Bachelor thesis
Computing Science

Radboud University

Comparing privacy plugins

Author:
Nick Nibbeling
s4616146

First supervisor/assessor:
Dr. Veelasha Moonsamy
v.moonsamy@cs.ru.nl

Second supervisor:
Dr. Ir. Hugo (H.L.) Jonker
hugo.jonker@ou.nl

June 25, 2019

This page intentionally left blank

Abstract

In today's society, be tracked on the internet is no longer a particular case. Web tracking techniques keep tabs on the user's internet activity while the user is probably not aware of it. In recent years, many new tracking techniques have been added, and this will only become more. Privacy plugins such as Ghostery and Privacy Badger protect you against third-party trackers on the internet. In this study, we analyze privacy plugins by making a two-part comparison. First, we collect general information about every plugin and compare them with each other. Second, we test the performance of the privacy plugins on a large scale of 25,000 popular websites. We do this by retrieving the number of blocked domains/request directly from the privacy plugin itself. We have not seen this in any other study so far. The goal of this two-part comparison is to provide advice to the end-user, so he or she knows which plugin to pick. We have concluded that algorithm-based plugins have an advantage compared to blacklists-based plugins in theory. We also have statistics on the performances of the privacy plugins in practice. These results are used for the recommendation.

Contents

Chapter 1

Introduction

Nowadays, it is not uncommon to be tracked on the internet. Companies have adapted their business model to the collection of user data. This data can be used to create profiles of users. A clear identification of the user can be leveraged to generate money, for example, by placing targeted advertisements. Companies use a tracking mechanism, or merely a tracker, to collect user data. One of the best-known trackers is the cookie. Website owners can use a cookie to store a session identifier or save language preferences. In this context, the cookie is used for session management by a first-party: the website currently visiting.

A cookie can also be used by third parties to track the user. In that case, we speak of third-party tracking: tracking is done by a different website than the one you are visiting. Third-party tracking, in particular, infringes the privacy and it is therefore wise that the user arm against this.

Fortunately, this can be done by privacy plugins. Plugins serve as an extension of the web browser. By privacy plugins, we mean plugins that protect against tracking. When researching privacy plugins, we encounter three different types of privacy plugins: plugins that are specialized in blocking advertisements, plugins that completely isolate each tab of a browser (containers), and plugins that blocks trackers. The last mentioned category is investigated in this study. For convenience, we continue to call them privacy plugins. The other categories are not further discussed, as they serve different purposes which are not relevant to this study. Blocking advertisement is not the same as blocking trackers. A tracker may be an advertisement, but not every advertisement is a tracker. The second category isolates tabs, so tracking between tabs should be harder. However, a user can still be tracked in an individual tab, regardless of the isolation.

In April 2017, Boumans published his bachelor thesis [6] *'Web tracking and current countermeasures'*. He discusses the tracking techniques of that time, privacy measures in web browsers, and browser extensions for privacy. In the end, Boumans makes a recommendation to the end user what would

be a good configuration for users to minimize web tracking, but not lose usability. Not only Boumans has performed a comparison of plugins. There are many studies [2, 18, 15, 17, 5, 26, 19, 23, 18] that compare privacy plugins and provides recommendations.

Methodology
However, the internet has not been idle and changed over time. For example, more privacy plugins were added, privacy plugins were updated, and new tracking techniques were discovered. In this thesis, we dive into the world of online tracking. This is a specific part of online privacy. We look for privacy plugins that protect the user against tracking. There exist many privacy plugins that differ widely in performance and usability. That is why it is essential that internet users get a good recommendation of which plugins they should use. In particular, we make a recommendation of which plugin offers the best privacy protection by the number of blocked trackers according to plugins themselves. So the data comes directly from the plugin. To our knowledge, this is the first study that uses this method. We compare the claims of several plugins that Boumans and many other studies also compared and see if they produce differences. Boumans study is theoretical [6], while the others are more practical [15, 17, 5, 26, 19, 23, 18]. We do both theoretical and practical comparisons, which sets us apart from many other studies. We examine the plugins and also provide a clear description of how these plugins offer privacy. Besides, we use a combination of Firefox and Selenium to run the plugins over 25,000 websites for practical comparison. We can validate our theoretical conclusions by performing experiments with privacy plugins in practice. The methodology will be described in more detail in Chapter 4 and 6.

Contributions & limitations
A complete list of contributions can found below:

- Insight in the different types of definitions for privacy
- Analysis of privacy plugins from a privacy perspective
- Theoretical and practical comparison of privacy plugins and their anti-tracking methods
- Providing a method to extract data from privacy plugins
- Large scale analysis of the performance of privacy plugins
- Firefox Content Blocking performance tests
- Number of trackers compared on popular and less popular websites
- Recent recommendation of privacy plugins

Before we go further, it is important to clarify the limitations and scope of this thesis:

1. Not all privacy plugins have taken into account, only those who meet the selection criteria discussed in Section 4.1.

2. No scientific tools/plugins were used in this thesis. This has to do with usability. Experience shows that scientific tools/plugins are harder to use.

3. We assume that privacy plugins show the correct results.

4. We only use Firefox as a web browser. Firefox is easy to configure, offers a large collection of plugins, works stable in combination with Selenium, is focused on privacy, and is used for many plugin comparison studies [15, 17, 5, 26, 19, 23, 18].

5. Regarding the definition of a tracker, we use the same as Englehardt and Narayanan (2016) [8]. We do not determine for ourselves if a domain/URL is a tracker but assume that a domain/URL is a tracker if a privacy plugin blocks it.

6. Ad-blockers are not covered in this thesis because they serve a different purpose: blocking advertisements and not necessarily trackers.

Overview
This thesis discusses essential privacy subjects in Chapter 2. We discuss some related work in Chapter 3. General information about the investigated privacy plugins is given in Chapter 4. The plugins are theoretically examined in Chapter 5 and practically examined in Chapter 6. In Chapter 7, we give a conclusion and recommendation of privacy plugins. Eventually, we discuss some future work in Chapter 8.

Chapter 2

What is Privacy?

In this chapter, we try to approach the question 'What is privacy?' from different angles to finally come to a suitable standard for testing the privacy plugins.

2.1 Laws affecting privacy

In most countries, the right to privacy is a human right. However, what is privacy according to a judicial approach? First of all, we decided to reduce our scope to European Law since this will cover multiple countries. The European Convention on Human Rights (ECHR) [20] is an international treaty to protect human rights and political freedoms in Europe. It was founded in 1950. According to Art. 8(1) of the ECHR, *'everyone has the right to respect for his private and family life, his home, and his correspondence'*. At that time, there was no use of the internet, so 'correspondence' mainly concerned written communications like letters and telegrams. Many years later, rights for EU citizens were established, which are called Charter of Fundamental Rights of the European Union (CFREU). The latest version was published on October 26, 2012. Pursuant to Art. 7 of the CFREU [25], privacy means that *'everyone has the right to respect for his or her private and family life, home, and communications'*. Now, 'correspondence' has been replaced by 'communications', we assume this is a result of which internet communication is now also covered. This means that within the EU, citizens have the right to respect for their communications on the internet. Before going further, it is wise to have a look at the General Data Protection Regulation (GDPR)[9]. The GDPR is a regulation in EU law that was adopted in 2016. It has the purpose to guarantee protection of personal data and privacy within the European Union. This regulation applies also to the European Economic Area. The GDPR provides some definitions that can be useful in the following sections.

2.2 Privacy

In this section, we discuss the term privacy. Thanks to the articles [13, 28, 9], we can formulate a clear definition of privacy:

> *Privacy: an action or communication M of a subject S that cannot be traced back to S by observer O*

Subject S is a user of a browser. Observer O is an attacker and is discussed further in Section 2.3. Furthermore, privacy is about content, maintaining confidentiality, and keeping secrets. For example, you are private in your own home, as no one knows what you do in your own home. You are not anonymous as everyone knows that you live there. This is an important observation, which must be kept in mind. The purpose of a privacy plugin that must guarantee privacy is to keep the actions and activities secret, but not necessarily your identity. This can be established by blocking trackers.

2.3 Attacker model

Someone who tries to infringe somebodies privacy can be defined as an attacker. We define two types of attackers relevant to this study. The local attacker and the remote attacker. Asaka, Onabura, Inoue, and Goto (2010) provide definitions for local and remote attacks that we can use as starting point[3].

- **Local attacker** has direct access to a machine. This can be a family member of the owner of the machine.
- **Remote attacker** has **no** no direct access to a machine. This is someone who tries to access the machine from a remote location.

Web tracking can be seen as a variant of remote tracking, and for this reason, the focus in this research lies on the remote attacker. The remote attacker can also be divided into subcategories. A remote attacker can, for example, be a first party tracker, a third party tracker, a combination of both or an ISP (Internet Service Provider). We discuss this further in Section 2.5.

2.4 What makes a tracker?

We already mentioned the term tracker a few times, but we have not explained what it means? A tracker can have multiple meanings. It can be

considered as a party, mechanism, or domain. Within the scope of this research, it is important to distinguish what we mean by a tracker.

We start with parties. Among parties, we principally mean companies that are focused on collecting user data. Parties make use of mechanisms to track a user across the internet; for that reason, we call them trackers or tracking parties. The most important parties are first and third parties; they are discussed in Section 2.5.

In addition to parties, the mechanisms they used can also be considered as trackers. A tracking mechanism makes it possible to collect user data. Some examples of tracking mechanisms are cookies, tracking pixels, and fingerprinting. It is basically a piece of code.

Domains and URLs are often also regarded as trackers. Blacklists or blocklists used by plugins and browsers contain domains and/or URLs. A URL is a specific path to the location of a resource on the internet, while a domain usually maps to some web space. Tracking parties need a domain to communicate over the internet. If you block a domain, then there is no more data traffic possible between the domain and the tracking mechanism. For this reason, blocking a domain is seen as blocking a tracker.

However, we need the definition even more clearly for this study. Englehardt and Narayanan made the same decision in their research: Online Tracking: A 1-million-site Measurement and Analysis [8]. They not simply classifying domains as trackers or non-trackers, but assume a domain is a tracker when the plugin has blocked this domain. We make the same assumption in this study. We do not determine for ourselves whether a domain is a tracker but assume that a domain is a tracker if a privacy plugin blocks it.

2.5 First and third-party trackers

In Section 2.3, we already mentioned that there are first and third-party trackers on the internet. First-party tracking is done by the website you are currently visiting. It has, in general, a functional purpose, such as maintaining your shopping cart or remembering your preferences. A third-party tracker instead is a tracker set by a different webpage than the one you are currently visiting. It has the purpose to track you across multiple websites en monitor your behavior. Third-party trackers are often used to create an image of the user, after which advertisements can be tailored to the user. Within the scope of this research, we assume that the attacker is almost always a third-party tracker. We have to keep in mind it can also be a first and third-party tracker at the same time. For example, Facebook is a first party tracker when visiting the Facebook website, but a third party tracker when a Facebook button is embedded in a non-Facebook webpage.

2.6 How do privacy plugins provide privacy?

In this section, we answer the question of how privacy plugins can provide privacy. We do not discuss privacy for each plugin but want to show precisely how privacy plugins can add something together to protect privacy. First, we repeat the definition of privacy: *an action or communication M of a subject S that cannot be traced back to S by observer O.*

A privacy plugin meets the requirement of privacy by simply blocking the tracker. In this way, it is no longer possible for a tracker to see what the user is doing because the communication between the tracker and the third party is interrupted. However, a privacy plugin has to establish privacy without breaking a website, so there is a trade-off between privacy and usability [27]. It is essential to minimize websites breaks to not only satisfy the user but also prevent the user from solving problems themselves. When a user notices that there is something wrong with the functionality of a website, he might whitelist the website, which causes gaps in security protection.

2.7 Private browsing and Tor Browser

Private browsing
Privacy plugins are not the only method that protects your privacy. There is also something called private browsing. According to Firefox, private browsing means using Firefox without saving history[1]. It works the same as normal browsing, but your data will be erased after you end the session. For Firefox, the data that will erased is limited to **Visited pages**, **Form and Search Bar entries**, **Passwords**, **Download List entries**, **Cookies**, **Cached Web Content**, **Offline Web Content**, and **User Data**. Deleting all this data decrease the change of being tracked across the internet. Especially cookies are a common tracking method used by third parties.

It is wise to keep in mind that private browsing does not protect against all forms of tracking. For example, browser fingerprinting is still possible.

Private browsing is especially effective against a local attacker. Someone who has direct access to your device cannot retrieve your browsing history when you use private tabs.

Not only Firefox provides private browsing. Almost every modern browser provide private browsing or something similar to it.

Tor Browser
Besides private browsing, there is another method you can use to protect

[1] `https://support.mozilla.org/en-US/kb/private-browsing-use-firefox-without-history` accessed on June 8, 2019

your privacy. The Tor Browser[2] makes use of onion routing. This means that it routes traffic through multiple servers. At every server, the traffic is encrypted. The goal of Tor is to provide privacy for everyone on the internet.

The Tor Browser is very effective against trackers since they cannot follow you through the servers. Even fingerprinting will be harder since all users look the same when using Tor. It also prevents against monitoring your connection. This becomes very handy in countries with censorship.

The Tor Browser is a useful tool when it concerning your privacy. Since it is not a privacy plugin, it will not be compared with the plugins. However, it is certainly worth mentioning the existence of the Tor Browser.

[2] `https://www.torproject.org/` accessed on June 8, 2019

Chapter 3

Related Work

In the past few years, many research has been done in the area of online tracking. A large number of these studies rely on OpenWPM[1], 31 to be precise[2]. OpenWPM is an open-source web privacy measurement framework based on Firefox and Selenium. The studies vary from detecting web trackers to comparing anti-tracking techniques.

Altaweel, Good, and Hoofnagle (2015) survey online tracking mechanisms used by top 25,000 most popular websites[2]. They use OpenWPM to simulate browsing behavior. They also performed a Web Privacy Census in 2011 and 2012. One of the results is that cookies are one of the most used tracking methods. In 2015, they collected twice as many cookies on the top 100 most popular websites as detected in 2012. Eighty-three percent of the cookies they found, consist of third party cookies. They also discovered that Google has the biggest tracking infrastructure based on the top 1000 most popular websites. The infrastructure is so large that the surveillance of Google only can be compared with the surveillance of an Internet Service Provider. This shows that trackers can be a major threat to privacy.

A year later, Englehardt and Narayanan (2016) perform a large en detailed measurement of online tracking [8]. Therefore, they use the Alexa top 1 million websites. The goal was to detect tracking techniques that are active on each website. They took advantage of OpenWPM and encounter several new fingerprinting techniques. Besides that, they show that tracking protection works. Ghostery, for example, seems very effective in blocking trackers. It reduces the average number of third-parties from 17.7 to 3.3. This is important for our research because if tracking protection does not work, comparing privacy plugins is useless.

According to Yu, Macbeth, Modi and Pujol [27] there are two main anti-tracking techniques. The most common approach is to use a blocklist of trackers, also called blacklist. Blacklists contain mainly domains and/or

[1] `https://github.com/mozilla/OpenWPM`

[2] `https://webtransparency.cs.princeton.edu/webcensus/`

URLs of third-party trackers. If a third party that is on the blacklist tries to communicate with the user's browser, the communication is blocked. This is usually limited to cookies or the complete HTTP request.

The other anti-tracking technique they mention is the use of an algorithm for identifying trackers. This technique relies on heuristics and is not dependent on a blacklist. It determines based on behavior whether a domain is a tracker.

Yu et al. [27] implemented their own algorithmic solution. They decide which data is safe or unsafe to send to third-parties, rather than identifying and blocking trackers.

Studies we are also interested in are privacy plugin comparison studies. Mazel, Garnier, and Fukuda (2017) provide an overview of previous privacy plugin comparison studies [18]. Between two and seven plugins were used for these studies [15, 17, 5, 26, 19, 23, 18]. All studies have a unique method for collecting data. A summary of the methods used by each study is given below:

- Krishnamurthy et al. [15] use the Firefox Pagestats plugin and a proxy to collect HTTP request and responses.
- Mayer et al. [17] use FourthParty[3] to collect HTTP requests, responses and cookies. FourthParty is a dynamic web content measuring platform build on Firefox.
- Balebako et al. [5] capture text advertisements on a page with browser history. Then they compare them with text advertisements captured without browser history.
- Wills et al. [26] use Firefox in combination with Selenium. They use a Firefox plugin called HTTP LiveHeaders to collect all HTTP requests.
- Merzdovnik et al. [19] developed a framework called CRAWLIUM. This framework can run multiple browser configurations in parallel. It is based on Firefox in combination with Selenium and mitmproxy. It loads web pages and collects HTTP requests and responses.
- Traverso et al. [23] use Firefox in combination with Selenium and dump statistics via HAR files. These files contain HTTP tracing information.
- Mazel et al. [18] use OpenWPM. They collect HTTP requests, the number of third-party domains, the number of profile cookies, and the total amount of data transferred.

[3] `http://fourthparty.info/`

Remarkable is that most of these studies collect requests, responses, domains, and cookies. What makes this study unique is that we collect data from the plugins themselves. We have not seen this in any other study so far. In comparison to the other studies, our method is much more efficient. We only have to extract a number from the plugins and do not have to determine if something is a tracker since the plugin already does that.

Mazel et al. (2017) also provide a table stating which privacy plugin is used in which study. AdBlock Plus, uBlockOrigin, Ghostery, Privacy Badger, and Disconnect are commonly used plugins. Boumans also use them in his bachelor thesis [6]. This thesis was published in early 2017, and over the past, new techniques and countermeasures are developed. He compares the plugins based on blocking techniques, user-friendliness, and data integrity. In comparison to earlier mentioned studies, Boumans did not measure plugins in practice.

According to Boumans, almost all privacy plugins he looked into in his thesis, make use of a blacklist. He also mentioned that using an algorithm to block trackers has more potential. An algorithm does not rely on a blacklist that has to be updated. Boumans also makes a recommendation. Firefox is recommended since a commercial party does not own it, has many plugins and many configuration options.

As for the plugins, Privacy Badger and uBlock Origin are recommended. He advises to use them together. uBlock Origin will block most trackers thanks to its blacklists. Privacy Badger's algorithm can block the remaining trackers.

Chapter 4

Privacy Plugins

This chapter will cover general information about the investigated privacy plugins. First, we justify our selection of privacy plugins in Section 4.1. Next, we discuss anti-tracking techniques of privacy plugins in Section 4.2. This is followed by Section 4.3 where we discuss the methodology. In Section 4.4 we theoretically examine the selected plugins. In the end, we focus briefly on the Firefox built-in privacy plugin in Section 4.5.

4.1 Justification for selection of plugins

Before we can perform a comparison, we must have a selection of privacy plugins. There are several options for selecting plugins. An easy way is to select them randomly or by popularity. However, that is not how we are going to do it. We have established several criteria that a plugin has to meet:

- A privacy plugin must be focused on blocking trackers. From Section 2.6 follows that a privacy plugin meets the requirement of privacy by simply blocking the tracker.
- The plugin has to keep rid of the number of blocked trackers. This number is the data we use to compare the plugins in practice.
- The plugin has to be available for multiple browsers, but Firefox in particular. Firefox is used for the practical experiment. Availability for multiple does not provide restrictions on choosing a browser for the end-users. They have the freedom to choose a browser they prefer.

We collect the plugins from two places. The first place is Bits of Freedom[1]. This is an independent Dutch foundation that stands up for digital civil

[1] `https://toolbox.bitsoffreedom.nl/overzicht/categorie/add-ons/` accessed on 21 November 2018

rights. Bits of Freedom provides a toolbox that contains recommended privacy plugins. It consists of tracker blockers, container plugins, password managers, and some other types of privacy plugins. Since we are interested in blocking trackers, we filter the plugins on blocking trackers. This results in the following plugins: Privacy Badger, uBlock Origin, Better, Disconnect, and Ghostery. Better is only available for Safari, so this plugin is excluded. The other plugins meet all requirements.

The next list we use is from the Firefox add-ons store[2]. Firefox features these plugins. We apply the criteria in the same way we do for Bits of Freedom. The following plugins will remain: uBlock Origin, Ghostery, Privacy Badger, Disconnect, and Privacy Possum. Plugins featured by Firefox will probably change over time, so a tutorial on how they get to this list has been added to Appendix A.1.

Remarkably, these privacy plugins are often used in comparison studies. From Chapter 3 follows that AdBlock Plus, uBlock Origin, Ghostery, Privacy Badger, and Disconnect are commonly used plugins for these studies. Some of the studies add AdBlock Plus to their set of privacy plugins. In this research, we do not. The reason for this is that AdBlock Plus is more known as an AdBlocker rather than a universal blocker. On the other hand, uBlock Origin is also known as an adblocker, but offers, in comparison to AdBlock Plus, more protection options against third-party tracking. Besides that, uBlock Origin does not label itself as an AdBlocker but as a 'wide-spectrum' blocker.

Privacy Possum is not used in other privacy plugin comparison studies and has no recommendation by Bits of Freedom. Nevertheless, we add it to the selection of privacy plugins since it should be an improved version of Privacy Badger. This is due to the fact that an old Privacy Badger developer has developed the plugin. The ex-developer created his own plugin since it was not possible to make major changes to Privacy Badger due to its policy. Therefore, it is worthwhile to compare these two plugins with each other. We examine Privacy Possum only theoretical since the way of counting blocked trackers cannot be compared to any other plugin.

Furthermore, all plugins from Table 4.1 are examined in Firefox. Installing a plugin in Firefox is a straightforward action which takes less than a second; this holds for all plugins in this research. In the next section, common features and anti-tracking techniques will be discussed. The selected plugins are listed in Table 4.1.

[2] `https://addons.mozilla.org/en-US/firefox/search/?category=privacy-security&featured=true&sort=recommended%2Cusers&type=extension` accessed on November 21, 2018

Privacy plugins
Ghostery
Privacy Badger
Disconnect
uBlock Origin
Privacy Possum (only theoretical)

Table 4.1: Privacy plugins

4.2 Anti-tracking techniques

How the plugins determine what should be blocked depends on the implementation of the plugin. The anti-tracking techniques are limited to the privacy plugins in Table 4.1. During the investigation of the plugins, we came across three anti-tracking techniques: Blocking trackers with a blacklist, blocking trackers with an algorithm, and anonymizing personal information in a request. The first two techniques have already been briefly discussed in Chapter 3. The last technique is only used by Ghostery. It anonymizes all personal information in a request. Since it does not block trackers, we do not further discuss this technique. For clarity, a summary of the two main anti-tracking techniques is given below:

1. Blocking trackers using a blacklist. A blacklist is a list of third-party trackers. If a domain or URL occurs on this list, it will be blocked by the plugin.

2. Blocking trackers using an algorithm. This technique relies on heuristics and is not dependent on a blacklist. It determines based on behavior whether a domain is a tracker. So if the plugin determines you are being tracked, it blocks the domain or URL.

Advantages and disadvantages of these two anti-tracking techniques are discussed in Chapter 5.

You can determine if something is a tracker on different levels. The privacy plugins in this study do this at the domain or URL level. Since a URL provides more detailed information of a specific location than a domain, blocking trackers on URL level lead to fewer false positives and less site breakage in comparison with blocking trackers on a domain level. This is because there is a chance some useful resources cannot be loaded if the domain is blocked. Blocking on URL level only blocks the resources you do not want.

4.3 Methodology

In Chapter 3, we have become acquainted with some privacy plugin comparison studies. Most of them are practical [15, 17, 5, 26, 19, 23, 18]. Only Boumans study was theoretical [6]. In this thesis, we make both theoretical and practical comparison, which sets us apart from many other studies. We can validate our theoretical conclusions by performing experiments with privacy plugins in practice.

Before we can make a practical comparison in the next chapter, we had to gather information about the privacy plugins. Section 4.1 announced which plugins are being investigated, namely: Ghostery, Privacy Badger, Disconnect, uBlock Origin, and Privacy Possum. These five plugins have several common features but differ in some aspects. We evaluated the plugins briefly by discussing the following aspects:

- **General information:** Some noteworthy information, such as who is the developer and what is the reputation of the plugin. Furthermore, is the plugin itself a risk for the user?
- **Anti-tracking technique:** Which anti-tracking techniques are used by the plugin?
- **Openness:** Is the source code open source or not?
- **Usability:** Is the plugin user-friendly, and what are the options/features for consumers?

The criteria are applied to privacy plugins in Section 4.4. The actual theoretical comparison of the privacy plugins takes place in Chapter 5.

Remark
Boumans had also evaluated a number of these plugins two years ago. This can result in an overlap of information when plugins were not updated. However, our information is gathered **independently** of Boumans, and we use the latest information. So we do not use his study for gathering information about privacy plugins. The only purpose is to find differences between plugins now and two years ago.

4.4 Privacy plugins

We start by discussing Ghostery in Section 4.4.1. This plugin has broader general information compared to the other third part blockers since Ghostery has a negative past when it comes to the use of user data. Privacy Badger, Disconnect, uBlock Origin and Privacy Possum are discussed respectively in Section 4.4.2, Section 4.4.3, Section 4.4.4 and Section 4.4.5.

4.4.1 Ghostery

The information sources for this plugin are the Firefox add-ons store [10] and the official Ghostery FAQ page [11] both accessed on December 6, 2018.

General information
Ghostery has a remarkable history concerning other privacy plugins. In the past few years, Ghostery was not as reliable when it comes to the use of user data [16]. Before privacy-oriented web browser Cliqz in 2017 acquired it, Ghostery was owned by Evidon and used a skeptical business model. Users can choose the opt-in program to share their data on tracking technologies that they encounter. This data then sold to third-parties. Also, Ghostery was not open source at this time.

After Ghostery has become part of Cliqz[3], it seems like they have become more reliable. In March 2018, Ghostery went open source by posting all its code on GitHub[4]. Besides that, Ghostery has moved to the German jurisdiction, where it must comply with the regulation of the GDPR [9]. The company is now owned and funded by Cliqz and makes no money from the free extension. Since Ghostery was taken over by Cliqz, it will evaluate different business models. Ghostery now offers Ghostery Plus, a paid version of its browser extension. Furthermore, users can still participate in the opt-in program, but Ghostery claims that it did not share any user data with other companies, including Evidon.

It seems that Ghostery has made significant changes since the acquisition in 2017. That they have gone open source shows that they do not want to hide anything anymore. Unfortunately, the past will not be forgotten, and it remains the question of whether Ghostery can be trusted entirely.

Anti-tracking technique
As the primary technique, Ghostery makes use of a blacklist, a widely used technique by privacy plugins. At January 22, 2019, it has a library of over 4500 trackers from over 2600 companies. This library will update continually. However, the Ghostery blacklist is not open source. It purposely remains proprietary to Ghostery. Therefore, we can only trust Ghostery on their word. The trackers are blocked on a URL level. Besides a blacklist, Ghostery also has a feature called Enhanced Anti-Tracking. This feature keeps your browser anonymous to protect your privacy. It supplements Ghostery's blacklist by anonymizing all personal information in a request. A tracker may be allowed to execute, but your privacy is still protected.

Openness
As previously discussed, Ghostery has been open source since March 2018.

[3] `https://cliqz.com/en/magazine/press-release-cliqz-acquires-ghostery`
[4] `https://github.com/ghostery/ghostery-extension`

The code can be found on their GitHub[5].

Usability
While using Ghostery, the user can choose between a simple or a detailed interface. The user can easily switch between the two interfaces. The simple interface can be found on the left in Figure 4.1. It shows how many trackers are detected and which of them are blocked. In addition, it also shows which category the tracker belongs to. From this interface, it is possible to trust or not trust a website or pause Ghostery for a certain time. The detailed interface shows the same as the simple interface with the addition that it also provides information about the trackers. The detailed interface can be seen on the right of figure 4.1. It is possible to block a specific tracker from here. When you start Ghostery for the first time, it uses the default settings. This means that it blocks trackers that match a tracker from the blacklist, allows trackers created by site owners (first party trackers) and blocked a Ghostery icon will replace social media buttons. These are the defaults settings and are easy to modify in the Ghostery menu. You can also indicate which notifications you want to get, add trackers by your own and much more.

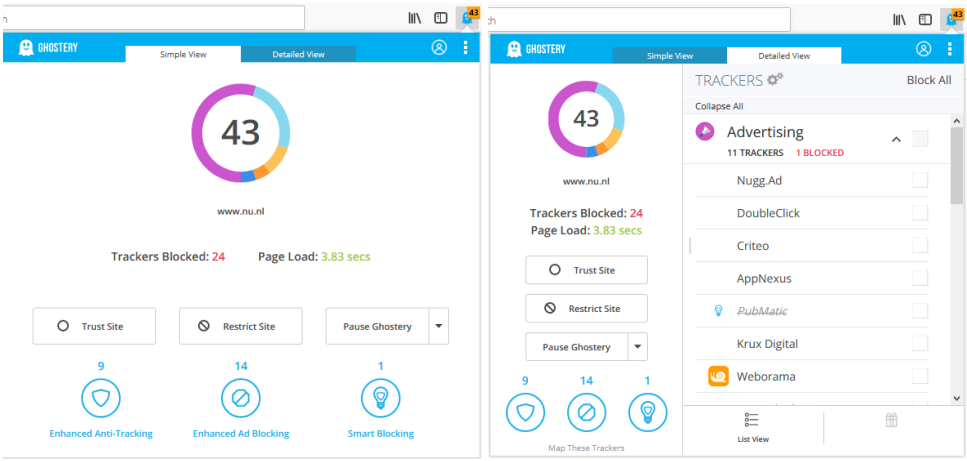


Figure 4.1: Ghostery interface

4.4.2 Privacy Badger

The information source for this addon is the official Privacy Badger FAQ page [4] accessed on January 30, 2019.

[5] `https://github.com/ghostery`

General information
Privacy Badger is developed by the Electronic Frontier Foundation (EFF). The EFF is an American non-profit foundation that deals with civil rights in cyberspace, like self-determination rights of internet users and freedom of expression. Privacy Badger also does not have a business model, which some other plugins have. For this reason, Privacy Badger seems a reliable plugin when it comes to the privacy of the user.

Anti-tracking technique
In contrast with Ghostery, Privacy Badger is algorithm-based and blocks trackers on the domain level. This means that Privacy Badger uses an algorithm to determine when to block a domain. Even though Privacy Badger does not use a blacklist, it does use a yellowlist. A yellowlist contains websites that are known for providing essential third party resources[6]. Instead of the entire website, only cookies are blocked from these websites. Privacy Badger shows such websites with a yellow slider in Figure 4.2. Their yellowlist contain 645 domains[7]. It is the intention that in the long term, the yellowlist will disappear if the parties on the list respect Do Not Track signal.

Openness
Privacy Badger has always been open source. The code can be found on GitHub[8].

Usability
When you start Privacy Badger for the first time, it uses the default settings. This means it replaces social widgets, send websites a "Do Not Track" signal and check if third-party domains comply with EFF's Do Not Track policy[9]. Privacy Badger has a straightforward interface where you can easily adjust the sliders to allow a domain, block a domain, or only block the cookies of a domain (see Figure 4.2). Privacy Badger also has some useful options, such as providing access to the detected tracking domains and filter them. You also have the option to add domains. Altogether Privacy Badger provides a very suitable plugin for all kinds of users.

[6] `https://github.com/EFForg/privacybadger/blob/master/doc/yellowlist-criteria.md`
[7] `https://github.com/EFForg/privacybadger/blob/08b61e85e5c361fe8b535ec9e33950431e28632a/src/data/yellowlist.txt` accessed on January 23, 2019
[8] `https://github.com/EFForg/privacybadger`
[9] `https://www.eff.org/dnt-policy`

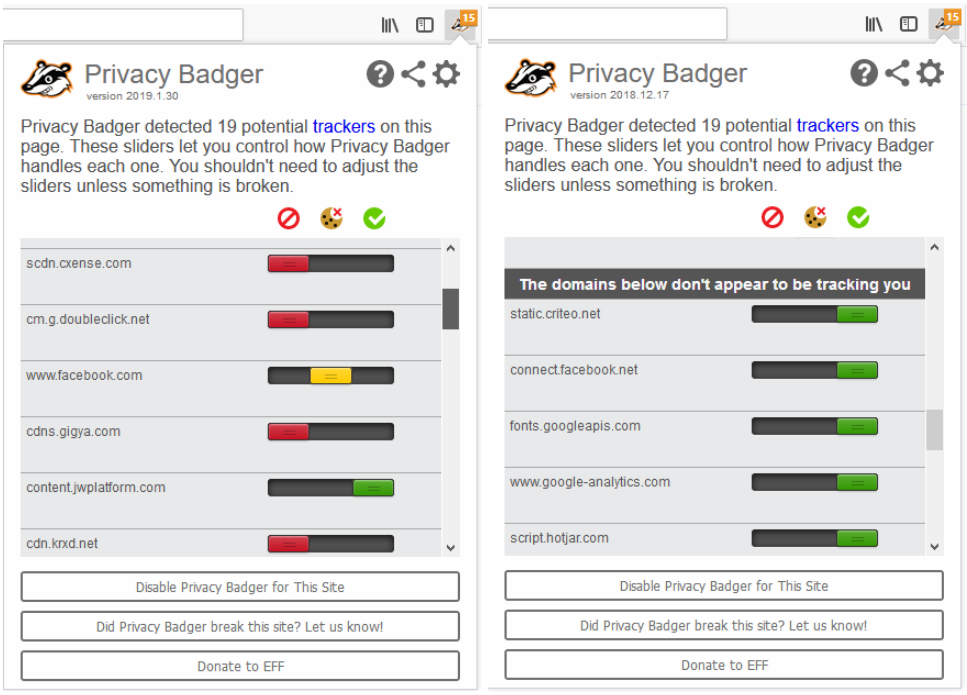


Figure 4.2: Privacy Badger interface

4.4.3 Disconnect

The information source for this addon is the Disconnect website[7] and their GitHub page[10] accessed on January 30, 2019.

General information
Disconnect was founded in 2011 by Brian Kennish and Casey Oppenheim[11]. Kennish is a former Google and DoubleClick engineer. DoubleClick is a company that developed and provided internet ad serving services. Oppenheim is a consumer- and privacy-rights advocate and attorney. Together they form a solid mix of technology and rights. Disconnect uses a business model by selling paid versions of their application.

Anti-tracking technique
Disconnect makes use of several blacklists[12]. The open source list can be

[10] https://github.com/disconnectme/disconnect
[11] https://www.crunchbase.com/organization/disconnect#section-overview
[12] https://disconnect.me/trackerprotection

found on their GitHub[13] and is also used by Firefox and many other popular privacy tools. Currently, it contains 1183 domains. Another open source list[14] contains 2568 domains. Together they contain 3194 unique domains. It is not known whether Disconnect uses even more blacklists. Disconnect does not block parties that deliver content by default. This is because these parties may track you, but blocking can lead to break pages. Disconnect blocks trackers on the domain level.

Openness
Disconnect is open source and the code can be found on their GitHub page [15].

Usability
Disconnect uses a clear interface. The main interface is shown on the left in Figure 4.3. For a clear overview of the meaning of the buttons of the Disconnect interface, it is recommended to check out the help page[16]. The help page briefly explains every piece of the interface. In short, Disconnect shows in the upper right corner, the total number of tracking requests on the current visiting page. Furthermore, it shows the number of tracking requests per company. Facebook, Google, and Twitter are separately shown so they can easily be blocked and unblocked. Green means blocked, and grey means unblocked. There is an option to graph the request on the page you are visiting, an example on the right in Figure 4.3. During the use of Disconnect, we hardly experienced any delays in loading a website. Disconnect feels very fast and has a simple, straightforward interface which makes it ideal for all types of users.

[13] `https://github.com/disconnectme/disconnect-tracking-protection` accessed on January 23, 2019
[14] `https://disconnect.me/trackerprotection/blocked` used by Disconnect accessed on January 23, 2019
[15] `https://github.com/disconnectme/disconnect`
[16] `https://disconnect.me/disconnect/help`

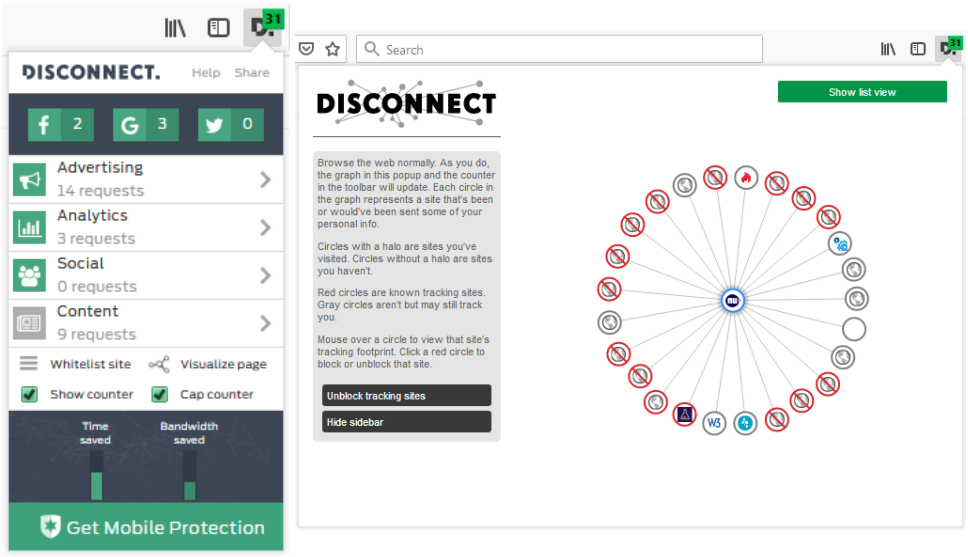


Figure 4.3: Disconnect interface

4.4.4 uBlock Origin

The information source for this addon is the official uBlock Origin GitHub page [24] accessed on May 2, 2019.

General information

uBlock Origin was founded in 2014 by Raymond Hill. The project started as uBlock. However, since Chris Aljoudi forked it in 2015, founder Raymond Hill decided to go further under the name uBlock Origin. uBlock Origin is entirely free and is not looking for donations. It does not have a business model and clearly states that this plugin is nothing without the lists of filters. They appreciate the people who contribute to the filter lists.

The plugin does not see itself as an adblocker, but as a 'wide-spectrum' blocker. It enables *EasyList*, *Peter Lowe's Adservers*, *EasyPrivacy* and *Malware domains* by default. uBlock Origin tries to distinguish itself in terms of memory usage and speed. We do not test this.

Anti-tracking technique

uBlock Origin makes use of several blacklists. This includes the lists mentioned above. uBlock Origin also provides their own lists. These contribute to the protection of privacy, resource abuse, and malicious actions. On June 5, 2019, the default lists contain together 99.950 network filters and 47.009 cosmetic filters (element hiding). The lists are updated very regularly, which

is important since uBlock Origin is almost entirely dependent on the lists. It does not use an algorithm for blocking trackers. uBlock Origin blocks trackers on the URL level.

Openness
uBlock Origin is open source. The code can be found on GitHub[17].

Usability
After installing uBlock Origin, you are protected by the default settings right away. The interface contains a prominent I/O symbol (power symbol) which the plugin can be switched on and off. Furthermore, uBlock Origin shows how many requests are blocked and how many domains are connected. By clicking on them, a screen is opened which contain all the current domains. A very detailed description of uBlock Origins interface can be found on their GitHub page[18].

uBlock Origin offers many options. It is possible to enable a color-blind friendly mode for people with colorblindness. Besides, there are several options to block different types of media on websites. Most important is that you can easily add new filter lists to uBlock Origin. Alternatively, disable lists.

Although we have not tested uBlock Origin for speed and memory usage, it still feels very smooth to use. uBlock Origin looks very simple but offers many options, and has a straightforward interface. It is ideal for all types of users, whether it comes to computer amateurs or experts.

[17] `https://github.com/gorhill/uBlock`

[18] `https://github.com/gorhill/uBlock/wiki/Quick-guide:-popup-user-interface`

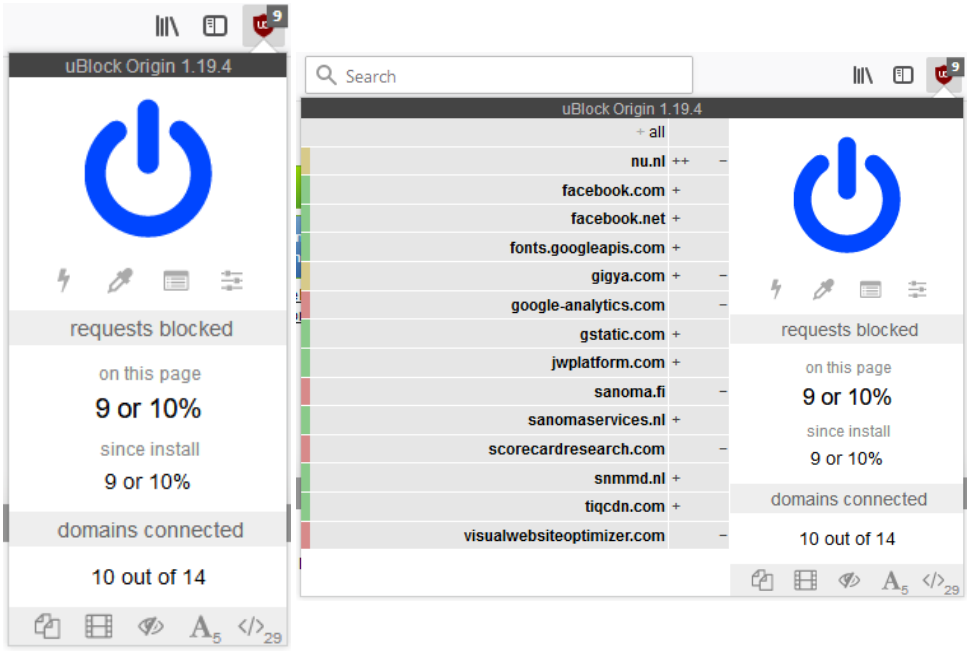


Figure 4.4: uBlock Origin interface

4.4.5 Privacy Possum

The information for this plugin comes from the official GitHub [21] page visited on December 4, 2018.

General information

Privacy Possum was developed by Blake Griffith, one of the developers of Privacy Badger. Since adding new features was difficult with the current architecture of Privacy Badger, Griffith decides to develop his own privacy plugin. Privacy Possum clearly states in its privacy policy that it does not collect or send data to any server.

What makes Privacy Possum unique compared to the other privacy plugins is that it does not have a threat model. Privacy Possum looks at the tracking problem from an economic perspective. Instead of wondering whether a tracking company can circumvent an anti-tracking measure, they decide how much money it would cost for a company to implement this. If the costs are higher then the effects, a company may pay no attention to it.

Anti-tracking technique

Privacy Possum is algorithm-based, just like Privacy Badger. Currently, it blocks the following tracking techniques:

- Browser Fingerprinting
- Cookie Tracking
- Etag Tracking
- Referer headers

Openness
Privacy Possum is open source. The code can be found on GitHub[19].

Usability
Once installed, Privacy Possum immediately starts with blocking trackers. Figure 4.5 shows what the interface looks like. As you can see, the interface is very straightforward and shows which sorts of tracking are blocked and how many of them. Unfortunately, it does not display the sources of the tracking headers. On the other hand, the source of the browser fingerprinting script is displayed. There is also no possibility to set additional options. Besides that, Privacy Possum is a plugin that is easy to use. The only things you can do are enable or disable the plugin and show or not show blocked tracking in the interface. Typically, a plugin for the non-demanding user.

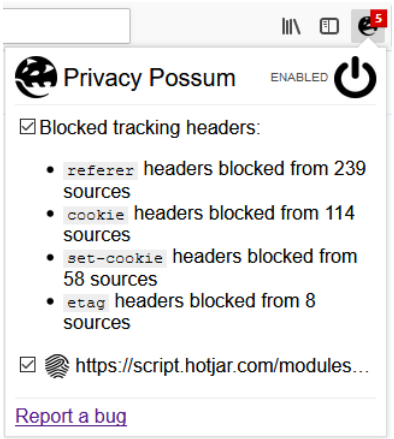


Figure 4.5: Privacy Possum interface

4.5 Firefox Content Blocking

Firefox has its built-in anti-tracking tool. In older versions of Firefox, this feature was called Tracking Protection. Since Firefox 63, Tracking Protection becomes part of a new set of features called Content Blocking [1]. It

[19] https://github.com/cowlicks/privacypossum

offers the same tracking protection but gives the user more options what they want to block.

Firefox Content Blocking is blacklist-based, and you can choose which blacklist Firefox uses to block trackers. There are two options you can choose from, namely basic protection and strict protection. Basic protection allows some trackers so websites function properly while strict protection blocks all known trackers, which may cause in websites function not correctly. What is remarkable is that Disconnect [14] supplies both lists. Therefore we would expect the results of Disconnect to match the results of Firefox Content Blocking.

Firefox Content Blocking contains three modes: Standard, Strict, and Custom. Standard blocks third-party tracking cookies in every window, but known trackers only in Private Windows. Strict blocks all trackers Firefox detects. The custom mode gives you the possibility to decide what Firefox should block. Firefox Content Blocking gives you also the possibility to send a "Do Not Track" signal to websites. This tells websites that you do not want them to track your browsing behavior.

When Firefox is blocking a tracker, it shows a shield icon. The user can click on this icon, after which Firefox shows what is blocked. Figure 4.6 shows an example of Firefox Content Blocking interface.

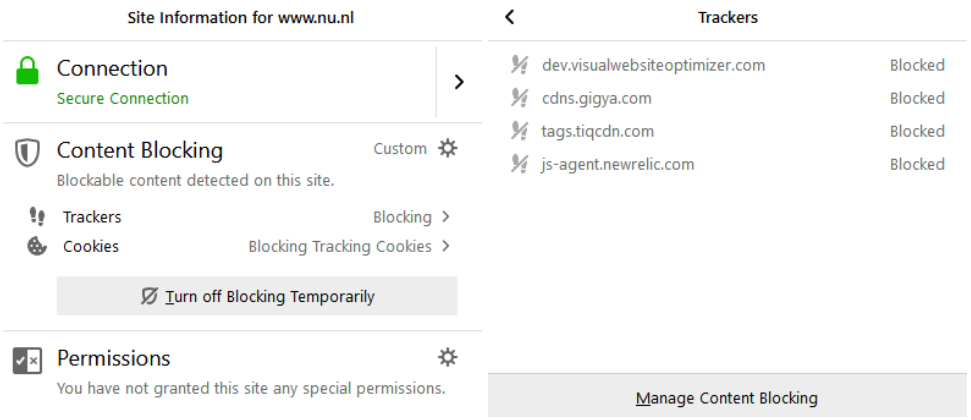


Figure 4.6: Firefox Content Blocking interface

Chapter 5

Theoretical comparison of privacy plugins

Now we have discussed different privacy plugins; it is time to compare them. It is important to keep in mind that most information from Section 4.4 comes from the plugins itself, so we have no evidence if their claims are valid. Boumans also provided a theoretical comparison between privacy plugins in his thesis [6] and came to some same advantages and disadvantages of blacklists and algorithms. However, this comparison was made independently of Boumans information. An overview of properties for each plugin is given in Table 5.1.

	Ghostery	Privacy Badger	Disconnect	uBlock Origin	Privacy Possum	Firefox Content Blocking
Uses a blacklist	Yes	No, but uses a Yellowlist	Yes	Yes	No	Yes
Uses an algorithm	Yes	Yes	No	No	Yes	No
Open source	Yes	Yes	Yes	Yes	Yes	Yes
Collects user data	No	No	No	No	No	No
Recommended by BOF	Yes	Yes	Yes	Yes	No	N/A
Block level	URLs	Domains	Domains	URLs	Domains	Domains
Business model	Sells plus version / funded by Cliqz	Non-profit	Sells premium version	Non-profit	Non-profit	Non-profit
Options offered	Many	Many	Some	Many	Few	Some
Firefox users (June 10, 2019)	1.160.504	535.427	176.400	4.664.435	95.895	N/A

Table 5.1: Overview of plugin properties

5.1 Blacklist vs. Algorithm

In Section 4.2, it becomes clear that blacklist-based and algorithm-based are the two main anti-tracking techniques used by privacy plugins. Blacklist-based plugins use a list of third-party trackers. If a domain or URL occurs on this list, it will be blocked. Algorithm-based plugins determine on behavior whether a domain/URL is a tracker.

Both have their advantages and disadvantages [27]. A drawback of the use of a blacklist is that it must always keep up-to-date, while an algorithm can recognize a new tracker by its behavior. A blacklist can also contain false positives since it operates most of the time on the domain level. This means that it blocks possibly more than necessary. The classification of a

tracker by a blacklist is often not accurate enough. Another side effect is that blocking false positives rather leads to website breaks, which does not benefit the functionality of a website.

An advantage of blacklist-based over algorithm-based blocking is that a blacklist is more consistent in contrast to algorithm-based. When the behavior of a tracker is modified to bypass the security, it is possible that an algorithm-based plugin no longer sees the tracker as a threat, while a blacklist-based plugin will block the tracker as long as it occurs on the blacklist. However, if a tracking party decides to change their URLs or domains, the blacklist will possibly miss these trackers while an algorithm-based blocker would possibly detect these trackers. So it is not quite an advantage.

Besides, it is also the question of how to determine whether a tracker should include in a blacklist. Trackers can now have a mixed tracking behavior which makes it even more difficult classify the tracker [27]. This means that at one time a tracker behaves like a tracker and the other time it does not. An algorithm-based is possible in the advantage because it checks the behavior of a tracker each time.

Algorithm-based plugins also have a disadvantage. In the past, it turns out that the approach of Privacy Badger leads in practice to a significant amount of website breakage [12, 14, 22].

Taking every advantage and disadvantage into account, an algorithm-based plugin seems to perform better in theory, because it does not depend on a blacklist that needs to be kept up-to-date. In addition, it probably leads to fewer false positives since it determines on the behavior of a tracker, whether it should be blocked or not.

However, a combination of blacklists and an algorithm that complement each other would seem an ideal setup. This combines the best of both worlds.

5.2 Ghostery, Disconnect and uBlock Origin

It is obvious to compare Ghostery, Disconnect, and uBlock since all use a blacklist. The easiest way to do this is to compare the blacklists. Unfortunately, Ghostery did not make its blacklist public. Ghostery claims it has a library of over 4500 trackers, while Disconnect has a list of 3194 trackers currently. If we have to believe Ghostery, then it offers protection against more trackers than Disconnect. It is possible to reconstruct the Ghostery blacklist. However, this costs much work and could, therefore, better be covered by future work.

We must treat uBlock Origin differently. This plugin uses (besides their own blacklists) blacklists supplied by third parties. It is therefore not fair to compare this with Ghostery and Disconnect. The EasyPrivacy list is such a list and currently contains 17,177 trackers. So uBlock Origin has larger

blacklists than the other two plugins, but are not created by uBlock Origin. Moreover, these lists can also be used by several other privacy plugins.

Ghostery and Disconnect have both invented techniques to prevent site breaks. Ghostery has a feature called 'Smart Blocking' which automatically block trackers that slow down webpages and unblock trackers if they are breaking the functionality of the website. According to Section 4.4.3, Disconnect does not block parties that deliver content by default. uBlock Origin takes a different approach. It offers an option that disables cosmetic filtering.

Bits of Freedom recommends all three plugins. Ghostery has in the past been negative in the news due to their handling of user data. Now, Ghostery offering a paid version of the plugin and is funded by Cliqz. Disconnect is financed by offering paid versions of the plugin. It says that it does not collect user data or sell it to third parties. uBlock Origin has no business model.

Which plugin performs better in theory? That is difficult to say without performance results. Ghostery, Disconnect, and uBlock Origin all provide a well-organized interface. Ghostery and uBlock Origin offer more options to the user. The plugins also use methods to prevent site breaks, are open source, and it seems that they respect the privacy of the user. If we then purely look at the largest tracker database then uBlock Origin is, in theory, the better plugin of the three. However, Ghostery implemented an algorithm to supplement their blacklist by anonymizing all personal data in a request from a tracker. So Ghostery is a combination of blacklist-based and algorithm-based, which also has its advantages. As a remark, Ghostery uses the blacklist as the primary blocking mechanism supplemented by an algorithm to block trackers that slip through. This can still result in false positives due to the blacklist.

5.3 Privacy Badger vs. Privacy Possum

Privacy Possum is developed because adding new privacy protections was difficult with the current architecture of Privacy Badger. An example of this is fingerprinting. When Privacy Badger detects fingerprinting, it blocks the domain instead of the URL. Everything from that domain is now blocked, which can lead to site breakage. Privacy Badger knows this and therefore adds the domain to the yellowlist. The problem is that this only prevents cookies from being sent to the domain, so the fingerprinting script is not blocked[1].

A big difference between these two plugins is the interface. While Privacy Badger offers the user many options, Privacy Possum offers almost no options. For the non-demanding user, this is probably not a problem, but

[1] `https://github.com/cowlicks/privacypossum` page visited on December 4, 2018.

it is in our eyes a loss. Privacy Badger shows which trackers are blocked, while Privacy Possum only shows how many sources they have blocked.

Bits of Freedom recommends privacy Badger, but Privacy Possum not. Privacy Badger is founded by the EFF, which has the purpose of protecting the digital rights of the people on the internet.

If we have to decide between these two plugins, we choose for Privacy Badger. Privacy Badger has more extensive interface compared to Privacy Possum. Besides, it is developed by the EFF that has a good reputation. Eventually, practical tests will have to show which plugin performs better in practice.

5.4 Changes in the last two years

As we mentioned earlier, we use Boumans [6] to find differences between plugins now and two years ago. We start with Ghostery who went from closed source to open source in March 2018. It also blocks now trackers by default. Two years ago, this needed a configuration step. Besides that, Ghostery has implemented several new features[2]. They use a feature called 'Smart Blocking' which automatically blocks trackers that slow down webpages and unblock trackers if they are breaking the functionality of the website. Furthermore, they have also features called Enhanced Ad Blocking and Enhanced Anti-Tracking. Enhanced Ad Blocking blocks ads while Enhanced Anti-Tracking keeps your browser anonymous to protect your privacy. It supplements Ghostery's blacklist by anonymizing all personal information in a request. A tracker may be allowed to execute, but your privacy is still protected.

Privacy Badger did not implement any significant changes in the last two years. Apart from the growth of the blacklists, Disconnect and uBlock Origin also have not made any major changes.

Firefox Tracking Protection turns into Firefox Content Blocking. Whereas it was only possible to use tracking protection in private browsing two years ago, it is now always possible. It also offers more options, such as standard, strict, and custom protection.

[2] https://www.ghostery.com/faqs/what-are-the-new-ghostery-8-features/

Chapter 6

Experimental comparison of privacy plugins

In Chapter 4 we made a comparison of privacy plugins based on theory. In this chapter, we make an experimental comparison of privacy plugins to validate our findings. The purpose of this research is to figure out which plugin blocks most trackers using default settings. We do this on a large scale of 25,000 websites. The privacy plugin that blocks most trackers without harming the functionality of the websites contributes the most to privacy. We also try to answer the question of whether a third party blocker is needed or if a good Firefox privacy configuration is sufficient enough. Therefore we compare the selected privacy plugins with two configurations of Firefox Content Blocking. These configurations, together with the plugins, are listed in Table 6.1.

First, we discuss the methodology in Section 6.1. After this, the details of our implementation are discussed in Section 6.2. After that, we explain the research setup in Section 6.3. The analysis and the results are discussed in Section 6.4.

1	Ghostery
2	Privacy Badger
3	Disconnect
4	uBlockOrigin
5	Firefox standard mode
6	Firefox strict mode
7	No tracking protection

Table 6.1: Privacy plugins and Firefox Content Blocking configurations

6.1 Methodology

We are doing experimental research since we want to figure out how privacy plugins and Firefox Content Blocking configurations of Table 6.1 perform in practice. We use 25,000 websites to experiment. The idea is to visit 25,000 websites and log the number of blocked trackers by a plugin to the browser console so that we can collect the data. The measurement is based on the number of trackers that a plugin blocks on a website. We assume a privacy plugin offers more privacy when it blocks more trackers. The research took place in April and May 2019. The entire data collection took approximately 1.5 months.

Every plugin is tested with default settings. The reason for this is that we want to simulate the experience of the end user as similar as possible. Therefore, we assume that most users of a privacy plugin do not make any changes to the settings.

Alexa Top 1 Million

We use the Alexa Top 1 Million in all of our measurements. Alexa[1] is an amazon.com company, which ranks sites. According to Alexa itself, the ranking is based on their global traffic panel (millions of people who use Alexa products, such as the Alexa toolbar[2]) and traffic data from direct sources. These direct sources run Alexa script and therefore can easily exchange traffic data directly with Alexa.

We downloaded the Alexa Top 1 Million once, on March 29, 2019[3]. By downloading the list once and not updating, we know for sure that every measurement uses the same websites. The list comes in CSV-format, which means comma-separated values and is easy to implement. In this study, we do not use all 1 million sites. Therefore we truncate the list to the desired format. In our case, we select the top 15,000 sites and a random selection of 10,000 sites from the Alexa Top 1 Million minus the top 15,000 sites. Ten thousand websites were randomly chosen to investigate whether there is a difference between the number of trackers on the most popular and less popular websites.

Remarks

Privacy Badger and Ghostery define blocked domains as the trackers, while Disconnect and uBlock Origin define blocked requests as the trackers. Unfortunately, we were unable to change the code of Disconnect and uBlock Origin so that we could log the number of blocked domains to the browser console without breaking the plugin. The result of this is that we cannot

[1] https://www.alexa.com/

[2] `https://www.alexa.com/toolbar`

[3] `http://s3.amazonaws.com/alexa-static/top-1m.csv.zip`

directly compare these two groups of plugins. As a solution, we investigate how many requests a third party domain sends on average. We do this by running a proxy when visiting the 25,000 websites with no tracking protection and collect all the requests. Then we look at how many third-party domains we can extract from this.

We assume all domains other than the host domain are third-party domains, but not necessarily third-party trackers. Every unique domain is managed by an administrative authority. However, the same administrative authority can manage multiple domains. This is not uncommon, take, for example, Facebook. Facebook manage multiple domains like: `facebook.com`, `fbcdn.com`, `fbsbx.com`, and many more. We only exclude `facebook.com` from our third-party domain list when visiting `facebook.com`. Although it is quite possible that other domains of Facebook provide additional non-tracking services to the user (e.g. static content).

We could reduce our list of gathered third-party domains by using Mozilla's Public Suffix List[4] and blacklists like EasyList. However, in this study, we do not determine for ourselves whether a domain is a tracker but assume that a domain is a tracker if a privacy plugin blocks it.

When visiting the websites, we have almost no interaction. In fact, we only visit the homepage. A real user might log in, scroll down or click on some links during their visit. This is something to keep in mind. We stay on the homepage until the 10 seconds timeout is reached. 10 seconds is more than enough time for a website to fully load. Also, the time interval gives enough time for plugins to collect necessary tracker data. Some websites load infinitely long and will, therefore, be interrupted after 10 seconds to ensure a good flow of the program.

In the next paragraph, we discuss the instruments we use and explain why we use them.

Instruments

We mentioned earlier that we do not determine if a domain is a tracker or not. Instead, if a privacy plugin blocks a domain, we assume the domain is a tracker. Luckily almost every privacy plugin we investigated, keeps rid of the number of trackers they block. Usually, they display the number of blocked trackers on their plugin icon at the browser menu. So we need a way to extract this number from the privacy plugins. The most obvious solution would be logging the number of blocked trackers to the browser console. To establish this, we modify the source code of the plugins. How we are doing this is explained in Section 6.2.1.

Visiting 25,000 websites can be done in multiple ways. Of course, we do not want to do it manually, so we rely on a script. A script makes this experiment scalable since we can easily adjust the number of websites. Using

[4] `https://publicsuffix.org/`

many websites promotes the reliability of the results because false positives can be compensated. There are roughly three methods we can choose. The first method is by injecting JavaScript code directly in the Firefox browser console. This opens a URL in a new window. After x seconds it closes the window and opens the following URL in a new window. It is by far the easiest method since it is easy to implement and requires less programming. However, this method has also some drawbacks. Due to safety reasons, it is not possible to read files on your local machine from the browser console. Therefore, you have to hard code a list of 25,000 URLs. This is not a convenient way. Besides that, if the browser for some reason shuts down, we lose all gathered browser logs. Firefox also is not very stable when we make use of time intervals. An example of how the code would look like can be found in Appendix B.4.

A more convenient method is to make use of Firefox in combination with Selenium[5]. Selenium is a browser automation tool. It offers APIs for a selection of popular programming languages including Java and Python. The idea is basically the same a the previous method, but now we let Selenium open and close windows. Advantages are that Selenium is more stable than executing JavaScript directly from the browser console. It also lets you program in a language you prefer. Besides that, we can read local files like we normally do by using a specific programming language. But most important, Selenium can run Firefox in headless mode. This is more efficient compared to Firefox with a head. Therefore, this would be a better method concerning the first method.

A tool we mentioned earlier (Chapter 3) is OpenWPM. This is a web privacy measurement framework and is used in many privacy studies to collect data[6]. It is built on top of Firefox in combination with Selenium. OpenWPM contains many instruments for data collection. However, the only data we want to collect is the data produced by plugins. Unfortunately, OpenWPM offers no instrument for this.

Since we do not need all the extra options offered by OpenWPM, we choose the second method. This means we build our program based on Selenium. Everything we need can be implemented in a couple of lines of Python code. Some details about the script are explained in Section 6.2.2.

6.2 Implementation

There are several steps that we must implement before we can experiment. We have to **modify the plugins**, write a **script that visits websites** and write a **geckodriver log parser**. In the following sections, we discuss the implementation of the various components of this experiment.

[5] `https://www.seleniumhq.org/`

[6] `https://webtransparency.cs.princeton.edu/webcensus/`

6.2.1 Modifying plugins

First, we describe how we modify the plugins. The plugins can be downloaded from the official Firefox Add-ons page[7]. After we download a plugin, it is stored in the Firefox profile directory on your computer. The storage location can easily be found by typing in the following string in the Firefox search bar: **about:support**. The extension is saved as an XPI file, which is a ZIP file used by Mozilla applications. It contains an install script or manifest at the root of the file[8]. We can extract this file by simply renaming XPI to ZIP. Firefox plugins have their own structure[9]. To keep it brief, we only mention the files we had to modify.

Every Firefox plugin must contain a `manifest.json` file. It contains important information about the name, version, permissions, and files. When Firefox installs a plugin, it expects some 'id' belonging to the plugin. The expected 'id' can be found in the browser console as an error when trying to install a modified plugin. This 'id' must match the 'id' in the `manifest.json`. Therefore, we have to change the 'id' in `manifest.json`, to the 'id' Firefox expects. If we do not, Firefox will regard the plugin as corrupt, and it will not be usable. Below you can find an example of what `manifest.json` should look like for Disconnect:

```
"browser_specific_settings": {
  "gecko": {
    "id": "Disconnect@5.18.27",
    "strict_min_version": "42.0"
  }
},
```

Background scripts are another type of files we modify to be able to log data from the plugin to the browser console. These scripts are loaded after the plugin is loaded and stay loaded. The plugins we investigate contain mostly one background script which is often called `background.js`. This file should provide a function where the number of blocked trackers are count. If we found the right variable, we log it to the console by adding the following JavaScript code:

```
console.log("BBBB: [" + variable + "]");
```

We use `BBBB` as an identifier so we can easily parse the log file and only collect the data we need. Sometimes it is hard to find the right variable, which is representing the number of blocked trackers. Therefore, for every plugin we investigate in this research, we add the location where to add the above JavaScript code. Now, we compress all files to a ZIP file. Rename ZIP to XPI, and the plugin is ready to use.

[7] `https://addons.mozilla.org/en-US/firefox/`

[8] `https://developer.mozilla.org/en-US/docs/Archive/Mozilla/XPInstall`

[9] `https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Anatomy_of_a_WebExtension`

6.2.2 Python scripts

The idea of this experiment is to visit 25,000 websites while the plugins log the number of blocked trackers to the browser log. Therefore, we need a script that visits the websites and a script that parses the browser log. We start by discussing the script that visits the websites, our main script. The full code is documented and can be found in Appendix B.1. We only discuss some important decision in the next paragraph.

Main script

The main script starts with reading the CSV file. This file contains all the websites that will be used. Reading in a CSV file is a small step and therefore not special enough to discuss further here. A more interesting part is configuring Firefox. The first thing we do is creating a new Firefox profile.

```
profile = webdriver.FirefoxProfile()
```

Next, we add an plugin (extension) to the profile. This plugin will automatically be installed and enabled.

```
profile.add_extension(extension)
```

We also need to make some changes to the Firefox settings. This can be done by the `set_preference` method. The first thing we change is routing every console API call to stdout. The output will end up in the geckodriver.log file.

```
profile.set_preference("devtools.console.stdout.content", True)
```

Firefox tracking protection needs to be disabled to make sure that the privacy plugins block trackers instead of Firefox itself.

```
profile.set_preference("privacy.trackingprotection.pbmode.
    enabled", False)
```

We also have to disable private browsing mode. Otherwise, the plugin will not work.

```
profile.set_preference("network.cookie.cookieBehavior", 0)
```

`cookieBehavior` is set to `0`, this means that Firefox will not block any cookies. Now its time to create the webdriver with the desired preferences.

```
driver = webdriver.Firefox(profile)
```

Proxy

We have already said that we test not only plugins but also Firefox configurations. Unfortunately, it is not possible to use the same script for this. This is because the Firefox Content Blocking is built into Firefox. Since it is not possible to log the number of blocked trackers by Firefox Content Blocking, we need a different method to measure. Therefore, we use a proxy server

to collect requests from websites. BrowserMob proxy[10] is a proxy based on Selenium. This is ideal for us because we already use Selenium to automate the browser. With the BrowserMob proxy, it is possible to monitor traffic and therefore collects all requests that are made for loading a specific website. We also use this method for creating a baseline of third-party requests and domains by using Firefox with no tracking protection. The code is very similar to the **main script**. The difference lies in the way we collect the data. The script can be found in Appendix B.2.

Geckodriver log parser

When we run our main script, the gecko driver produces a log file. Listing 6.1 shows a part of the log file.

```
JavaScript error: chrome://browser/content/parent/ext-browser.js
    , line 655: TypeError: browser.ownerGlobal is null
console.log: "UUUU: [everis.com]"
console.log: "BBBB: [0]"
JavaScript warning: https://www.everis.com/s3fs-js/js/
    js_UxzRxogeeGHS7uzD6S-L4x1T1KcJmuHLcTjPSADLrxM.js, line 2:
    Using //@ to indicate sourceMappingURL pragmas is deprecated.
     Use //# instead
console.log: "BBBB: [1]"
console.log: "BBBB: [2]"
console.log: "UUUU: [phillipjeffries.com]"
console.log: "BBBB: [0]"
console.log: "BBBB: [1]"
console.log: "fonts loaded!"
console.log: "BBBB: [2]"
console.log: "UUUU: [faberspa.com]"
console.log: "BBBB: [0]"
console.log: "BBBB: [2]"
console.log: "BBBB: [2]"
console.log: "BBBB: [3]"
console.log: "BBBB: [4]"
```

Listing 6.1: Part of Gecko driver log file

Since the Gecko driver log contains more than just log information from the privacy plugin, it is necessary to filter out only the information that we need. We do this on the basis of identifier `'UUUU:'` and `'BBBB:'`. The line that contains `'UUUU:'` provides the current URL, while the line with `'BBBB:'` in it, provides the number of blocked trackers for the current URL. We store the maximum number of blocked trackers. As soon as the parser encounters a new line with `'UUUU:'`, the combination of the previous URL with the corresponding number of blocked trackers is saved, and the max blocked counter is reset to 0. The script can be found in Appendix B.3.

[10] `https://bmp.lightbody.net/`

6.3 Setup

We only need a few things for this experiment. First, we need a computer that can run for a long time. Preferably a Linux server. We used Ubuntu 18.04 LTS on an BladeVPS PureSSD X4 from Transip[11]. The VPS makes use of 2 Intel Xeon CPU's and has 4GB ram. This is enough to run the program serially, but not in parallel. Furthermore, we use the latest version of Firefox. As the time of writing, version 67.0 (64-bit) is the latest version. Besides, we use Selenium 3.14.0 with Python bindings, Gecko driver 0.24.0, and browsermob-proxy 2.1.4. Selenium requires geckodriver[12] to interface with Firefox. Every plugins runs with **default** settings.

Execution
The experiment is easy to implement. For the plugins, we use the script in Appendix B.1 The only thing we have to change is the path to the location of the edited plugin.

For the Firefox Content Blocking configurations and measuring the total number of third-party requests, we use the script in Appendix B.2. Here we only need to adjust the Firefox settings to the desired result.

The 25000 websites can be visited in one go for each plugin/Firefox configuration. We have to merge the top 15000 sites with 10000 random sites to one CSV file. In the end, we run the gecko driver log parser over the gecko driver log file to get the desired results.

Error handling
Different types of errors may occur while running the program. By adjusting the python program, we try to catch as many errors as possible. Unfortunately, there are some issues for which this is not possible. Therefore, we discuss several main problems and explain how to solve them.

One problem is that it is possible for the program to terminate randomly. In that case, we check how websites the program has visited and restart it from where it left off.

Another problem has to do with the system on which the program runs. If there is not enough memory available, the program may terminate. To prevent this, the program restarts automatically after every 500 websites. Depending on the system on which the program runs, the restart value can be adjusted down or up. In practice, it appeared that restarting after 500 websites have a positive effect on the stability of the program.

The biggest problem is that some sites ensure that the program ends up in an infinite loop. Of the 25,000 sites that we have used, two have caused

[11] `https://www.transip.nl/`
[12] `https://github.com/mozilla/geckodriver/releases`

this. The only solution is to exclude these websites and replace them with other websites.

6.4 Results

Before we go to the results, we want to emphasize again that the data from the privacy plugins and Firefox Content Blocking were not obtained in the same way. The privacy plugins provide the number of blocked requests/domains themselves. For Firefox Content Blocking, we use a proxy server. A direct comparison between these two groups is therefore not entirely fair.

Blocking

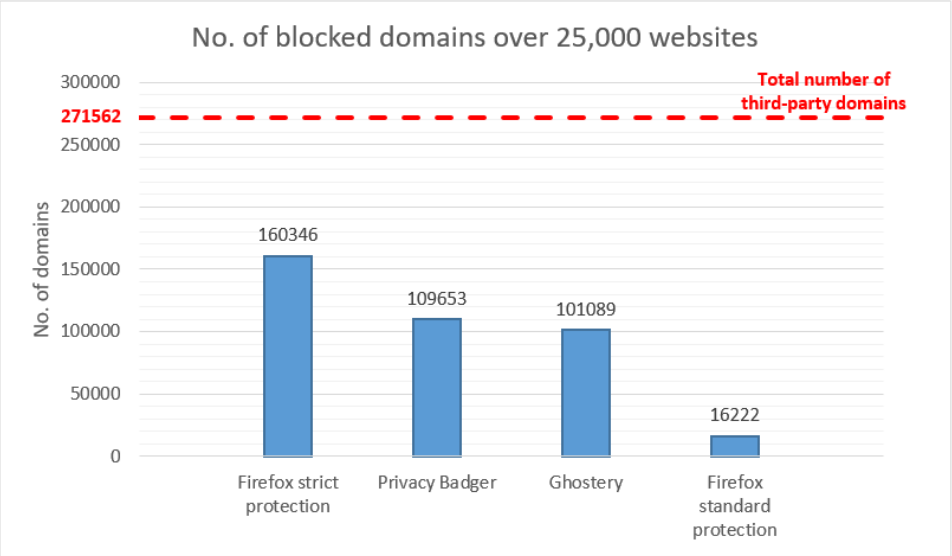


Figure 6.1: Number of blocked domains over 25,000 websites

We do not directly compare Privacy Badger and Ghostery with Disconnect and uBlock Origin, since the first group counts domains, while the other count's requests. Therefore we have separated them into two different graphs. Figure 6.1 shows the number of domains blocked over 25,000 websites. The red line represents the third-party domains collected with the proxy server (with no tracking protection).

Firefox strict protection performs best, followed by Privacy Badger and Ghostery. The difference between these last two plugins is minimal. Firefox standard protection blocks almost no tracking domains.

Remarkable is that many third-party domains have been found. About a half or less are considered as trackers by the plugins. On average, 35.7% of third-party domains are considered as trackers.

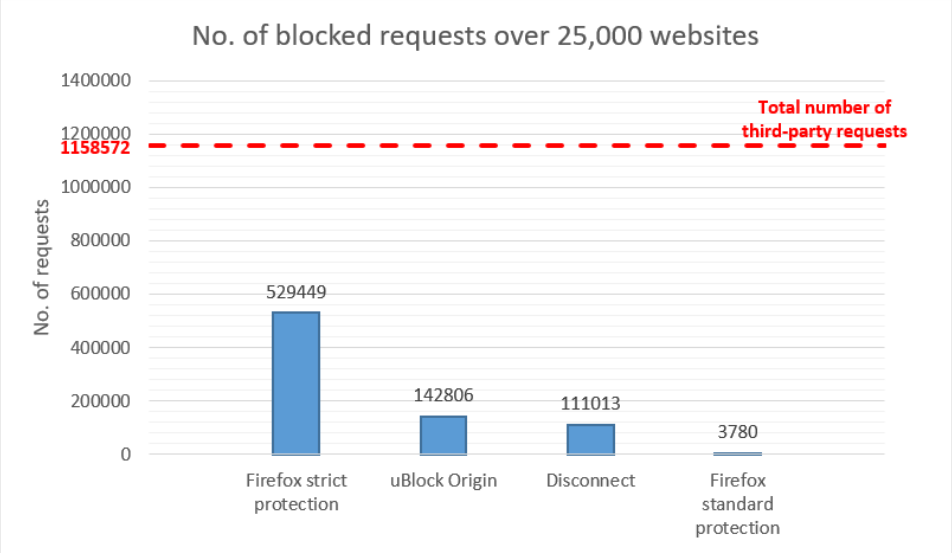


Figure 6.2: Number of blocked requests over 25,000 websites

Figure 6.2 shows the number of blocked request over 25,000 websites. The red line represents the third-party requests collected with the proxy server (with no tracking protection).

Also here, Firefox strict protection performs best. Followed by uBlock Origin and Disconnect. Firefox standard protection only blocks 0.33% of the third-party requests. On average, 17.0% of third-party domains are considered as trackers.

With the proxy server (without tracking protection) we have collected a total of 1,158,572 third-party requests. The requests can be reduced to 271,562 third-party domains. This would mean that a third-party domain has an average of 4.27 requests. If we apply this number on Disconnect and uBlock Origin, the results will look as in Figure 6.3.

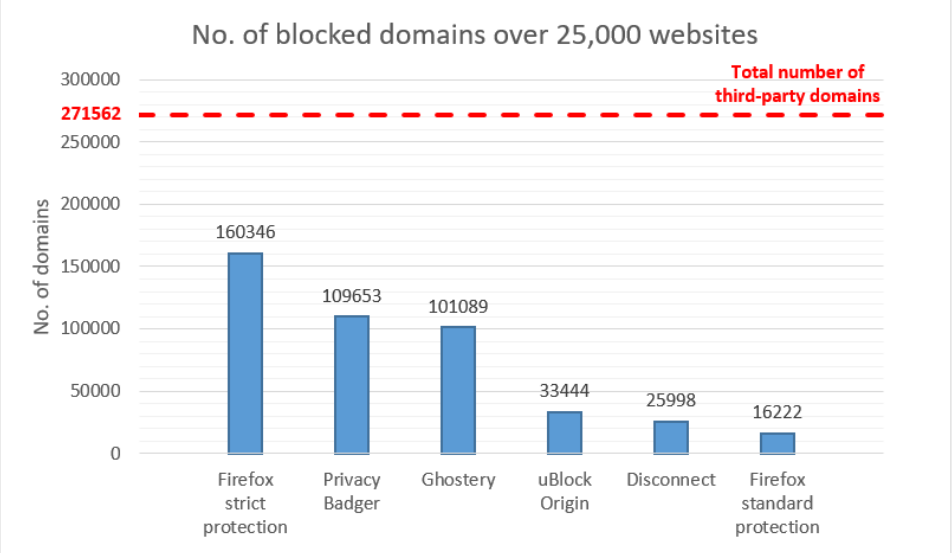


Figure 6.3: Number of blocked domains over 25,000 websites (modified)

However, Figure 6.3 seems to be incorrect. Disconnect and Firefox Content Blocking use the same blacklists. Therefore you would expect the results to be somewhat the same. It is therefore quite possible that the number of requests per third-party domain is lower. The administrative authority may be the cause.

An administrative authority manages the domain you visit. This authority can manage multiple domains that work together (deliver resources to each other). We assume the domain we currently visiting is the only first-party domain. The investigation revealed that a first-party make significantly more requests than third-parties for the current domain. However, if the first-party domain administrative authority has multiple domains, it is possible that these other domains make many requests in order to deliver content to the currently visiting website. Since we did not exclude these other domains, the average number of requests per third-party will be higher.

If we compare Figure 6.1 with Figure 6.2, we see that the results of Disconnect and uBlock Origin are close to the results in Figure 6.1. Therefore we combine all results in Figure 6.4. Keep in mind that this cannot be compared with each other since uBlock Origin and Disconnect count requests as a tracker. Therefore, they are marked in yellow.

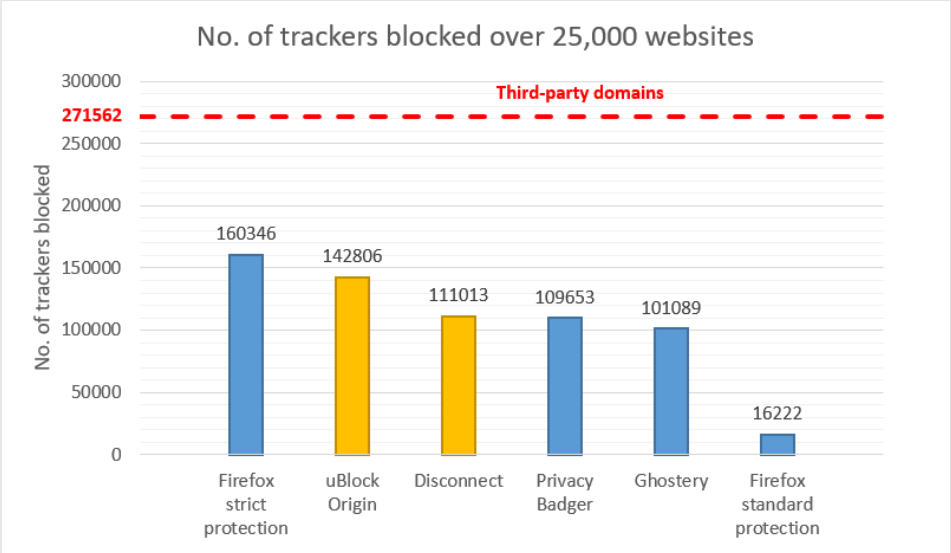


Figure 6.4: Number of blocked trackers over 25,000 websites

Detecting

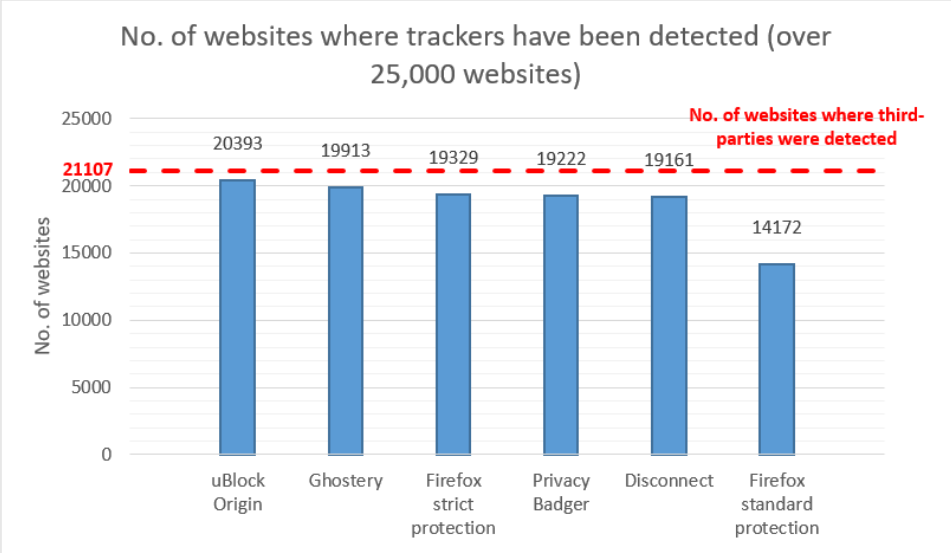


Figure 6.5: Percentage (%) of websites where trackers have been detected

Fortunately, it is possible to compare the number of websites where trackers are found. For this, it does not matter whether the plugin counts domains or requests.

From Figure 6.5, it follows that uBlock Origin detected trackers on most

websites. Furthermore, the result of all privacy plugins lies close to each other. Only Firefox standard protection has a big difference compared to the rest.

Alexa Top 15,000 vs. Alexa Random 10,000

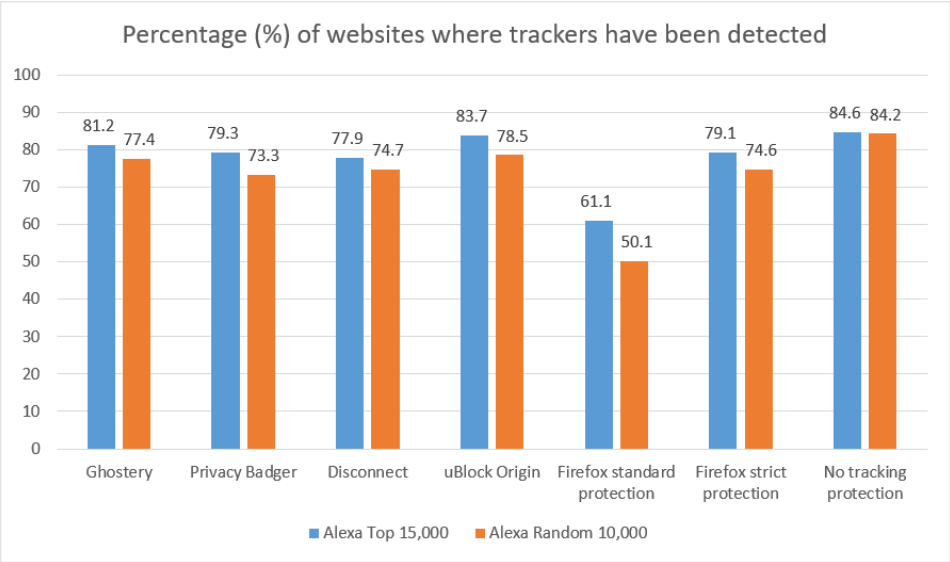


Figure 6.6: Percentage (%) of websites where trackers have been detected

Figure 6.6 shows us the percentage of websites where trackers have been detected. We mentioned earlier that we want to figure out if the Alexa top 15,000 differs from 10,000 random selected sites from the Alexa top 1 million when it comes down on tracking. From Figure 6.6, we can conclude that every plugin and Firefox configuration detects more websites containing trackers on the Alexa top 15,000 than the Alexa random 10,000.

The differences are not particularly significant, but large enough to conclude that the Alexa random 10,000 contains fewer websites with trackers.

Besides that, we also found more third-party domains on the Alexa top 15,000 relatives to Alexa random 10,000. On the Alexa top 15,000, we found a total of 188651 third-party domains. While, the Alexa random 10,000, contains a total of 82,911 third-party domains. This means that the Alexa top 15,000 contains **151.7%** more third-party domains relative to the Alexa random 10,000.

There is a chance that the 10,000 randomly selected websites contain fewer trackers by accident. However, we estimate this chance to be very low.

Chapter 7

Conclusions

In this thesis, we performed a practical and theoretical comparison of privacy plugins. We were able to draw several conclusions during the process.

There are many definitions of privacy. Within the scope of this research, we have formulated a definition of privacy. *Privacy: an action or communication M of a subject S that cannot be traced back to S by observer O.* We use this definition to examine privacy plugins. A plugin provides more privacy when it blocks more third-party trackers. There are many privacy plugins. Each type of plugin has a different purpose and guarantees a different form of privacy. In this study, we have focused on plugins that are specialized in protecting against third-party tracking.

The two main anti-tracking techniques used by privacy plugins are blacklists and algorithms. A blacklist-based plugin uses a list of trackers to determine what is a tracker and what not. It must always keep up-to-date to guarantee protection against most trackers. Algorithm-based plugin use heuristics to determine whether something is a tracker. It analyzes the behavior of a tracker. The most important advantage of an algorithm-based plugin over a blacklist-based plugin is that you are not dependent on a blacklist that has to be kept up-to-date. You are therefore better protected against new trackers. An algorithm also leads to fewer false positives, since each tracker is analyzed over and over again.

We provide a method to extract data directly from privacy plugins. By modifying the source-code of the plugin, we let the plugin log the desired data to the browser console. From here, the data can be collected. We have not seen this in any other study so far.

Every plugin visited 25,000 websites. Every website stayed open for 10 seconds. This was long enough to load a website and gave the plugin time to do its job.

The selection of websites contains the Alexa top 15,000 and 10,000 randomly chosen websites from the Alexa top 1 million (top 15,000 excluded). We found a total of 1,158,572 third-party requests. 271,562 third-party do-

mains can be extracted from this. The 10,000 random selected websites from the Alexa top 1 million contain relatively fewer trackers than the Alexa top 15,000. The chance that this is based on coincidence is small.

The number of trackers on a website depends on the time. The results show that sometimes even more trackers are blocked than the total number of trackers measured on a website. This is only possible if the number of trackers changes over time. Disconnect and Firefox Content Blocking use the same blacklists. Therefore you would expect the number of blocked requests to be somewhat the same. We compute an average of 4.27 requests per third-party domain.

Although we cannot compare uBlock Origin and Disconnect with Privacy Badger and Ghostery, we can conclude that in uBlock Origin blocks more trackers than Disconnect. Privacy Badger blocks more trackers than Ghostery. Besides that, Firefox strict protection is the overall winner when it comes to blocking trackers.

For some plugins, the information of Boumans [6] was outdated. Ghostery went from closed source to open source. It also implemented an algorithm that supplements their blacklist. Firefox has also changed in the meantime. It now offers more extensive options for blocking trackers.

Recommendation

One of the goals of this research is to make a recommendation to the end user, which plugin should be used. Results provide a basis for a ranking of which plugin blocks most trackers. Firefox strict protection performs best when it comes to blocking trackers. It offers comprehensive protection without the need for a plugin. However, Firefox already indicates that strict protection may cause some sites to break. We did not test this. On the other hand, Firefox standard protection hardly does anything to increase privacy.

All plugins are open source, have a well-arranged interface, and are user-friendly. If we have to recommend a privacy plugin, then two plugins get our attention. First of all, uBlock Origin. This plugin has the biggest blacklist of all plugins we examined. The blacklists are frequently updated. The present findings confirm that uBlock Origin detects trackers on most of the websites we visited. Besides that, we can conclude that uBlock Origin at least blocks more trackers than Disconnect and Firefox standard protection.

The next plugin we recommend is Privacy Badger. Privacy Badger is the only plugin we examined that relies on an algorithm as primary anti-tracking technique. Algorithm-based plugins are more robust against new trackers than blacklist-based plugins. A combination of uBlock Origin supplemented with the Privacy Badger algorithm should work best against blocking trackers. This means that together, they can contribute most to privacy. Besides, both plugins have a good reputation and no business model.

This does not mean that Ghostery and Disconnect are bad. These two

plugins also offer privacy. However, they perform less on both theoretical and practical level, in comparison to uBlock Origin and Privacy Badger.

Are you looking for something else than a privacy plugin? The Tor Browser is a good alternative against tracking. Not only tracking but also against censorship and enhancing privacy.

Chapter 8

Future Work

While there already exist many studies related to comparing privacy plugins, there are still some interesting topics that can be done.

In this thesis, we did not investigate ad-blockers because they serve a different purpose. However, it can be beneficial to compare ad-blockers to other privacy plugins and check if they perform better. Also, it is possible to test less popular plugins instead of popular plugins. Alternatively, using scientific privacy tools instead of plugins.

In this study, we assumed that we do not determine whether a domain is a tracker. However, assume that a domain is a tracker if a privacy plugin blocks it. The third-party domains found using the proxy server are therefore not compared to different blacklists (e.g. EasyList). In a follow-up study, it would be nice to determine how many of the domains found are trackers.

Even better, run a proxy server next to a privacy plugin and compare the data from the proxy server with that of the plugin. In this way, it is possible to demonstrate how reliable the data of a privacy plugin is.

This immediately brings us to another potential study. Ghostery keeps its blacklist secret. Running a proxy server next to Ghostery could be a method to reconstruct the blacklist. For this, you will have to visit many websites, a very time-consuming process. By making use of the scripts in this research, this is already a lot easier to achieve.

Unfortunately, we were unable to change the code of Disconnect and uBlock Origin in such a way that we can get the number of blocked domains out. It was only possible to collect the number of blocked requests. In a new study, this could be looked at extensively.

Another topic is to investigate Privacy Possum extensively. Privacy Possum differs from the other third party blockers by not showing the number of blocked trackers, but the number of sources for which it blocks different headers. It would be interesting to investigate whether the number of blocked sources can be linked to the number of blocked trackers.

Bibliography

[1] Joni. AliceWyman, Michele Rodaro. Content blocking. `https://support.mozilla.org/en-US/kb/content-blocking` accessed on February 20, 2019, no date.

[2] Hoofnagle C. Altaweel I, Good N. Web privacy census. *Technology Science*. 2015121502. december 15, 2015. `https://techscience.org/a/2015121502`.

[3] M. Asaka, T. Onabura, T. Inoue, and S. Goto. Remote attack detection method in ida: Mlsi-based intrusion detection using discriminant analysis. In *Proceedings 2002 Symposium on Applications and the Internet (SAINT 2002)*, pages 64–73, Jan 2002.

[4] Privacy Badger. Frequently asked questions. Accessed on January 30, 2019. Available: https://www.eff.org/privacybadger/faq.

[5] Rebecca Balebako, Pedro Giovanni Leon, Richard Shay, Blase Ur, and Yang Wang. Measuring the effectiveness of privacy tools for limiting behavioral advertising. 2012.

[6] W. Boumans. Web Tracking And Current Countermeasures. Bachelor's thesis, Radboud University Nijmegen, 2017.

[7] Disconnect. Accessed on January 30, 2019. Available: `https://disconnect.me/`.

[8] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1388–1401, New York, NY, USA, 2016. ACM.

[9] Council of the European Union European Parliament. *Directive (EU) 2016/680 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data by competent authorities for the purposes of the prevention, investigation, detection or prosecution of criminal offences or the execution of criminal penalties, and on the free movement of such*

data, and repealing Council Framework Decision 2008/977/JHA". 4 May 2016. available at: `https://gdpr-info.eu/`, 4 May 2016.

[10] Mozilla Foundation. Firefox add-ons. Accessed on November 21, 2018. Available: `https://addons.mozilla.org/en-US/firefox/`.

[11] Ghostery. Frequently asked questions. Accessed on December 6, 2018. Available: `https://www.ghostery.com/faqs/`.

[12] Jonathan Mayer H. P. Jason Bau and J. C. Mitchell. A promising direction for web tracking countermeasures. In *Proceedings of Web 2.0 Security and Privacy (W2SP)*. IEEE Computer Society, 2013.

[13] Mikael Berglund Jacob Palme. Anonymity on the internet, december 2004. available at: `https://people.dsv.su.se/~jpalme/society/anonymity.pdf`.

[14] Georgios Kontaxis and Monica Chew. Tracking protection in firefox for privacy and performance. In *Proceedings of Web 2.0 Security and Privacy (W2SP)*. IEEE Computer Society, 2015.

[15] Balachander Krishnamurthy, Delfina Malandrino, and Craig E. Wills. Measuring privacy loss and the impact of privacy protection in web browsing. In *SOUPS*, 2007.

[16] Louise Matsakis. Ad-blocker ghostery just went open source—and has a new business model. *Wired*, 2018.

[17] Jonathan R. Mayer and John C. Mitchell. Third-party web tracking: Policy and technology. *2012 IEEE Symposium on Security and Privacy*, pages 413–427, 2012.

[18] Johan Mazel, Richard Garnier, and Kensuke Fukuda. A comparison of web privacy protection techniques. *CoRR*, abs/1712.06850, 2017.

[19] Georg Merzdovnik, Markus Huber, Damjan Buhov, Nick Nikiforakis, Sebastian Neuner, Martin Schmiedecker, and Edgar R. Weippl. Block me if you can: A large-scale study of tracker-blocking tools. *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 319–333, 2017.

[20] Council of Europe. *European Convention for the Protection of Human Rights and Fundamental Freedoms, as amended by Protocols Nos. 11 and 14*. ETS 5. available at: `https://www.echr.coe.int/Documents/Convention_ENG.pdf` [accessed 31 October 2018], 4 November 1950.

[21] Privacy Possum. Accessed on December 4, 2018. Available: `https://github.com/cowlicks/privacypossum`.

[22] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and defending against third-party tracking on the web. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 155–168, San Jose, CA, 2012. USENIX.

[23] Stefano Traverso, Martino Trevisan, Leonardo Giannantoni, Marco Mellia, and Hassan Metwalley. Benchmark and comparison of tracker-blockers: Should you trust them? *2017 Network Traffic Measurement and Analysis Conference (TMA)*, pages 1–9, 2017.

[24] uBlock Origin. Accessed on May 2, 2019. Available: `https://github.com/gorhill/uBlock`.

[25] European Union. *Charter of Fundamental Rights of the European Union.* 2012/C 326/02. available at: `https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:12012P/TXT&from=EN` [accessed 30 October 2018], 26 October 2012.

[26] Craig E. Wills and Doruk C. Uzunoglu. What ad blockers are (and are not) doing. *2016 Fourth IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, pages 72–77, 2016.

[27] Zhonghao Yu, Sam Macbeth, Konark Modi, and Josep M. Pujol. Tracking the trackers. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 121–132, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee.

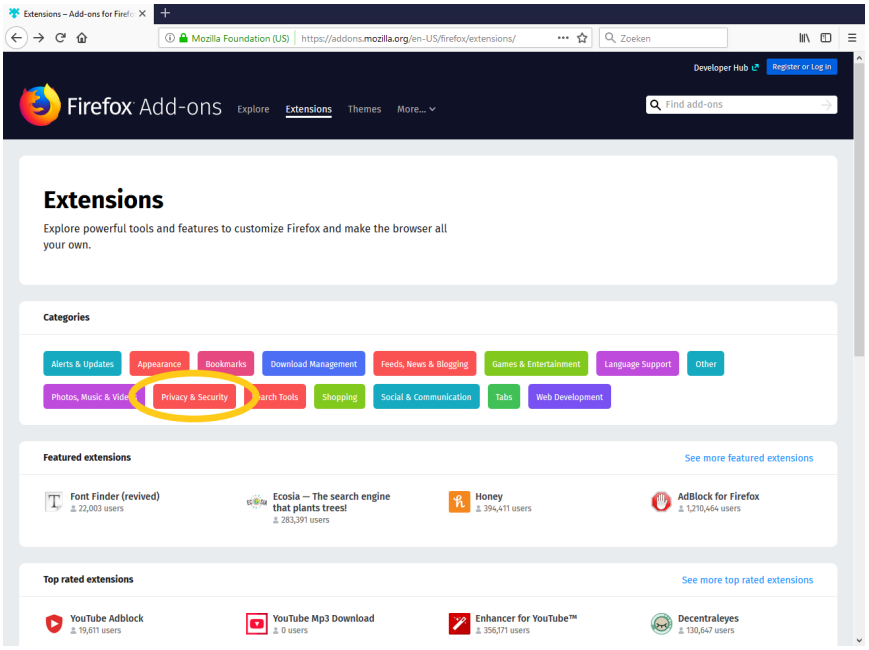
[28] T. Z. Zarsky. Thinking outside the box: Considering transparency, anonymity, and pseudonymity as overall solutions to the problems in information privacy in the internet society. pages 991–1044. University of Miami Law Review 58(4), 2004.

Appendix A

Privacy Plugins

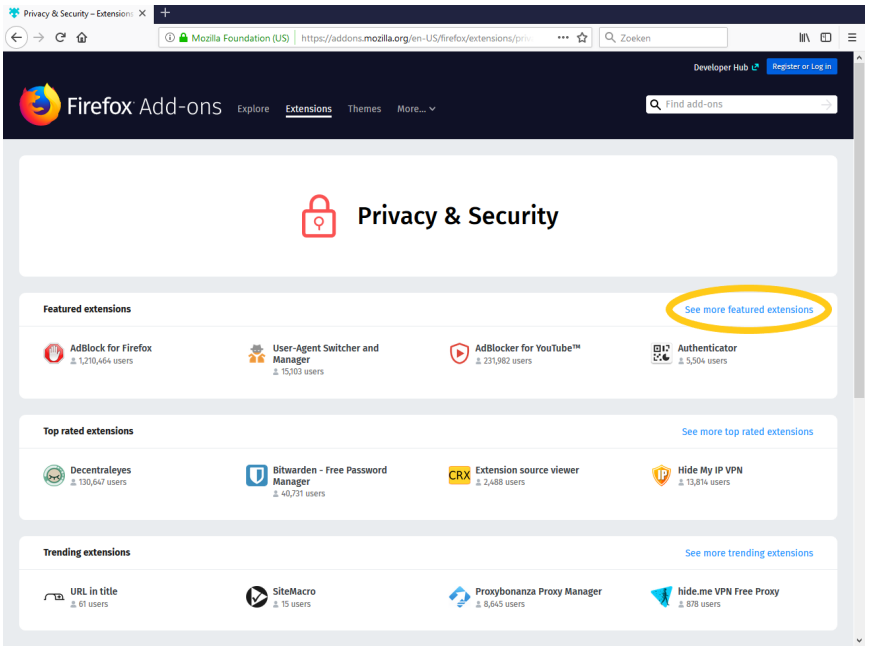
A.1 Firefox Add-ons Store

1. Go to the Firefox Add-ons store[1] and click on 'Privacy & Security'

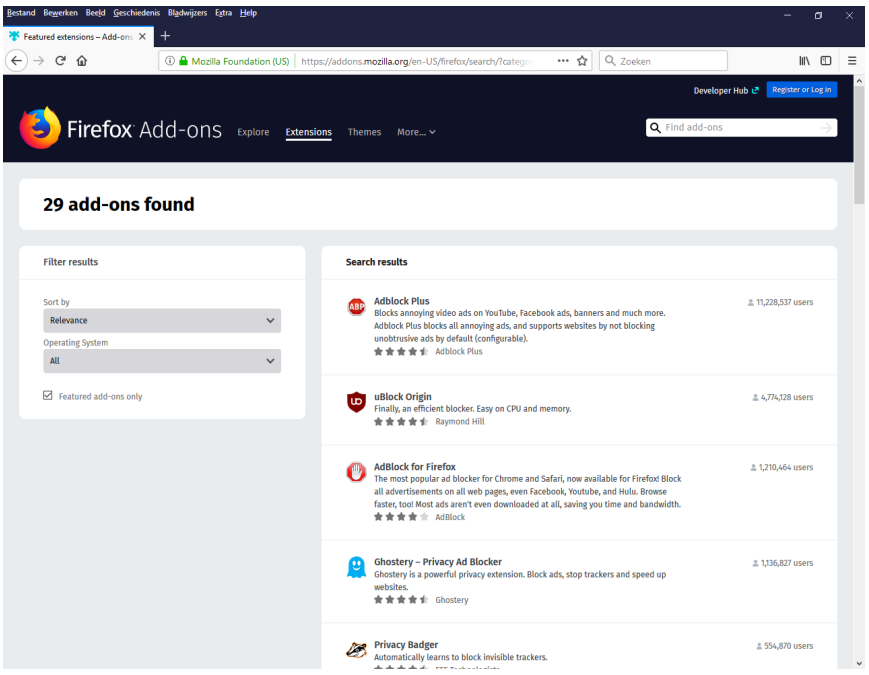


2. Click on 'See more featured extensions'

[1] https://addons.mozilla.org/en-US/firefox/extensions/

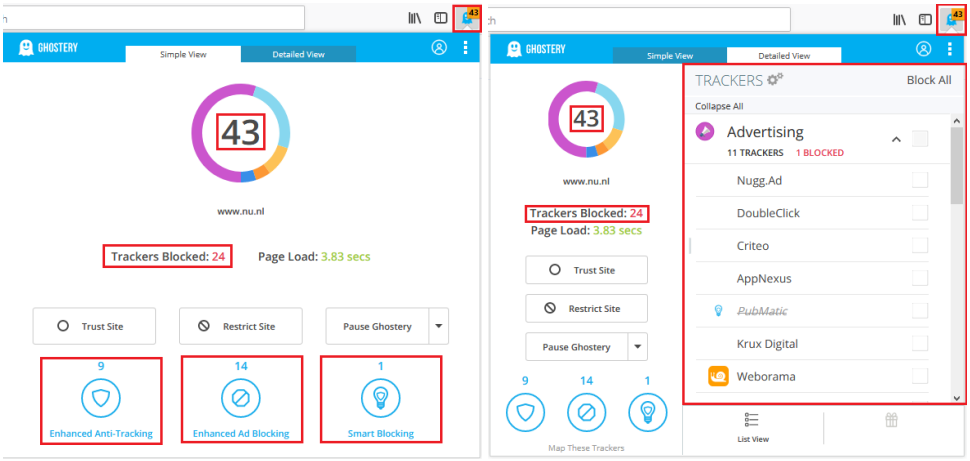


3. You will get a list of Privacy & Security plugins featured by Firefox

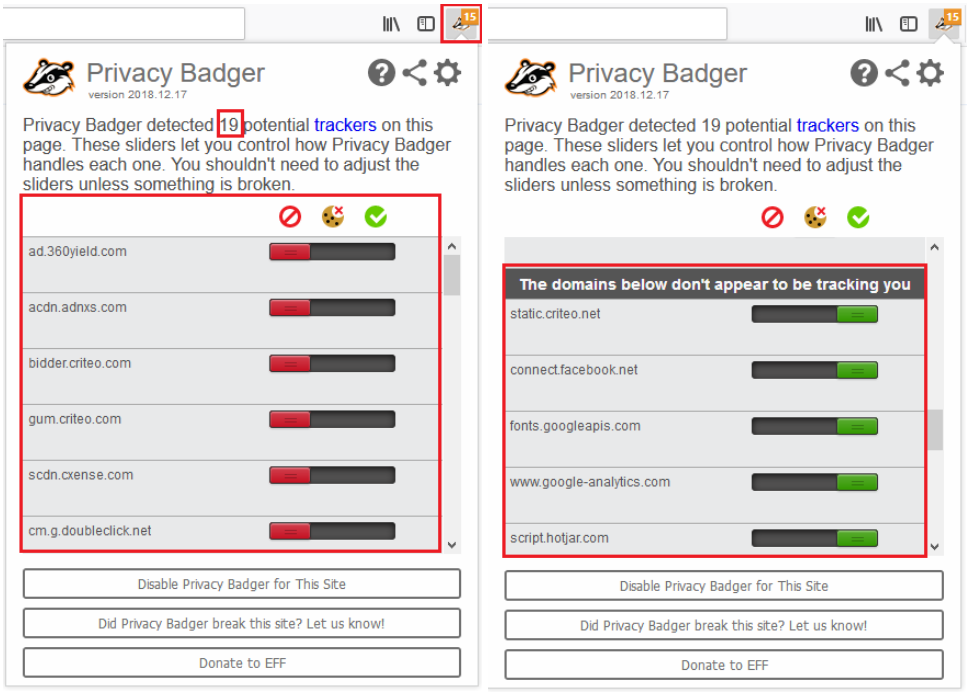


A.2 Interfaces

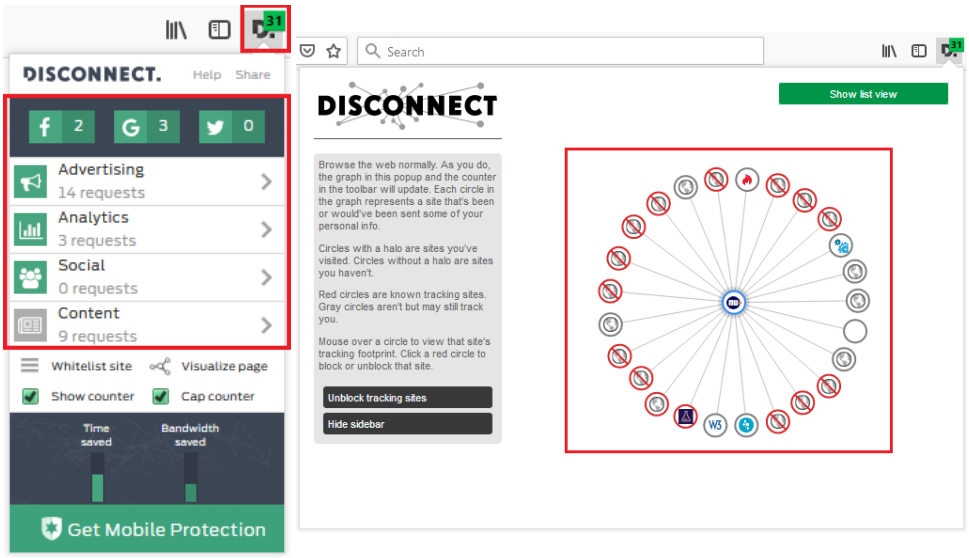
A.2.1 Ghostery



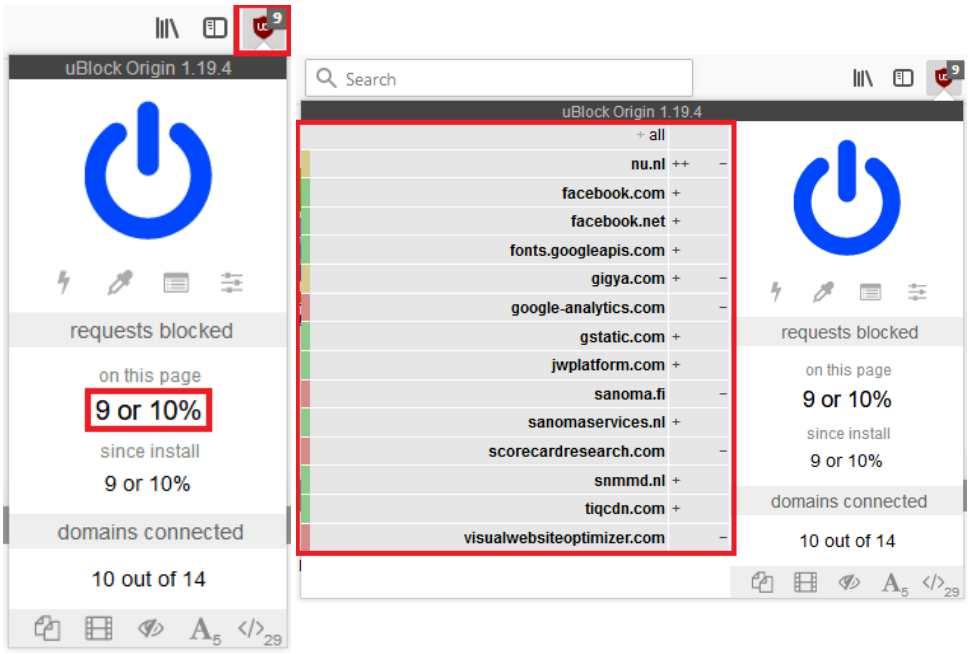
A.2.2 Privacy Badger



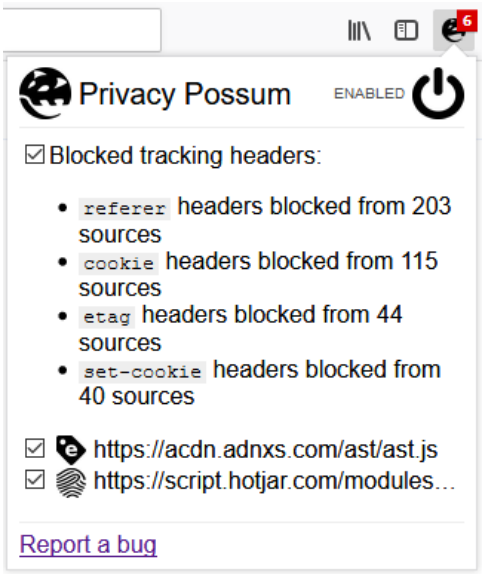
A.2.3 Disconnect



A.2.4 uBlock Origin



A.2.5 Privacy Possum



Appendix B

Scripts

B.1 Main python script

```
1  from selenium import webdriver
2  from selenium.common.exceptions import
       UnexpectedAlertPresentException
3  import csv
4  import time
5  import logging
6
7  file = 'top-1m.csv' # choose which csv file has to be loaded
8  extension = "/home/nick/Desktop/Ghostery edit/firefox@ghostery.
       com.xpi_FILES/firefox@ghostery.com.xpi" # path to extension
9  numberOfWebsites = 25000 # total number of websites to be
       scanned
10 restartNeeded = 500 # number of websites after restart is needed
11
12 def loadCsvFile():
13   # read URLs from csv file
14   with open(file) as csvfile:
15     readCSV = csv.reader(csvfile, delimiter=',')
16     websites = []
17     for row in readCSV:
18       website = row[1]
19       websites.append(website)
20   print("Csv file loaded")
21   return websites
22
23 def createFirefoxProfile():
24   # create new Firefox profile
25   profile = webdriver.FirefoxProfile()
26   # add plugin to the browser
27   profile.add_extension(extension)
28   # route every Console API call to stdout. The output will end-
         up in geckodriver.log
29   profile.set_preference("devtools.console.stdout.content", True
         )
30   # disable firefox private browsing
```

```
31    profile.set_preference("privacy.trackingprotection.pbmode.
          enabled", False)
32    # disable firefox cookieblocking
33    profile.set_preference("network.cookie.cookieBehavior", 0)
34    # prevent firefox from automatic downloading
35    profile.set_preference("browser.safebrowsing.downloads.enabled
          ", True)
36    # create driver instance
37    driver = webdriver.Firefox(profile)
38    return(driver)
39
40  # close all other tabs
41  def closeOtherWindows(driver):
42    while len(driver.window_handles) > 1:
43      driver.switch_to.window(driver.window_handles[-1])
44      driver.close()
45    # switch back to main tab
46    driver.switch_to.window(driver.window_handles[0])
47    time.sleep(1)
48
49
50
51  def executeProgram(driver, websites):
52    restartCounter = 1
53    for x in range (numberOfWebsites):
54      url = websites[x]
55      if restartCounter == (restartNeeded + 1): # check if restart
            is needed
56        driver.quit() # quit the current driver instance
57        time.sleep(10)
58        driver = createFirefoxProfile() # create a new driver
              instance
59        time.sleep(10)
60        closeOtherWindows(driver)
61        restartCounter = 1
62      restartCounter += 1
63      try:
64        current_url = "console.log('UUUU: [" + url + "]');" #
              create javascript string
65        driver.execute_script(current_url) # print current url to
              console
66        url_string = "window.open('http://" + url + "','_blank');"
               # create javascript string
67        driver.execute_script(url_string) # open url in a new
              window
68        time.sleep(10) # stay 10 seconds on the current page
69        closeOtherWindows(driver) # close every window except the
              main window
70      except UnexpectedAlertPresentException:
71        print("UnexpectedAlertPresentException")
72        closeOtherWindows(driver)
73    driver.quit()
74
75
```

```
76 websites = loadCsvFile()
77 driver = createFirefoxProfile()
78 time.sleep(10)
79 closeOtherWindows(driver)
80 executeProgram(driver, websites)
```

Listing B.1: Main script

B.2 Proxy script

```
1  from browsermobproxy import Server
2  from selenium import webdriver
3  from collections import OrderedDict
4  from selenium.common.exceptions import
       UnexpectedAlertPresentException
5  import json
6  import csv
7  import time
8  import collections
9
10 # Purpose of this script: collect all HTTP requests
11
12 file = 'top-1m.csv' # choose which csv file has to be loaded
13 numberOfWebsites = 25000 # total number of websites to be
       scanned
14 restartNeeded = 500 # number of websites after restart is needed
15 writingFile = open("alldata.txt","a")
16
17 def loadCsvFile():
18   # read URLs from csv file
19   with open(file) as csvfile:
20     readCSV = csv.reader(csvfile, delimiter=',')
21     websites = []
22     for row in readCSV:
23       website = row[1]
24       websites.append(website)
25   print("Csv file loaded")
26   return websites
27
28 def configureServer():
29   dict = {'port':8090} # port options
30   browsermobproxy_location = "/home/nick/Documents/browsermob-
         proxy-2.1.4-bin/browsermob-proxy-2.1.4/bin/browsermob-proxy
         " # path to browsermob-proxy
31   # Start browsermob proxy
32   server = Server(path=browsermobproxy_location,options=dict)
33   return(server)
34
35 def createFirefoxProfile(server):
36   server.start()
37   proxy = server.create_proxy()
38   # create new Firefox profile
```

```
39    profile = webdriver.FirefoxProfile()
40    # Setup proxy to point to our browsermob so that it can track
          requests
41    profile.set_proxy(proxy.selenium_proxy())
42    # enable firefox tracking protection
43    profile.set_preference("privacy.trackingprotection.enabled",
          True)
44    # enable firefox cookieblocking
45    profile.set_preference("network.cookie.cookieBehavior", 4)
46    # prevent firefox from automatic downloading
47    profile.set_preference("browser.safebrowsing.downloads.enabled
          ", True)
48    # firefox content blocking configuration
49    profile.set_preference("browser.contentblocking.category", "
          strict")
50    # create driver instance
51    driver = webdriver.Firefox(profile)
52    return(driver,proxy)
53
54  # close all other tabs
55  def closeOtherWindows(driver):
56    while len(driver.window_handles) > 1:
57      driver.switch_to.window(driver.window_handles[-1])
58      driver.close()
59    # switch back to main tab
60    driver.switch_to.window(driver.window_handles[0])
61    time.sleep(1)
62
63  def executeProgram(driver, websites, proxy):
64    restartCounter = 1
65    for x in range (numberOfWebsites):
66      url = websites[x]
67      proxy.new_har("req", options={'captureHeaders': True})
68      print(str(x) + " | " + url)
69      writingFile.write("CURRENT URL: " + url + "\n")
70      if restartCounter == (restartNeeded + 1): # check if restart
            is needed
71        driver.quit() # quit the current driver instance
72        time.sleep(10)
73        driver = createFirefoxProfile() # create a new driver
              instance
74        time.sleep(10)
75        closeOtherWindows(driver)
76        restartCounter = 1
77      restartCounter += 1
78      try:
79        url_string = "window.open('http://" + url + "','_blank');"
              # create javascript string
80        driver.execute_script(url_string) # open url in a new
              window
81        time.sleep(10) # stay 10 seconds on the current page
82        closeOtherWindows(driver) # close every window except the
              main window
83      except UnexpectedAlertPresentException:
```

```
84          print("UnexpectedAlertPresentException")
85          closeOtherWindows(driver)
86      # Print all URLs that were requested to a file
87      entries = proxy.har['log']["entries"]
88      for entry in entries:
89          if 'request' in entry.keys():
90              if entry['request']['headers']:
91                  y = entry['request']['headers'][0]['value']
92                  writingFile.write(str(y) + "\n")
93      # The string below is printed as divider between each
            measurements
94      writingFile.write("##########################\n")
95    driver.quit()
96
97
98  websites = loadCsvFile()
99  server = configureServer()
100 driver, proxy = createFirefoxProfile(server)
101 time.sleep(10)
102 closeOtherWindows(driver)
103 executeProgram(driver, websites, proxy)
```

Listing B.2: Proxy script

B.3 Gecko driver log parser

```
1  import time, os
2  import re
3  import csv
4
5  # This program keeps reading lines without terminating. It
       parses a gecko driver log file and converts it into a CSV
       file.
6
7  # Set the filename and open the file
8  filename = 'geckodriver.log'
9  file = open(filename,'r')
10
11 current_url = ""
12 maxBlocked = 0
13 counter = 1
14 firstRun = False
15
16 while True:
17      nextLine = file.readline()
18      if not nextLine:
19          time.sleep(1)
20          file.seek(file.tell())
21      else:
22          if nextLine.startswith('console.log: "UUUU: ['):
23              if firstRun:
24                  row = [counter, current_url, maxBlocked]
```

```
25                print(str(counter) + ' | ' + current_url + ' | ' +
                      str(maxBlocked))
26                with open('results.csv', 'a') as csvFile:
27                     writer = csv.writer(csvFile)
28                     writer.writerow(row)
29            firstRun = True
30            counter += 1
31            maxBlocked = 0
32            current_url = re.search('UUUU: \[(.*)\]', nextLine).
                  group(1)
33        elif nextLine.startswith('console.log: "BBBB: ['):
34            blocked = int(re.search('BBBB: \[(.*)\]', nextLine).
                  group(1))
35            maxBlocked = max(blocked, maxBlocked)
```

Listing B.3: Gecko driver log parser

B.4 JavaScript code

```
1  var websites = [   //Websites have to be added manually
2    'http://google.com',
3    'http://youtube.com',
4    'http://facebook.com',
5    'http://baidu.com',
6    'http://wikipedia.org',
7    'http://qq.com',
8    'http://yahoo.com',
9    'http://tmall.com',
10   'http://taobao.com',
11   'http://amazon.com',
12   'http://twitter.com',
13   'http://sohu.com',
14   'http://live.com',
15   ];
16
17     var index;
18   var myVar;
19   var timer;
20
21   function nextWebsite(){
22     window.open( websites[index] );
23     if( ++index >= websites.length ) {
24       clearInterval( myVar );
25       console.log( "All websites are loaded" );
26     }
27   }
28
29   function start() {
30     timer = setInterval( timeIsOver, 120000 ); //time interval
           here 120000 = 120 Sec
31     index = 0;
32     nextWebsite();
```

```
33      myVar = setInterval( nextWebsite, 3000 );   //time interval
            here 3000 = 3 Sec
34    }
35
36    function timeIsOver(){
37      console.log("Time is over");
38      clearInterval( timer );
39    }
```

Listing B.4: Example code if you want to inject JavaScript code directly into the browser console