module that executes the scripts is JavaScript code, the system should still be able to detect the anti-adblocking code. Unfortunately, it did not manage to do so and the most probable reason seems to be a bit aggressive noise filtering. If some function performs a DOM traversal and the content changes a little each time the page is loaded, it is probable

that it is filtered out. We suspect that it is the case here. That is, Rocker Loader has to find all scripts (a DOM traversal) and execute them as a part of one long function.

torrentcity.pl

This is an example of a rather sophisticated custom anti-adblocking script. Listing 5.2 shows the code that performs the check.

```
function e() {
    var t = "adsSupported";
    var r = window.document.createElement("script");
    r.setAttribute("async", "async");
    r.setAttribute("src", c + "ads.js");
    r.addEventListener("error", function() {
        f(false);
    });
    r.addEventListener("load", function() {
        var e = 0;
        function r() {
            if (!window[t]) {
                if (e++ < 10)
                    window.setTimeout(r, 100);
                else
                    f(false);
            } else {
                setTimeout(function() {
                    if (!window[t]) {
                        f(false);
                    } else {
                        f(window[t].offsetHeight !== 0);
                        if (window[t].parentElement)
                            window[t].parentElement.removeChild(window[t]);
                        return;
                    }
                }, 50)
            }
        }
        r();
    });
    window.document.body.appendChild(r);
}
```

Listing 5.2: An anti-adblocking script of torrentcity.pl

The script first sets up a bait by injecting a new script element into the DOM. It is set up to download a file named *ads.js*, which should be blocked by any adblocker.

Next, the error handler is set up to activate the reaction. If the load succeeds, it still makes sure that the ads are actually displayed.

Functions called as event handlers or via *setTimeout* are a weak spot of the Differential Execution Analysis method, not just this exact implementation and there is no known way to automatically utilize such unpaired events.

player.pl

An example of a really robust solution, probably used on every site belonging to the TVN media company. Listing 5.3 shows the checks. The website utilizes more than one check to increase the detection effectiveness. The checks itself are similar to what we have already

seen. The first one creates an element as a bait and checks if it is visible. The second one uses a different bait, a file, and reacts within callbacks. The system fails here because the script uses the techniques that are the weak spot of the presented method.

```
var isAdBlockEnabled = function() {
    if (void 0 === window.isAdBlockEnabledCheckCounter && (window.
    isAdBlockEnabledCheckCounter = 0),
    !document.body)
        return window.isAdBlockEnabledCheckCounter < 15 && setTimeout(
    function() {
            window.isAdBlockEnabledCheckCounter++,
            isAdBlockEnabled()
        }, 10),
        window.adBlockEnabled = !1;
    var e = document.createElement("div");
    e.setAttribute("class", "sponsored-ad adoSlave advertisement pub_300x250
    pub_300x250m pub_728x90 text-ad textAd text_ad text_ads text-ads text-ad-
    links"),
    e.setAttribute("style", "width: 1px !important; height: 1px !important;
    position: absolute !important; left: -1000px !important; top: -1000px !
    important;"),
    document.body.appendChild(e);
    var t = $(e)
        , i = $(e).is(":visible");
    return t.remove(),
    window.adBlockEnabled = !i,
    !i
};
// ...
v.isAdBlockEnabledAgain = function(e, t) {
    var i = nuvi.settings.accessProtocol + "//s1-player5.cdntvn.pl/advert.js"
    ;
    -1 !== i.indexOf("tvwisla.com.pl") && -1 === i.indexOf("cdntvn.stage.
    online") && (i = (i = i.replace("player5", "s1-player5")).replace(".stage.
    online", "-cdntvn.stage.online")),
    $.ajax({
        url: i,
        type: "GET",
        cache: !1,
        async: !1,
        success: function() {
            return v.adblockEnabledAgain = !1,
            "function" == typeof e && e(),
            !1
        },
        error: function() {
            return v.adblockEnabledAgain = !0,
            v.options().adBlockCheck && !tvn24ScrollTvnCheck() ? "function"
    == typeof t && t(v) : "function" == typeof e && e(),
            !0
        }
    })
}
```

Listing 5.3: An anti-adblocking script of `player.pl`

Other websites

In cases of other websites where the relevant execution divergences were not found the reasons were the same – either the check was intertwined between other activities in a way that filtering would discard it or the check was fired via callbacks.

5.4.2. Detected anti-adblockers

This section presents some interesting mechanisms that have not been covered yet but have been detected. Simple examples were preferred to avoid showing cluttered code.

www.audiklub.pl

This is the simplest anti-adblocking script one can think of. The website includes just one advertisement, but when an adblocker is detected, the entire content is blocked (Listing 5.4).

```
window.onload = function() {
    if( window.canRunAds === undefined ){
        // adblocker detected, show fallback
        $('div#contentDiv').html("<b>Wylacz adblock na tej stronie. Tu jest tylko jedna mala niewinwazyjna reklama.</b>");
        //showFallbackImage();
    }
}
```

Listing 5.4: An anti-adblocking script of www.audiklub.pl

First, the trap is set up. It is a file named *ads.js*. Its content is just one line which sets the *canRunAds* variable to true.

The main check is performed in the *window.onload* callback. The script simply checks if the *canRunAds* variable has been set. If not, it replaces the content div with an anti-adblocking warning.

kursbootstrap.pl

The script on this website serves as an example of an anti-adblocker that uses a file as a bait but the check is performed by injecting an element into the DOM instead of setting a global variable (Listing 5.5).

```
// In "advertisement.js"
document.write('<div id="tester">an advertisement</div>');

// In the main page, inline
if (document.getElementById("tester") != undefined) {
    document.write('<div class="alert alert-success alert-dismissible" role="alert"><button type="button" class="close" data-dismiss="alert"><span aria-hidden="true">&times;</span><span class="sr-only">Zamknij</span></button><strong>Dziekuje!</strong> Widze, ze nie uzywasz AdBlocka. To dobrze:) Reklamy sa nieodlaczna czescia darmowych stron i pozwalaja autorom na ich rozwoj. Jesli doceniasz prace innych, nie uzywaj AdBlocka na tych witrynach.</div>');
} else {
    document.write('<div class="alert alert-danger alert-dismissible" role="alert"><button type="button" class="close" data-dismiss="alert"><span aria-hidden="true">&times;</span><span class="sr-only">Zamknij</span></button><strong>Oj! niedobrze:(</strong> Wyglada na to, ze uzywasz AdBlocka. Reklamy na moim blogu nie sa inwazyjne i w zaden sposob nie przeszkadzaja w odbiorze tresci. Reklamy sa nieodlaczna czescia darmowych stron i
```

```
    pozwalaja autorom na ich rozwoj. Jesli to mozliwe wylacz AdBlocka dla tej
    domeny. Dziekuje!</div>');
}
```

Listing 5.5: An anti-adblocking script of kursbootstrap.pl

who-called.eu

The code from this website demonstrates how an anti-adblocking script can detect if real ads are displayed, without using a bait (Listing 5.6). The detecting function is run 5 seconds after the page finishes loading. It then checks if the content of the element displaying ads is empty (contains 0 non-whitespace characters).

```
window.onload = function () {
    setTimeout(function () {
        var ad = document.querySelector("ins.adsbygoogle");
        if (ad && ad.innerHTML.replace(/\s/g, "").length == 0) {
            $('.no-ads-info').show();
        }
        console.log('jest ok');
    }, 5000);
};
```

Listing 5.6: An anti-adblocking script of who-called.eu

kimcartoon.to

This website employs a check based on making sure that advertising scripts are loaded (Listing 5.7). It seems, though, that the code contains a bug. Perhaps it would be desirable to invert the condition and change the variable only when one of the scripts fails to load (currently it is enough that one of them is loaded correctly)

```
if (adbWarn == null) {
    $.ajaxSetup({
        cache: false
    });
    var alb = true;
    $.getScript('https://pubmatic.com/wp-content/themes/pubmatic/js/jquery.
    alignHeight.js?ver=1.0').done(function(script, textStatus) {
        alb = false;
    });
    $.getScript('https://propellerads.com/wp-content/plugins/radiantthemes-
    addons/tabs/js/radiantthemes-tab-element-four.js').done(function(script,
    textStatus) {
        alb = false;
    });
    $.getScript('https://www.bebi.com/js/plugins.js').done(function(script,
    textStatus) {
        alb = false;
    });
    setTimeout(function() {
        if (alb) {
            $('#adbWarnContainer').css({
                display: "block"
            });
        }
    }, 8000);
```

```
}
```

Listing 5.7: An anti-adblocking script of kimcartoon.to

e-dokument.org and others using BlockAdBlock with *eval* obfuscation

BlockAdBlock is one of the most popular off-the-shelf anti-adblocking solutions. The obfuscated code is quite unreadable to a human (Listing 5.8). Fortunately, the string to be evaluated contains "BlockAdBlock" as a substring and we can be sure that it was the solution used. Since BlockAdBlock is an open-source module, an interested reader can check out its code in the plain, unobfuscated form on github [37].

```
// Place this code snippet near the footer of your page before the close of the /body tag
// LEGAL NOTICE: The content of this website and all associated program code are protected under the Digital Millennium Copyright Act. Intentionally circumventing this code may constitute a violation of the DMCA.
eval(function(p, a, c, k, e, d) {
    e = function(c) {
        return (c < a ? '' : e(parseInt(c / a))) + ((c = c % a) > 35 ? String.fromCharCode(c + 29) : c.toString(36))
    };
    if (!''.replace(/^/, String)) {
        while (c--) {
            d[e(c)] = k[c] || e(c);
        }
        k = [ function(e) { return d[e] } ];
        e = function() { return '\\w+' };
        c = 1;
    }
    while (c--) {
        if (k[c]) {
            p = p.replace(new RegExp('\\b' + e(c) + '\\b','g'), k[c]);
        }
    }
    return p;
}(';q Q=\'\',27=\'21\';1N(q i=0;i<12...') // the string is truncated
```

Listing 5.8: An anti-adblocking script of e-dokument.org

polscygracze.pl

Another website that uses BlockAdBlock, but adds a fallback in case the module is blocked by an adblocker (Listing 5.9). It is worth mentioning that the idea of making the id of the adblock warning element random is good but the identifier should be dynamic, i.e., generated randomly each time the page is loaded. If it is static, it can still be added to adblockers' filter lists.

```
function adBlockDetected() {
    $('#07th202231').show();
}
function adBlockNotDetected() {
    $('#07th202231').hide();
}
if (typeof blockAdBlock === 'undefined') {
    adBlockDetected();
} else {
```

```
10      blockAdBlock.onDetected(adBlockDetected).onNotDetected(adBlockNotDetected
        );
11  }
```

Listing 5.9: An anti-adblocking script of polscygracze.pl

Other off-the-shelf solutions

Other solutions utilize mechanisms similar to the already described ones. They can be more polished and use more than one check but they do not introduce any new ideas. For this reason they are not described further here. An interested reader can check their documentation.

5.5. Small scale study

The experiment has been conducted on almost 300 top websites from Alexa 500 list (the version for Poland) [7]. The list of the visited websites and the results is attached in Appendix A. Some websites were skipped because they either contain adult content or their main domain does not contain any web page – this is the case, e.g., for content delivery networks. These websites, instead of results, contain dashes (--) in the last two columns.

Some results are marked with "TO" (timeout) instead of a number of differences found. This is because there is a limited time for each trace collection (30 seconds). After that the browser is closed. In some cases the browser times out instead of closing. The system then waits another 90 seconds and closes the browser forcefully. Usually this is a sign of a website producing an enormously large execution trace (some can reach almost 50GB in 2 minutes). Such pages are also skipped.

Although the number of ads, social media links and similar elements was not counted, it seems that the more elements of this kind a website incorporates, the more likely it is that there will be an anti-adblocking reaction of some kind. It is especially true in case of third-party ads which more often than not report an adblocker presence.

Out of 281 visited websites, 33 contained visible anti-adblocking warnings. The system detected anti-adblocking reactions in 163 web pages, 103 were found to be free of anti-adblockers. The rest (15 pages) timed out. Looking at the results, it seems that the anti-adblocking scripts are widespread. Silent anti-adblocking reactions are over 5 times more frequent than adblock walls (163 reactions detected vs 33 adblock walls detected). Over 50% of the visited websites exhibit some sort of anti-adblocking reactions. It is not that surprising if we consider that most websites contain advertisements and the biggest ad providers report adblockers usage.

5.6. Conclusion

The performed tests show that the system is capable of detecting the anti-adblocking scripts in many scenarios. The exact numbers are not presented purposefully – the tests should be conducted on larger set of websites and the methodology should be different when it comes to the negative examples.

The presented solution certainly has some weak spots. The first one is the event coverage. It is easy to hide anti-adblockers by utilizing callbacks. The second one is the data filtering mechanism which in some cases may filter out too much data, and in others it lets through too many traces, which limits its usefulness (the analysis becomes much harder, the optimum would be to always have less that 10 trace diffs for manual verification).

Here are some ideas that could alleviate both problems:

- Make an extension that would put a thin wrapper around some critical built-in functions. Such wrapper would only call the original built-in function but the entry and the exit would be logged by the system. It might be useful for functions such as *getElementById* or direct DOM manipulations.
- Manually analyze and tag the results of more traces (at least hundreds of them) and try some machine learning method to filter the resulting diffs. A first try could be to use the random decision forests.

The system is also a bit too slow. The trace collection should have less overhead on the browser. For a large scale study spending 10 minutes on each website to collect traces is too long. One idea is to have Chromium assemble a strings map instead of constructing it the during analysis. This, in conjunction with a use of some binary format, would result in much more compact files and less time wasted on writing to files.

Last but not least, the conducted experiments give lots of insights on how to build an effective anti-adblocker:

- Use more than one detection method, preferably utilizing callbacks.
- Obfuscate the code by using minification combined with the *eval* function to make manual analysis extremely hard.
- Randomize ids of injected adblock warnings to make it harder for the adblockers to filter them out.

Bibliography

[1] Acceptable Ads initiative. Acceptable Ads. https://acceptableads.com, 2019. [Online; accessed 6-August-2019].

[2] Adblock Analytics. Detect Adblock. https://www.detectadblock.com, 2019. [Online; accessed 29-July-2019].

[3] AdBlock authors. What is Adware? How to Identify and Prevent Adware. https://blog.getadblock.com/what-is-adware-ce135d866898, 2019. [Online; accessed 6-August-2019].

[4] AdBlock authors. Why doesn't AdBlock bypass anti-adblock walls? https://help.getadblock.com/support/solutions/articles/6000199143-why-doesn-t-adblock-bypass-anti-adblock-walls-, 2019. [Online; accessed 15-August-2019].

[5] Adelina Tuca. 5 best anti adblock wordpress plugins. https://themeisle.com/blog/best-anti-adblock-wordpress-plugins/, 2018. [Online; accessed 6-August-2019].

[6] Admiral. Admiral: A Proven Platform Helping Over 12,000 Publishers Worldwide. https://getadmiral.com/products, 2019. [Online; accessed 19-August-2019].

[7] Alexa. Top Sites in Poland. https://www.alexa.com/topsites/countries/PL, 2019. [Online; accessed 4-August-2019].

[8] badmetevils. Kill Adblock. https://www.drupal.org/project/killadblock, 2018. [Online; accessed 19-August-2019].

[9] Baiju Muthukadan. Selenium with Python. https://selenium-python.readthedocs.io, 2019. [Online; accessed 6-August-2019].

[10] Ben L. Titzer, Jaroslav Sevcik. A year with Spectre: a V8 perspective. https://v8.dev/blog/spectre, 2019. [Online; accessed 2-August-2019].

[11] Ben Titzer et al. TurboFan JIT Design. https://docs.google.com/presentation/d/1sOEF4MlF7LeO7uq-uThJSulJlTh--wgLeaVibsbb3tc/edit#slide=id.p, 2019. [Online; accessed 29-July-2019].

[12] Bryan O'Sullivan. attoparsec: Fast combinator parsing for bytestrings and text. http://hackage.haskell.org/package/attoparsec, 2019. [Online; accessed 5-August-2019].

[13] Christopher Elliott. Yes, There Are Too Many Ads Online. Yes, You Can Stop Them. Here's How. https://www.huffpost.com/entry/yes-there-are-too-many-ads-online-yes-you-can-stop_b_589b888de4b02bbb1816c297, 2017. [Online; accessed 6-August-2019].

[14] Chrome team. Content Scripts. https://developer.chrome.com/extensions/content_scripts, 2019. [Online; accessed 2-August-2019].

[15] Chrome team. Extensions Overview. https://developer.chrome.com/extensions/overview, 2019. [Online; accessed 2-August-2019].

[16] Chrome team. Site Isolation. https://www.chromium.org/Home/chromium-security/site-isolation, 2019. [Online; accessed 2-August-2019].

[17] Cloudflare Support. What does Rocket Loader do? https://support.cloudflare.com/hc/en-us/articles/200168056-What-does-Rocket-Loader-do-, 2019. [Online; accessed 19-August-2019].

[18] Damir Brekalo. Adblock detect. https://dbrekalo.github.io/adblockdetect/, 2019. [Online; accessed 19-August-2019].

[19] Don Stewart, Duncan Coutts. bytestring: Fast, compact, strict and lazy byte strings with a list interface. http://hackage.haskell.org/package/bytestring, 2019. [Online; accessed 5-August-2019].

[20] Felix Meier. Iroh - Dynamic code analysis for JavaScript. https://maierfelix.github.io/Iroh/, 2018. [Online; accessed 29-July-2019].

[21] Franziska Hinkelmann. Understanding V8's Bytecode. https://medium.com/dailyjs/understanding-v8s-bytecode-317d46c94775, 2017. [Online; accessed 1-August-2019].

[22] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[23] Google LLC. Web Tracing Framework. https://google.github.io/tracing-framework/index.html, 2019. [Online; accessed 29-July-2019].

[24] Haskell contributors. Haskell. https://www.haskell.org, 2019. [Online; accessed 5-August-2019].

[25] IAB. IAB Internet Advertising Revenue Report 2018 Full Year Results. https://www.iab.com/insights/iab-internet-advertising-revenue-report-2018-full-year-results/, 2019. [Online; accessed 6-August-2019].

[26] Jason Mander and Chris Beer. Ad-Blocking. A deep-dive into ad-blocking trends. https://www.globalwebindex.com/hubfs/Downloads/Ad-Blocking-trends-report.pdf, 2018. [Online; accessed 27-August-2019].

[27] Noah Johnson, Juan Caballero, Kevin Zhijie Chen, Stephen McCamant, Pongsin Poosankam, Daniel Reynaud, and Dawn Song. Differential slicing: Identifying causal execution differences for security applications. In *Proceedings of the 32nd IEEE Symposium on Security and Privacy*, pages pp. 347–362. IEEE, May 2011.

[28] Kaleigh Rogers. Why Your Ad Blocker Doesn't Block Those 'Please Turn Off Your Ad Blocker' Popups. https://www.vice.com/en_us/article/j5zk8y/why-your-ad-blocker-doesnt-block-those-please-turn-off-your-ad-blocker-popups, 2018. [Online; accessed 15-August-2019].

[29] Paul Kocher, Jann Horn, Anders Fogh, , Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In *40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019.

[30] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown: Reading kernel memory from user space. In *27th USENIX Security Symposium (USENIX Security 18)*, 2018.

[31] Mark Karpov. Short ByteString and Text. https://markkarpov.com/post/short-bs-and-text.html, 2017. [Online; accessed 5-August-2019].

[32] Martin Brinkmann. Integrate Nano Defender with uBlock Origin to block Anti-Adblocker. https://www.ghacks.net/2019/02/15/integrate-nano-defender-with-ublock-origin-to-block-anti-adblocker/, 2019. [Online; accessed 6-August-2019].

[33] MDN. Concurrency model and event loop. https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop, 2019. [Online; accessed 26-July-2019].

[34] olegwukr. Polish Anti Adblock Filters. https://github.com/olegwukr/polish-privacy-filters/blob/master/anti-adblock.txt, 2019. [Online; accessed 15-August-2019].

[35] PageFair. 2017 Adblock Report. https://pagefair.com/blog/2017/adblockreport/, 2017. [Online; accessed 6-August-2019].

[36] Reek. Anti-Adblock Killer. https://reek.github.io/anti-adblock-killer/, 2019. [Online; accessed 6-August-2019].

[37] sitexw. BlockAdBlock. https://github.com/sitexw/BlockAdBlock, 2018. [Online; accessed 6-August-2019].

[38] StackOverflow users. Adding console.log to every function automatically. https://stackoverflow.com/questions/5033836/adding-console-log-to-every-function-automatically, 2017. [Online; accessed 25-July-2019].

[39] uBlock Origin Contributors. This is uBlock's manifesto. https://github.com/gorhill/uBlock/blob/master/MANIFESTO.md, 2015. [Online; accessed 27-July-2019].

[40] V8 Team. Built-in functions. https://v8.dev/docs/builtin-functions, 2019. [Online; accessed 2-August-2019].

[41] V8 team. Design of V8 bindings. https://chromium.googlesource.com/chromium/src/+/master/third_party/blink/renderer/bindings/core/v8/V8BindingDesign.md, 2019. [Online; accessed 6-August-2019].

[42] V8 Team. V8 Torque user manual. https://v8.dev/docs/torque, 2019. [Online; accessed 2-August-2019].

[43] V8 Team. What is V8? https://v8.dev, 2019. [Online; accessed 29-July-2019].

[44] W3Schools. Browser Statistics. https://www.w3schools.com/browsers/, 2019. [Online; accessed 29-July-2019].

[45] Wikipedia contributors. Wordpress — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=WordPress&oldid=909584536, 2019. [Online; accessed 6-August-2019].

[46] Bin Xin, William N. Sumner, and Xiangyu Zhang. Efficient program execution indexing. In *Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '08, pages 238–248, New York, NY, USA, 2008. ACM.

[47] Shitong Zhu, Xunchao Hu, Zhiyun Qian, Zubair Shafiq, and Heng Yin. Measuring and disrupting anti-adblockers using differential execution analysis. In *NDSS*. The Internet Society, 2018.

Appendix A

	URL	Vis. warn.	Diff. count
1	https://www.google.com/search?q=tom+cruise	N	35
2	https://www.youtube.com/watch?v=tp-wQYyrxd0	N	34
3	https://www.google.pl/search?q=tom+cruise	N	35
4	https://allegro.pl/oferta/telewizor-led-55-4k-mpeg4-usb-3-0-smart-wi-fi-a-7568489180	N	13
5	https://www.facebook.com/BradPittPlanB/	N	5
6	https://www.olx.pl/oferta/hyundai-i30-1-6-crdi-90-km-CID5-IDATnhX.html	N	9
7	https://wiadomosci.wp.pl/mini-czarnobyle-seria-niefortunnych-wpadek-rosyjskiego-wojska-6412764769474689a	Y	TO
8	https://wiadomosci.onet.pl/nauka/zmarl-wybitny-rosyjski-astrofizyk-nikolaj-kardaszow-tworca-skali-rozwoju-cywilizacji/f96d48q	Y	8
9	https://pl.wikipedia.org/wiki/Bokeh	N	0
10	https://fakty.interia.pl/swiat/news-iranska-telewizja-iran-przejal-zagraniczny-tankowiec,nId,3130879	N	4
11	https://bongacams.com	–	–
12	https://vk.com/club75379670	N	0
13	https://www.wiocha.pl/1589883,Zona-skarb	N	10
14	https://www.otomoto.pl/oferta/bmw-seria-8-840d-xdrive-coupe-upust-126000-pln-ID6CdinC.html	N	16
15	https://www.cda.pl/video/265923314/vfilm	N	2
16	https://www.aliexpress.com/item/32989785993.html	N	43
17	https://www.o2.pl/artykul/mlodzi-mniej-przejmuja-sie-alertami-rzadowego-centrum-bezpieczenstwa-6409157712651905a	Y	23
18	https://kobieta.gazeta.pl/kobieta/1,107881,25045075,pani-zacisnie-piesc-nie-dalam-rady-dlugie-paznokcie-mi-przeszkodzily.html	N	14
19	https://www.ceneo.pl/Torby_i_walizki	N	7
20	https://www.ipko.pl	N	0
21	https://www.reddit.com/r/europe/comments/cltm32/black_forest_germany/	N	TO
22	https://www.mbank.pl/indywidualny/	N	50
23	https://www.netflix.com/pl-en/	N	1
24	https://pornhub.com	–	–
25	https://www.filmweb.pl/film/Top+Gun%3A+Maverick-2020-602101	Y	7
26	https://www.gumtree.pl/a-mieszkania-i-domy-do-wynajecia/krakow/kawalerka-do-wynajacia/1005563446690911255418809	N	2
27	https://www.centrum24.pl/	N	0
28	https://www.booking.com/hotel/pl/willa-nad-potokiem-ladek-zdroj.pl.html	N	TO
29	https://uwaga.tvn.pl/reportaze,2671,n/jak-polacy-pija-malpki,296314.html	N	6
30	https://www.twitch.tv/summoningsalt/videos	N	TO
31	https://livejasmin.com	–	–
32	https://www.instagram.com/paulnicklen/	N	6
33	https://www.wykop.pl/link/5074087/genialny-pomysl-pkp-dworzec-centralny-ma-wygladac-jak-w-latach-70/	N	3
34	https://www.yahoo.com/news/texas-woman-killed-by-police-officer-attempting-to-shoot-her-dog-083943364.html	N	TO

35	https://fili.cc/serial/czarnobyl/s01e01/12345/63706	N	0
36	https://zalukaj.com/zalukaj-film/48411/krol-lew-the-lion-king-2019.html	N	0
37	https://www.bankmillennium.pl	N	0
38	http://wyborcza.pl/7,75399,25056799,latajacy-czlowiek-tym-razem-zwyciezyl-pokonal-kanal-la-manche.html	Y	6
39	https://zen.yandex.ru/media/clickoncar/skolkovo-tak-ne-sdelaet-nikogda-smotrim-na-hiphi-1-5d458a1c7cccba00ad56ec0a	N	0
40	https://chomikuj.pl/faworit2/Systemy+LINUX/Linux+Ubuntu-PL-ISO	N	1
41	http://paulinaphotographer.blogspot.com	N	1
42	https://www.pracuj.pl/praca/front-end-software-developer-wroclaw,oferta,6999667	N	2
43	https://www.money.pl/gospodarka/faktura-za-wizyte-prezydenta-mieszkaniec-szczecinka-oddal-pieniadze-ministrowi-6409902580459649a.html	N	20
44	https://xhamster.com	–	–
45	https://kinokrad.co/336686-yesterday.html	N	0
46	https://www.google.ru/search?q=tom+cruise	N	9
47	https://xvideos.com	–	–
48	https://www.fandom.com/articles/the-mandarin-shang-chi-father-mcu	N	1
49	https://www.roblox.com	N	0
50	https://news.mail.ru/politics/38207623/	N	138
51	https://www.otodom.pl/oferta/lux-2-pokoje-powisle-loggia-metro-park-ID41DlR.html	N	0
52	https://www.pekao24.pl	N	1
53	https://www.microsoft.com/pl-pl/p/surface-laptop-2/8XQJKK3DD91B	N	37
54	https://www.google.com.ua/search?q=tom+cruise	N	1
55	https://tygodnik.tvp.pl/43744932/albo-dzieci-albo-ekologia-jeden-nastolatek-mniej-to-ulga-dla-ziemi	N	5
56	https://www.pudelek.pl/artykul/149734/rihanna_atakuje_donalda_trumpa_po_strzelaninach_w_usa_latwiej_jest_tu_zdobyc_karabin_niz_wize/	N	6
57	https://naszemiasto.pl/nowelizacja-prawa-budowlanego-coraz-blizej-znikna-absurdy-i/ar/c9-7282497	Y	8
58	https://corporate.payu.com	N	1
59	https://www.komputerswiat.pl/aktualnosci/sprzet/nowy-laptop-gamingowy-od-xiaomi-juz-oficjalnie-poznalismy-cene-urzadzenia/yzcmnje	Y	6
60	https://www.gemius.com/e-commerce-news/ukraine-users-attitude-illegal-content.html	N	0
61	https://rezka.ag/films/documentary/31527-vo-slavu-tmy-2019.html	N	2
62	https://sprzedajemy.pl/vw-t1-bulik-ogorek-klasykiem-do-slubu-reklama-eventy-bielsko-biala-2-85b084-nr58175417	N	1
63	https://www.fakt.pl/wydarzenia/polityka/posel-janusz-sanocki-wylamal-bramke-w-opolskim-urzedzie-wojewodzkim-interweniowala/dp5sdxx	Y	7
64	https://demotywatory.pl/4940852/Nie-wiem-czy-kierowca-autobusu-po-tym-klaskal-ale-warto	N	32
65	http://www.meteo.pl	N	0
66	https://joemonster.org/art/47440/Slynni_artysci_kiedys_i_dzis	N	8
67	http://www.gemius.pl/agencje-aktualnosci/wyniki-badania-gemiuspbi-za-kwiecien-2019.html	N	6
68	https://login.live.com	N	0
69	http://www.sport.pl/kolarstwo/7,64993,25061164,koszmarny-wypadek-na-tour-de-pologne-kolarz-jest-reanimowany.html	N	0
70	https://www.ing.pl	N	2
71	https://kwejk.pl/obrazek/3421509/strefa-wolna-od-lpg.html	Y	4
72	https://www.aliorbank.pl	N	0
73	https://redtube.com	–	–
74	https://www.elektroda.pl/rtvforum/topic3602578.html	Y	0
75	https://businessinsider.com.pl	Y	6
76	https://www.euro.com.pl/cms/tg-samsung-galaxy-a50-z-opaska.bhtml	N	9
77	https://www.kinopoisk.ru/media/podcast/3398593/	N	0
78	https://bnpparibas.pl/klienci-indywidualni	N	13

79	https://www.x-kom.pl/p/506331-smartfon-telefon-xiaomi-mi-a3-4-64gb-blue.html	N	4
80	https://www.zalando.pl/weekday-dakota-felted-jacket-kurtka-jeansowa-rocky-blue-web22t021-k11.html	N	51
81	https://www.gry-online.pl/S018.asp?ID=2063	N	2
82	https://www.ing.pl	N	2
83	https://www.mediaexpert.pl/telewizory/telewizor-lg-led-55um7610,id-1367344	N	142
84	https://wpolityce.pl/sport/457898-banka-zapowiedzial-125-mln-zl-na-kilka-obiektow-sportowych	N	8
85	https://eblik.pl	N	0
86	https://xnxx.com	–	–
87	https://jbzdy.net	Y	24
88	https://www.olx.ua/obyavlenie/razborka-shrot-rozborka-skoda-superb-i-azm-IDAPIUA.html	N	7
89	https://www.dobreprogramy.pl/ASUS-ROG-Strix-XG438Q.-Najwiekszy-monitor-4K-UHD-FreeSync-2-HDR,News,103080.html	N	1
90	https://www.amazon.com/dp/B07MMZ2LTB/	N	184
91	https://twitter.com/pittofficial	N	0
92	https://plejada.pl/newsy/piotr-schramm-w-stacji-hejt-ofiary-mowy-nienawisci-targaja-sie-na-zycie-dosc-tego/lrhsqnn	Y	19
93	https://www.pomponik.pl/plotki/news-elzbieta-ii-miala-dosc-gaf-meghan-wyznaczyla-jej-specjalna-m,nId,3132626	N	TO
94	https://www.przelewy24.pl	N	15
95	https://home.pl/office365/	N	0
96	https://www.play.pl/youtube/	N	1
97	https://www.morele.net/karta-graficzna-msi-geforce-rtx-2070-gaming-8g-8gb-gddr6-geforce-rtx-2070-gaming-8g-5411029/	N	2
98	https://imgur.com/t/dogs_are_the_best_people/iCfQe0O	N	13
99	https://www.intercity.pl/pl/	N	1
100	https://www.empik.com/za-duzo-mysle-poradnik-dla-analizujacych-bez-konca-petitcollin-christel,p1222736270,ksiazka-p	N	5
101	https://datezone.com	–	–
102	https://niezalezna.pl/282703-usa-rasizm-nie-byl-motywem-strzelaniny	N	0
103	https://ok.ru/realmadrid.c.f	N	8
104	https://www.google.de/search?q=tom+cruise	N	9
105	https://www.flashscore.pl	N	0
106	https://kinogo.by/16305-avengers-endgame_2019_28-07.html	N	0
107	https://hbogo.pl/seriale/czarnobyl/sezon-1	N	3
108	https://www.teleman.pl/tv/Resident-Evil-Apokalipsa-637242	N	0
109	https://sport.se.pl/kolarstwo/tragedia-na-trasie-tour-de-pologne-bjorg-lambrecht-zmarl-na-stole-operacyjnym-aa-3S6v-cL87-kLXz.html	Y	7
110	https://stackoverflow.com/questions/57365580/is-main-return-a-program	N	0
111	https://pz.gov.pl/pz/index	N	0
112	https://vod.pl/filmy/alita-battle-angel-caly-film-online/35p9esb	N	10
113	https://www.imdb.com/title/tt6095472/	N	56
114	https://prm.ua/estoniya-ne-viznavatime-rosiyski-pasporti-vidani-zhitelyam-donbasu/	N	0
115	https://wordpress.com	N	0
116	https://www.adobe.com/products/photoshop-lightroom.html	N	2
117	http://rozklad-pkp.pl	N	1
118	https://www.pkobp.pl	N	0
119	https://microsoftonline.com	–	–
120	http://www.poczta-polska.pl/bohateron-oddajmy-hold-powstancom/	N	0
121	https://discordapp.com	N	0
122	https://shinden.pl/series/52932-dungeon-ni-deai-wo-motomeru-no-wa-machigatteiru-darou-ka-2nd-season	N	0
123	https://pl.bab.la/zwroty/rekrutacja/list-motywacyjny/polski-angielski	N	6
124	https://exsite24.pl/darmowe_ebook_download/czasopisma_zagraniczne/218673-practical-photography-august-2019.html	N	0

125	http://seasonvar.ru/serial-22766-Doktor__Han.html	N	0
126	https://www.office.com	N	7
127	https://portal.abczdrowie.pl/gotowane-czy-surowe-jak-jesc-warzywa-aby-mialy-najwiecej-wartosci-odzywczych	N	TO
128	https://soundcloud.com/nlechoppa/shotta-flow-3	N	0
129	https://www.spotify.com/pl/	N	0
130	http://krakow.pl/aktualnosci/231889,33,komunikat,za_nami_patriotyczna_lekcja_spiewania.html	N	2
131	https://mediamarkt.pl/sonywf1000xm3	N	2
132	https://www.messenger.com	N	5
133	https://www.orange.pl/love	N	11
134	https://biznes.radiozet.pl/News/Najbogatsze-i-najbiedniejsze-gminy-w-Polsce.-Ranking-wygral-Kleszczow-a-na-ostatnim-miejscu-Zawadzkie	N	0
135	https://www.biedronka.pl/pl/kuchnia-azjatycka-01-08	N	4
136	https://www.castorama.pl/domowa-jednostka-sterujaca-24v-id-1106728.html	N	420
137	https://www.ebay.com/itm/Polarized-Sunglasses-Mens-Womens-Sport-Running-Fishing-Golf-Driving-Glasses/273348514805	N	58
138	https://www2.hm.com/pl_pl/productpage.0488561032.html	N	2
139	https://jakdojade.pl/wroclaw/trasa/	N	TO
140	https://www.trojmiasto.pl/wiadomosci/Pierwszy-odcinek-trasy-tramwajowej-na-Stogi-zostanie-jednak-otwarty-jesienia-n136680.html	Y	7
141	https://steamcommunity.com	N	0
142	https://store.steampowered.com/app/1089350/NBA_2K20/	N	563
143	https://sex.com	–	–
144	http://www.gry.pl/gra/chrome-dino	Y	31
145	https://www.auto-swiat.pl/klasyki/oldtimer/klasyk-z-najwyzszej-polki-citroen-ds-21-presidentielle/bz1xrvg	Y	11
146	https://www.bankier.pl/wiadomosc/Donald-Trump-o-oslabieniu-juana-Chiny-manipuluja-waluta-7717876.html	N	33
147	https://inpost.pl/przesylki-paczkomaty-kurier	N	0
148	https://www.spidersweb.pl/2019/08/donald-trump-gry-wideo.html	N	TO
149	https://pl.jooble.org/jdp/-2553381723608201811/UI-%2F-Web-Developer-Warszawa	N	0
150	https://www.santander.pl	N	1
151	https://www.lotto.pl/eurojackpot/aktualnosci/ponad-1-mln-zlotych-w-eurojackpot-w-bydgoszczy-kumulacja-nadal-rosnie	N	20
152	https://www.infor.pl/prawo/powiat/rejestracja-samochodu/2758936,Podatek-przy-zakupie-auta-2019-r.html	N	8
153	https://youporn.com	–	–
154	https://blog.mozilla.org/blog/2019/06/04/when-it-comes-to-privacy-default-settings-matter/	N	0
155	https://www.pepper.pl/promocje/kupon-na-115-zl-do-hulajnog-citybee-177880	N	0
156	https://www.gamepedia.com/news/1977-defy-the-rules-of-mass-and-physics-with-morphies	N	16
157	https://showup.tv	–	–
158	http://www.reverso.net/text_translation.aspx?lang=EN	N	5
159	https://www.decathlon.pl/rower-mtb-st-100-275-id_8400336.html	N	TO
160	https://ekino-tv.pl/movie/show/ksiezniczka-labedzi-i-krolestwo-muzyki-hd-the-swan-princess-kingdom-of-music-2019-dubbing/26187	N	37
161	https://autokult.pl/34501,kamper-nie-musi-byc-nudny-taki-transit-to-marzenie-kazdego-fana-wyscigow	Y	1
162	https://thestartmagazine.com/article/28389333-0516-450d-8f8c-cd0d8eb77087	N	253
163	https://www.tripadvisor.com/AttractionProductReview-g60763-d11451132-Statue_of_Liberty_and_Ellis_Island_Award_Winning_Small_Group_Tour-New_York_City_New.html	N	42
164	http://www.medonet.pl/zdrowie,odciski--jakie-specjalistyczne-zabiegi-pomagaja-w-leczeniu-odciskow-,artykul,1732804.html	Y	12
165	https://www.ikea.com/pl/pl/catalog/products/40443835/	N	0

166	https://www.pwn.pl/careers	N	0
167	https://www.eobuwie.com.pl/sneakersy-tommy-hilfiger-branson-8c1-fm0fm00612-twilight-005.html	N	23
168	https://natemat.pl/280829,jacy-sa-wspolczesni-mezczyzni-nie-wszyscy-sa-zli-kobiety-wymagaja-za-duzo	N	34
169	http://rutracker.org/forum/viewtopic.php?t=5370852	N	0
170	https://player.pl/programy-online/kuchenne-rewolucje-odcinki,114	Y	64
171	https://oliviawhen.tumblr.com/post/186738461661/did-you-know-its-doggust-dog-august-my-new	N	0
172	https://login.poczta.home.pl	N	1
173	https://www.mazowieckie.pl/pl/aktualnosci/aktualnosci/38711,Rzez-Woli-pamiec-cywilnych-ofiar-w-75-rocznice.html	N	1
174	https://cbbp1.com	–	–
175	https://lowcygier.pl/promocje-cyfrowe/mass-effect-trilogy-w-wersji-cyfrowej-za-2990-zl-w-muvesklepie-gol/	Y	4
176	https://www.uw.edu.pl/droga-mleczna-w-trzech-wymiarach/	N	0
177	https://z1.fm/song/23170839	N	0
178	https://t.co	N	0
179	https://worldoftanks.eu/en/news/special-offers/Pz-Kpfw-IV-schmalturm-august/	N	0
180	https://www.nazwa.pl/cennik/cennik-wordpress-cloud/	N	0
181	https://zbiornik.com	–	–
182	https://www.targeo.pl	N	0
183	https://www.bing.com/search?q=tom+cruise	N	5
184	http://www.plotek.pl/plotek/7,154063,25061903,kamil-durczok-mial-kupic-wodke-na-pol-godziny-przed-kolizja.html	N	0
185	https://pclab.pl/art81578.html	N	4
186	http://www.benchmark.pl/testy_i_recenzje/gigabyte-x570-aorus-xtreme-i-ssd-m-2-na-pcie-4-0-recenzja.html	N	5
187	https://chaturbate.com	–	–
188	https://www.komputronik.pl/product/341252/goodram-8gb-1x8gb-1600mhz-ddr3-cl11-dimm-.html	N	67
189	https://9gag.com/gag/aQ1P42W	N	1
190	https://www.leroymerlin.pl/podlogi-drewniane-boazerie/panele-deski-parkiety/panele-podlogowe-laminowane/panele-podlogowe-dab-elbrus-ac4-8-mm-home-inspire,p469109,l188.html	N	0
191	https://wetransfer.com	N	0
192	https://smaker.pl/przepis-busz-poledwiczka,174227,kulinarnachwila.html	N	4
193	https://roksa.pl	–	–
194	https://www.rp.pl/Urzednicy/308069970-Kto-moglby-zaplacic-za-loty-marszalka-Marka-Kuchcinskiego.html	N	0
195	https://www.amazon.de/Samsonite-SCure-Spinner-Koffer-Black/dp/B007WQJSGM/	N	274
196	https://badoo.com	N	0
197	https://www.deviantart.com/yeraa/art/Midsummer-808465288	N	4
198	https://inteligo.pl	N	1
199	https://parenting.pl/7-najbardziej-absurdalnych-ciazowych-przesadow	N	11
200	https://www.dotpay.pl/metody-platnosci/platnosci-odroczone/kupuj-teraz/	N	0
201	https://github.com/google	N	0
202	https://www.credit-agricole.pl/klienci-indywidualni	N	0
203	https://naekranie.pl/aktualnosci/lowcy-cieni-bedzie-serial-stacji-nowa-tv-opis-fabuly-i-data-premiery-1565118604-1565119246	N	34
204	https://www.airbnb.pl/rooms/37253565	N	0
205	https://www.skapiec.pl/cat/308-lodowki.html	N	14
206	https://www.nocowanie.pl/noclegi/kolobrzeg/apartamenty/145748/	N	4
207	https://www.apple.com/iphone-xr/	N	0
208	https://jbzdy.net	Y	24
209	https://wbijam.pl	N	0
210	https://www.tauron.pl/dla-domu/prad/zwrot-za-prad	N	0

211	https://www.itaka.pl/wczasy/bulgaria/sloneczny-brzeg/hotel-julia,BOJJULI.html	N	29
212	https://www.zapmeta.com.pl/?q=tom+cruise	N	0
213	https://nk.pl/logowanie	N	78
214	https://www.rmf24.pl/fakty/polska/news-brunon-kwiecien-nie-zyje-zostal-skazany-za-probe-zamachu-na-,nId,3134454	N	7
215	https://www.polsatnews.pl/wiadomosc/2019-08-06/z-panstwowej-mennicy-w-meksyku-zrabowano-zlote-monety-za-ponad-2-mln-usd/	Y	4
216	https://echodnia.eu/swietokrzyskie/wicemarszalek-sejmu-malgorzata-gosiewska-skrytykowala-swietokrzyskich-parlamentarzystow-prawa-i-sprawiedliwosci-za-brak/ar/c1-14330679	Y	12
217	https://www.msn.com/pl-pl/wiadomosci/polska/bestie-z-okuninki-zgwa\T1\lcili-kolegę-kijem-katowali-kilka-godzin/ar-AAFqIOk	N	TO
218	https://www.linkedin.com/	N	1
219	https://sinoptik.pl	N	0
220	https://gfycat.com/snivelingmediocredeinonychus-thanksgiving-comedy	N	8
221	https://www.autocentrum.pl/newsy/wiozla-dwie-osoby-w-bagazniku-na-trzy-miesiace-straci-prawo-jazdy/	N	3
222	https://duckduckgo.com/?q=tom+cruise&ia=web	N	1
223	http://pogodynka.pl/wiadomosci/aktualnosci/1907-22072019-ocena-meteorologiczna-i-hydrologiczna-1343	N	3
224	https://siteadvisor.com	–	–
225	https://bm.pl	N	0
226	https://www.tradedoubler.com/en/publishers/	N	0
227	https://privatbank.ua	N	1
228	https://www.ryanair.com/gb/en/	N	4
229	https://www.zus.pl/o-zus/aktualnosci/-/publisher/aktualnosc/1/w-szczebrzeszynie-zus-brzmi-prosto/2816156	N	0
230	https://www.telemagazyn.pl/film/goracy-towar-502417/	N	20
231	https://login.gov.pl/login/secure	N	0
232	https://www.animezone.pl/gatunki	N	1
233	http://baskino.me/films/boeviki/19454-lyudi-v-chernom-interneshnl.html	N	18
234	https://www.znanylekarz.pl/magdalena-zychowska/dermatolog-lekarz-medycyny-estetycznej/wroclaw	N	TO
235	http://site-pl.com/?reqp=1&reqr=	N	0
236	https://en.wiktionary.org/wiki/amazing	N	0
237	https://best2019-games-web1.com	–	–
238	http://kinogo.eu/9265-mstiteli-final-2019.html	N	0
239	http://lubimyczytac.pl/aktualnosci/rozmowy/12508/szczescie-solo-czyli-o-zwiazku-z-samym-soba-wywiad-z-wanda-zolcinska	Y	4
240	http://www.speedtest.pl/kim_jestesmy	N	0
241	http://ulub.pl/utACbsOip7/bibsypl-100-obrazkow-na-minute	N	0
242	https://www.pyszne.pl/pizzeria-staromiejska	N	4
243	https://www.pinterest.com	N	TO
244	https://fotka.com	N	0
245	https://ex-torrenty.org/torrent/166668	N	0
246	https://www.gov.pl/web/gov/uslugi-dla-obywatela/#wyjazdy-i-wypoczynek	N	0
247	https://www.sadistic.pl/portal/14693	Y	2
248	https://www.praca.gov.pl/eurzad/index.eup#/inneSprawy/wyborUrzedu?dest=EURZAD	N	0
249	https://www.mp.pl/dostepnaczyniowy/ekspert/pytania-do-eksperta/211883,czy-cewnik-dializacyjny-moze-byc-wykorzystany-do-podania-przez-ratownika-lekow-podczas-resuscytacji	N	2
250	https://gratka.pl/motoryzacja/motocykl-yamaha-xvs-650-v-star/oi/12732681	N	17
251	https://vider.info/vid/+fxms588	Y	0
252	https://yandex.ua/search/?lr=143&oprnd=9981694847&text=tom%20cruise	N	0
253	https://www.tekstowo.pl/piosenka,lionel_richie,hello.html	N	4

254	https://www.twojapogoda.pl/wiadomosc/2019-07-12/wedkarze-lapcie-za-wedki-bo-ryby-beda-braly-jak-szalone-tak-dobrych-lowow-w-te-wakacje-juz-nie-bedzie/	Y	13
255	https://www.doz.pl/apteka/p131082-Estabiom_pregna_kapsulki_20_szt.	N	2
256	https://www.upc.pl/internet/kup-internet/	N	21
257	https://www.lidl-sklep.pl/PARKSIDE-Akumulatorowe-urzadzenie-wielofunkcyjne-4-w-1-PKGA-20-Li-B1-bez-akumulatora-i-ladowarki/p100259868	N	0
258	http://lordsfilms.tv/26676-priklyucheniya-paddingtona-2.html	N	0
259	https://www.eset.com/pl/business/	N	1
260	https://www.lidl.pl/pl/Oferta-spozywcza.htm?articleId=66407	N	2
261	https://www.przegladsportowy.pl	Y	20
262	http://search-pl.com/?reqp=1&reqr=	N	0
263	https://www.gov.pl/web/sprawiedliwosc/przelom-w-ksztalceniu-sluzby-wieziennej--slubowanie-studentow-pierwszej-w-historii-uczelni-polskiego-wieziennictwa	N	0
264	https://myanimelist.net/anime/37347/Dungeon_ni_Deai_wo_Motomeru_no_wa_Machigatteiru_Darou_ka_II	N	11
265	https://obywatel.gov.pl/kierowcy-i-pojazdy	N	0
266	https://rozetka.com.ua/xiaomi_redmi_note_7_4_64gb_black/p84686898/	N	10
267	https://www.kurnik.pl/szachy/	N	0
268	http://sklep.cyfrowypolsat.pl/telewizja-satelitarna	N	0
269	https://poradnikprzedsiebiorcy.pl/-trwa-nabor-do-xi-edycji-konkursu-innowator-mazowsza	N	1
270	https://www.e-podroznik.pl	N	0
271	https://www.t-mobile.pl/telefony-i-urzadzenia/telefony/samsung-galaxy-a50-samsung-galaxy-fit/SA-H-B1570.phtml?addoffer=tariprom_40335	N	1
272	https://en.sat24.com/en/forecastimages/europa/forecasttemp	N	0
273	http://home-nav.com/?reqp=1&reqr=	N	0
274	https://www.dropbox.com/?landing=dbv2	N	0
275	https://www.styl.pl/zdrowie/news-najlepsza-woda-dla-ciebie,nId,3135966	N	9
276	http://rutor.info/torrent/707276/ladja_the-rook-01h01-06-iz-08-2019-web-dlrip-lostfilm	N	0
277	https://www.popads.net	N	0
278	https://www.meczyki.pl/newsy/zwrot-akcji-w-sprawie-christiana-eriksena-manchester-united-zrezygnowal-z-walki-o-dunczyka/119664-n	Y	TO
279	https://www.tvp.info/43838474/spanikowalem-pracownik-ochrony-finalu-wosp-w-gdansku-przed-sadem	N	24
280	https://www.instalki.pl/testy/hardware/36923-xiaomi-mi-band-4-test-opaski-sportowej.html	N	1
281	https://www.wroclaw.pl/portal/dla-wygody-mieszkancow-wroclaw-stawia-na-osiedla-kompletne	N	0
282	https://genius.com/a/lana-del-rey-shares-looking-for-america-in-response-to-recent-mass-shootings	N	2
283	https://www.ipla.tv/kanaly-tv	N	0
284	https://play.eune.leagueoflegends.com/en_PL	N	0
285	http://l.ua	N	0
286	https://www.skyscanner.pl	N	TO
287	https://www.wakacje.pl/oferty/grecja/chalkidiki/metamorfosis/simeon-526145.html?od-2019-10-15,7-dni,HB,z-warszawy-modlin	N	3
288	https://vimeo.com/showcase/2380839/video/59207751	N	5
289	https://www.speedtest.net	Y	1
290	https://dorzeczy.pl/kraj/110342/szykuje-sie-najwiekszy-skandal-w-historii-polskiego-show-biznesu-znany-muzyk-jazzowy-oskarzony-o-pedofilie.html	N	0
291	https://www.efortuna.pl/pl/fortuna_z_bonusem/index.html	N	0
292	https://jelenia-gora.lento.pl/poszukujemy-opiekunki-opiekunow,8060605.html	N	0
293	https://serwisy.gazetaprawna.pl/edukacja/artykuly/1425114,500-plus-dla-nauczycieli-8-6-mln-zl-lacznie.html	N	4

294	https://www.cdaction.pl/news-57514/gamescom-2019-final-fantasy-7-remake-i-marvels-avengers-grywalne-dla-zwiedzajacych.html	N	1
295	https://akamaized.net	–	–
296	https://en.savefrom.net	N	0
297	https://mielno.webcamera.pl	Y	251
298	https://www.getinbank.pl	N	2
299	https://prom.ua/p771785763-nastolnyj-futbol-detskij.html	N	205
300	https://gazetawroclawska.pl/wypadek-na-traugutta-dzwig-wyrwal-drzwi-tramwaju-zdjecia/ar/c16-14333595	Y	21