CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Detecting reputation manipulation among browser extensions

Master's thesis in Computer science and engineering

LUKAS ANDERSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

Detecting reputation manipulation among browser extensions

LUKAS ANDERSSON

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

LUKAS ANDERSSON

Supervisor: Benjamin Eriksson, Computer Science and Engineering
Examiner: Andrei Sabelfeld, Computer Science and Engineering

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2023

LUKAS ANDERSSON
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

A common practice on platforms today is implementing a review system where the users can recommend products or warn other users of potential faults. With this, a rise of malicious actors abusing the system to favor their products follow. In this study, We examine user reviews with novel methodologies to detect reputation manipulation. We build a modular framework incorporating multi-step processes, including data crawling, indexing, filtering, processing, and classification.

The methods include Aggregated Time Window (ATW), Co-author Analysis (COA), Written ratio, and Spam detection. Each of the methods is designed to uncover different patterns of manipulation. For example, our novel method ATW targets fake accounts and coordinated review campaigns by linking reviews posted within close temporal proximity. On the other hand, the COA method identifies connections between users who frequently review the same extensions, detecting coordinated review campaigns and incentivized reviews.

Ultimately, the aim is to detect and cluster reputation manipulation, which could be used in various ways. One use case is exploring the results to find malicious extensions, which could be done by exploring the results and analyzing the top-scoring extensions. The cluster results could also be traversed to find similar extensions to already known malicious extensions, providing clusters of malicious extensions.

Keywords: Web Security, Browser Extensions, Fake Reviews

Acknowledgements

First, I would like to thank my supervisor, Benjamin Eriksson. His continuous support, guidance, and insightful critiques throughout this thesis have been invaluable. His deep knowledge in this field has inspired me and opened up new avenues of thought, enriching my research experience.

Moreover, I would like to thank the entire research group, Andrei Sabelfeld, Pablo Picazo-Sanchez, and Benjamin Eriksson. Their collective expertise and guidance have been essential in shaping this thesis. Also, working as a research assistant in their group during the thesis has greatly deepened my knowledge about malicious extensions and web security.

I would also like to thank Joakim Anderlind and Mårten Åsberg for valuable critique and insights from peer review and opposition.

Lukas Andersson, Gothenburg, 2023-06-20

Contents

List of Figures

List of Tables

1

Introduction

In today's digital era, our reliance on various software continues to grow, fueled by the widespread use of technological devices like smartphones. One particular aspect of software distribution involves web browser extensions, which are small web applications that reside within the browser. Browsers often feature their own app stores for distributing approved extensions. A prime example is the Chrome Web Store [1], which distributes extensions for the Google Chrome web browser. These extensions have attracted millions of users globally [2], contributing to Chrome's immense popularity.

However, the open nature of the Chrome Web Store's user review system has led to the emergence of reputation manipulation [3]. This issue impacts the platform's credibility and may cause users to install low-quality or malicious extensions [4]. Reputation manipulation occurs when individuals or groups artificially enhance or undermine an extension's reputation through fake reviews and ratings. This behavior compromises the platform's integrity and prevents users from making well-informed decisions about which extensions to install. The problem is exacerbated by security experts and online security blogs recommending that users read reviews to enhance their internet safety [5–7].

Researchers have been actively exploring methods to detect malicious extensions early, such as examining downloads, source code, and analyzing trends and values to group extensions into clusters [8–12]. Clustering extensions can reveal common properties that expose them as malicious. Nevertheless, there needs to be a more significant focus on reviews in this context. Faking reviews is not a new phenomenon but a known issue, with several websites and communities claiming to sell positive reviews online [13–15].

Assuming malicious actors frequently work with multiple malicious extensions, there should be many similarities if the same individual or group is behind them. Even if unintentional, everyone leaves some form of fingerprint behind.

In this study, we propose an approach to detect reputation manipulation in the Chrome Web Store by examining various factors, such as the temporal distribution of reviews and the relationship between reviewers, and answer the following questions:

RQ1: Does the Chrome Web Store contain reviews meant to manipulate reputation? what are efficient techniques to detect such reviews?

RQ2: Could methods that detect reputation manipulation be used to identify malicious extensions?

1.1 Outline

The thesis is divided into seven chapters, where the first chapter introduces the subject. After an introduction, necessary background knowledge is described to better understand the later chapters. The third chapter then is Methods, which introduces the reader to the theoretical methodology used in the research without going into too much detail about how the methods are implemented practically. The next chapter, chapter four, then describes the implementation details which were left out of the methods chapter on a more technical and practical level. After the implementation details, the results are presented in chapter five. Here various results are displayed with some insight into possible causes and reasons behind each result. In chapter six, related works are discussed, and the contribution this research has to the space. The thesis is then wrapped up with a conclusion in chapter seven.

1.2 Ethics

First off, all data accessed is publicly accessible and does not require any form of authentication to reach. We also made sure to blur out information that could be considered identifying information, such as usernames and profile pictures. We are currently in the process of reporting malicious extensions to the vendor, Google.

2

Background

This chapter describes the background knowledge required to fully understand the rest of the thesis. The subjects covered include browser extensions, the Chrome web store, models of reputation manipulation, and discrete optimization.

2.1 Browser extensions

Browser extensions represent user-friendly software applications intended to boost or personalize the browsing experience by introducing features, functionality or modifying the look of web pages. Developed using JavaScript, these extensions empower developers to craft various applications that interact smoothly with the browser's components and user interface.

Browser extensions primarily comprise two key components: content scripts and background scripts [16]. Each component fulfills a unique role, allowing the extension to interact with web pages or the browser in various ways.

2.1.1 Content Scripts

Content scripts execute within a web page's context and gain access to the page's DOM (Document Object Model). This ability enables content scripts to read and modify web page content, permitting extensions to carry out tasks such as adding new elements, modifying styles, or manipulating existing content. Moreover, content scripts can communicate with other parts of the extension, like background scripts, by utilizing messaging APIs [17].

Nevertheless, content scripts are sandboxed and granted limited access to the browser's extension APIs, which aids in maintaining the security and privacy of users. Consequently, content scripts cannot directly access sensitive data, like cookies, or execute privileged actions, such as modifying browser settings.

2.1.2 Background Scripts

Conversely, background scripts function in the browser's background and are granted access to the full suite of extension APIs. They are tasked with handling activities that necessitate privileged access or need to persist across multiple web pages, including managing browser settings, storing data, or interacting with other browser

components. Background scripts can also communicate with content scripts, enabling them to coordinate actions and exchange data [18].

2.1.3 Potential Malicious Acts

While extensions provide a host of benefits, they are also susceptible to misuse for malicious purposes. The same capabilities that allow extensions to enhance user experience can be manipulated to execute harmful actions. For example, extensions with access to a user's browsing history and search queries can be utilized for query stealing. Query stealing involves intercepting and redirecting search queries to alternate search engines or advertising platforms, often without the knowledge or consent of the user. The reason behind it is often intent to sell the query data to create financial gains for the malicious actor at the user's expense.

Other potential malicious activities include injecting unsolicited advertisements [19], monitoring user behavior [20], pilfering sensitive information [21], or even installing additional malware. To protect users from these threats, it is vital to rigorously review and supervise extensions available in the Chrome Web Store, ensuring their adherence to strict security and privacy guidelines.

2.2 Chrome Web Store

The Chrome Web Store is a repository accessible to Chrome web browser users, offering a diverse range of web applications, themes, and extensions. Though anyone can submit to this repository, Google's approval process aims to prevent malicious software distribution.

In the Chrome Web Store, extensions cater to various purposes, from productivity tools and security enhancements to social media integration and entertainment applications. This broad selection has significantly influenced the Chrome browser's popularity, attracting millions of users worldwide.

The Chrome Web Store incorporates a user review system, allowing users to share feedback on their installed extensions. This system is crucial for promoting high-quality extensions and assisting users in deciding which extensions to install. Users can rate extensions on a scale of 1 to 5 stars and provide written reviews that detail their experiences with the extension. These ratings and reviews, visible to the public as shown in Figure 2.1, can influence other users' decisions to either install or steer clear of specific extensions.

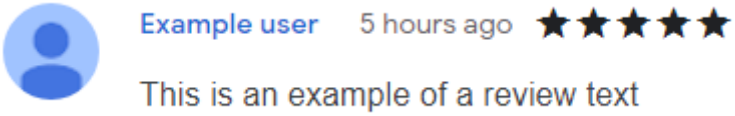

Figure 2.1: Example review as other users see it

However, the transparency of the review system also makes it vulnerable to manipulation by unscrupulous developers or other individuals who may attempt to boost the reputation of particular extensions through fake reviews artificially. This conduct, referred to as reputation manipulation, jeopardizes the credibility of the review system and may result in users installing substandard or even malicious extensions. Consequently, detecting and mitigating reputation manipulation in the Chrome Web Store is vital to preserve the platform's integrity and ensure a positive user experience.

2.3 Reputation Manipulation Techniques

In this section, we will discuss models of how malicious actors might behave when manipulating the reputation of extensions in the Chrome Web Store. By familiarizing ourselves with these prevalent reputation manipulation techniques, we can better understand the challenges and intricacies of detecting and addressing such behavior in the Chrome Web Store. This insight will also help guide the development of methods and algorithms employed in our study to identify and counter reputation manipulation effectively.

2.3.1 Fake Accounts

One prevalent technique used for reputation manipulation is creating and using fake accounts. Dishonest developers or other individuals may create multiple fake accounts to post positive reviews and high ratings for their extensions. These accounts may also be used to submit negative reviews and low ratings for competing extensions, intending to damage their reputation and reduce their perceived quality [22]. Detecting fake accounts is challenging, as malicious actors often employ strategies to make their accounts appear legitimate, such as using real names, authentic profile pictures, and even posting genuine-looking reviews for other extensions.

2.3.2 Spamming

Spamming involves posting a large volume of reviews or ratings within a short period to manipulate an extension's reputation quickly. This method can be used in conjunction with fake accounts, where each account posts multiple reviews or ratings in rapid succession. Alternatively, malicious actors may use automated tools or bots to submit a large number of reviews or ratings without the need for human intervention. Identifying spamming patterns requires analyzing the frequency, timing, and content of reviews and ratings to differentiate them from legitimate reviews.

2.3.3 Coordinated Review Campaigns

Coordinated review campaigns refer to the organized efforts of individuals who work together to manipulate an extension's reputation. These campaigns may involve submitting positive reviews and high ratings for a specific extension or negative reviews and low ratings for competing extensions. The coordination of these campaigns

often occurs through online forums, social media platforms, or other communication channels, making them challenging to detect without monitoring and analyzing information from external sources.

2.3.4 Incentivized Reviews

Incentivized reviews represent another form of reputation manipulation. Users are offered rewards, such as discounts or gift cards, for posting positive reviews and high ratings for a particular extension. This approach can result in biased feedback that fails to represent the extension's true quality accurately. In order to identify incentivized reviews, it is crucial to examine the content of reviews for patterns or phrases that indicate the reviewer received a reward in return for their review.

2.4 Discrete Optimization

Discrete optimization [23] is a branch of optimization techniques that deals with problems involving discrete or finite sets of variables. Discrete optimization aims to find the best possible solution from a set of feasible solutions, subject to certain constraints. This section will briefly overview some critical concepts in discrete optimization, which are relevant to our study on detecting reputation manipulation in the Chrome Web Store and used in later methods such as *Aggregated Time Window*.

2.4.1 Linear Programming (LP)

Linear Programming [24] is a mathematical optimization technique dealing with linear objective functions and constraints. LP problems can be solved using well-established algorithms like simplex or interior-point methods. However, many real-world problems involve discrete variables, which cannot be directly modeled using LP. In such cases, discrete optimization techniques, such as Integer Linear Programming, become necessary.

2.4.2 Integer Linear Programming (ILP)

Integer Linear Programming [25] is a mathematical optimization technique that deals with linear objective functions and constraints where some or all variables must be integers. ILPs are a subclass of discrete optimization problems and can be used to model many problems, including scheduling, routing, and resource allocation. ILPs are generally more challenging to solve than their continuous counterparts (LP) due to the added complexity of integer constraints. However, specialized algorithms and solvers, such as branch-and-bound and cutting-plane methods, can efficiently solve ILPs in many cases.

2.4.3 Linear Programming Relaxation

Linear Programming relaxation is a technique used to approximate the solution of an ILP by allowing the integer constraints on the variables to be relaxed, meaning

that the variables can take fractional values. Doing so transforms the problem into a linear programming problem, which is typically easier to solve. Once an optimal solution to the relaxed LP is found, various techniques, including rounding and branch-and-bound, can be applied to obtain an approximate integer solution for the original ILP. Although LP relaxation does not always guarantee an optimal integer solution, it can provide a useful starting point or a good approximation for many discrete optimization problems.

2.4.4 Bipartite Graphs

A bipartite graph is a particular class of graphs in which vertices can be divided into two separate, non-intersecting groups. This division ensures that an edge connects no pair of vertices within the same group. In simpler terms, all the edges in a bipartite graph exclusively link vertices from one group to those in the other group without any edges connecting vertices within an individual group. This unique characteristic of bipartite graphs makes them suitable for various applications in discrete optimization, such as solving matching problems, scheduling tasks, and analyzing network flow.

2.4.5 Total Unimodular Matrices

Total unimodular matrices (TUMs) are essential in discrete optimization, mainly due to their close relationship with bipartite graphs, which play a significant role in our study.

A matrix is considered totally unimodular if every square submatrix has a determinant of 0, 1, or -1. One fundamental property of TUMs is that the incidence matrix of a bipartite graph is guaranteed to be totally unimodular. Bipartite graphs can be represented as incidence matrices, which are binary matrices that indicate the presence of an edge between two vertices.

An essential benefit of working with TUMs is that the optimal solution of a linear programming problem will always be an integer solution when the constraint matrix is totally unimodular. In other words, if a matrix is TUM, the linear programming relaxation will yield the optimal integer solution without further rounding or employing more complex methods like branch-and-bound. This property enables us to solve the corresponding ILP problems more efficiently, as the optimal solution can be directly obtained from the linear programming relaxation.

By leveraging the connection between bipartite graphs and TUMs and utilizing the property that the LP optimal solution is always integer optimal when working with TUMs, we can design more efficient algorithms and solvers to detect reputation manipulation in the Chrome Web Store. This approach ensures that optimal solutions can be found in a timely manner, contributing to the effectiveness and robustness of our reputation manipulation detection methods.

3

Methods

This chapter introduces our novel methods used to analyze the extensions in the Chrome Web Store. Our primary goal is to detect reputation manipulation. We have developed a framework of modules to create multi-step processes that involve crawling the web store, indexing the data, filtering and processing the information, and finally, classifying and presenting the results. These processes are referred to as methods, and each method includes a chain of several modules, which lie under one of the mentioned categories. The modular approach is highly beneficial since we can reuse code, quickly expand the set of modules and rearrange the module chain in methods to configure methods. It is also easier to make frequent changes without depending on much code, for example, in the presentation modules, where the wanted output format can change frequently.

3.1 Crawler

The first step in analyzing the extensions contained in the web store is to acquire the data from the web store. Data is gathered using a crawler that browses the web store and saves all relevant information. The data is crawled by visiting each extension and saving the wanted data. The data saved on each extension includes the extension's name, ID, number of total ratings, and all written reviews. The information contained in reviews is the user's name, ID, the review text, the rating, the timestamp of the initial review, and the timestamp of the latest modification.

3.2 Index

The process is now divided into three steps to analyze the data modularly. First, the data needs indexing to allow efficient use, essentially taking the raw input from the crawler's snapshot and converting it into an efficient data structure. The specific structure depends on the requirements of the subsequent steps. For example, if processing modules benefit from efficient lookups, the data can be indexed into maps. The specific module using a map structure for lookups is referred to as *Efficient Maps*. The efficiently indexed data is then saved to a file, enabling loading later without re-indexing, preventing the need to run the indexer every time the program is executed.

3.3 Filtering and processing

Once the data is indexed into efficient data structures, it is ready for filtering and processing. This step takes the structured data as input and produces a refined version, either by filtering and narrowing it down to a subset or processing it into a new metric. An example of a module in this step is calculating the written ratio of reviews, which is how many out of the total reviews are written, as opposed to only a star rating. The written ratio can be calculated and added to the data by combining the known values of total reviews and written reviews. Although such calculations could be done in the first step, dividing the process decreases complexity and ensures that only necessary data is used in other modules to reduce space complexity. As a result, this step only gathers essential data for later modules or removes unnecessary data to reduce complexity.

3.4 Classification and presentation

After the data is processed, it is time for classification and presentation. We take the processed data from the previous steps and perform some classification algorithms in this step. Since this step is often quite complex, it is beneficial that most of the unwanted data are already removed to reduce complexity. This step produces some output, such as logs, graphs, or human-readable reports. Some examples of modules in this step are *Overview Stats* and *Review Length Graphs*, which produces a human-readable report and graphs, respectively.

3.5 Main methods

We will now discuss some of the main methods used in our research. The two most complex methods are *Aggregated Time Window* (ATW) and *Co-author Analysis* (COA), which are both described with examples to illustrate the general flow of each process. Some other examples of methods are *Written Ratio Analysis*, *Spam Detection Analysis*, and *Flag Merge*, which all are briefly mentioned as well. More detailed explanations and thresholds used can be found in the implementation chapter.

The ATW and COA methods offer complementary approaches to detecting reputation manipulation in extension reviews. ATW excels at identifying users with multiple accounts who post reviews in close temporal proximity, while COA focuses on discovering clusters of users who consistently review the same extensions. By combining these two methods, we can effectively uncover various forms of reputation manipulation.

3.5.1 Aggregated Time Window

The ATW method aims to link reviews posted within close temporal proximity to identify potential reputation manipulation. This method's primary targeted repu-

tation manipulation models are fake accounts and coordinated review campaigns. Since we are detecting temporal patterns, a user that creates accounts intending to write reviews will be detected if they switch between the accounts in close temporal proximity when submitting reviews. Groups of people writing reviews on specific extensions together, coordinated review campaigns, will also be detected since they would create clusters of reviews in close temporal proximity across the extensions they are reviewing. We can investigate the associated extensions by detecting such users to identify possible malicious activities. ATW is a novel approach to an unexplored problem, temporal connections.

First, the *Efficient Maps* indexing module organizes the data into nested hash tables, enabling fast and efficient lookups in both directions, from extension to the user and vice versa. The data is then divided into a series of time windows. We connect reviews within a specified threshold for each window, adding these connections to a graph for later use. Once all windows have been processed and connections added to the graph, a filtering process is applied based on review overlap ratios to reduce the data size and improve performance in subsequent, more complex steps.

Given the constructed graph, we face the issue of overlapping connections, as the algorithm connects every review to every other review in close temporal proximity. To address this, we apply discrete optimization, ensuring that each review is matched with at most one other review and always the optimal choice.

Implementation details, described in the implementation chapter, ensure that no review is ever connected to another review inside the same extension. Thus, we know that the graph we are constructing is bipartite. Given that the underlying graph is bipartite, and thereby also TUM, we can apply LP relaxation and solve the LP instead of directly solving it as an ILP and still compute the optimal integral value. LP relaxation lowers the complexity of the algorithm but keeps the results accurate.

The LP relaxed incidence matrix is constructed and solved as an LP, producing the actual number of shared reviews when solved. This information is stored in the data structure, replacing the previously calculated shared values from the time window iteration step.

With the accurate results, another filtering process is applied based on predefined thresholds, essentially just removing extensions in clusters with very few internal matches. Finally, the data is sorted and modified to produce readable and categorized results that can be manually evaluated or imported by the *Flag Merge* method to produce a report.

3.5.2 Co-author

The second method, Co-author, focuses on identifying connections between users who frequently review the same extensions, irrespective of the timing of the reviews. This approach helps uncover clusters of users manipulating the reputation of a common set of extensions. The primarily targeted models of reputation manipulation for this method are coordinated review campaigns and incentivized reviews. Coordi-

nated review campaigns are detected when users reuse accounts and review several extensions where the reviewed extensions are heavily overlapping. Incentivized reviews can be detected if users are incentivized to review a set of extensions before earning their reward. Requiring the user to review several extensions happens quite regularly, according to the results, especially in the crypto space, where users are rewarded with an airdrop[1] of some tokens for reviewing several wallet extensions. The main difference between the detection of COA and ATW is that COA only targets clusters of users reusing their accounts, but it can not detect users creating new accounts, which the ATW method addresses. Although this method is quite intuitive and addresses a problem that has been defined before, users collaborating as a group to review a target, it has not been done before to the extent we take it in this study.

Like the ATW method, COA loads the indexed data from the *Efficient Maps* module to enable quick lookups. However, this time the index is passed to a filtering module that filters out all users with only one written review and produces a new index. Filtering is done since a user naturally has to write several reviews to be detected by an algorithm like COA. Since we know that many users only write one comment, backed by the overview stats method shown in the results, we can reduce the complexity by removing them this early in the process.

Next, the algorithm iterates through each user and the extensions they have reviewed, calculating the overlap between the current user and other users who have also reviewed the same extensions. This process establishes the degree of overlap between users.

After calculating the overlap, it is time to recreate extension clusters based on the user clusters found, and this is done since the final result we seek in the method is a list of extensions and not a list of users.

Lastly, filtering is done before presenting the results in an output file for manual analysis and further investigation. This file is a lookup log and an input file for the *Flag Merge* method, like the ATW output file. It contains extension and user clusters since they can provide valuable information even though they are not the primary target.

3.5.3 Written ratio

Written ratio is a method that tries to leverage the fact that a user has a choice to make when submitting a review. They can leave a rating or attach text to it to make it a review. Since it takes extra effort to write some text, not all users who leave a rating decide to leave it along with that rating. We try to figure out how likely users are to write reviews instead of just leaving a rating, and with the statistics in hand, we can find outliers with very unlikely ratios.

[1]Airdrops refer to crypto projects sending tokens or other digital assets to reward their communities or generate excitement

3.5.4 Spam Detection

The following method, *Spam Detection*, looks at the deltas between reviews on a certain extension to mark reviews as potential spam. Every extension in the repository is looked at to gain statistical knowledge on how likely it is for this to occur naturally, producing outliers in the form of extensions with extraordinarily many extensions with low deltas.

3.5.5 Flag Merge

When we finally have some results from the previous methods, we use the method *Flag Merge* to merge the flags from the different methods. A list is produced that gives us a great overview of the top flagged extensions and provides a superset of the extensions flagged by any extension. This method also calculates some statistics on how much each method contributed to the superset of extensions and how much overlap was in each method pair's results.

4

Implementation

This chapter goes into technical details about the implementation of modules and methods. First, the three categories of modules are described, and then details are uncovered on two essential methods in this research; ATW and COA.

4.1 Index

As mentioned earlier, the indexing modules take the raw snapshot data and index it into an efficient data structure. The specific indexer most used in this project is the so-called *Efficient maps* module that creates two large maps, one for users and one for extensions, allowing lookups in both directions. Figure 4.1 shows the exact structure of the data. In the extensions map, the first layer includes entries where the key is the ID of the extension, and the value is another map containing reviews. The reviews map has entries where the key is the user ID, and the value is the actual review data in a list. The review data list had four values: the user ID, user name, review text, and the timestamp when the review was initially submitted to the Chrome Web Store.

On the other hand, if we look at the users' map, the first layer has entries with the user ID as keys and a map of extensions as values. The extensions map contains entries where the key is the extension ID, and the value is the review that said user has submitted to the chosen extension. The format of the review data at the deepest level is in the same structure as it is in the extensions map, as shown in Figure 4.1.

4.2 Processing and Filtering

Processing and filtering modules take the indexed snapshot and calculate new metrics or narrow down the data given to output data in the same format that was input into the function but with some new complementary info or as a subset of the input. An example of a module like this is the *Extract multi-users* module that takes the raw snapshot as input and then iterates through the entire users' list, filtering out any user with only one singular review (single-user). Filtering is beneficial if we know that we are not interested in single-users to make computations cheaper, for example, for the COA module, where co-authors naturally have multiple reviews (multi-users). At this stage, we can already remove any user that is not. The extensions map is also recalculated based on the new users' map before

```
users = {
  userID :{
    extID : [userID, userName, reviewText, reviewTimestamp],
    ...,
          },
  ...,
       }

extensions = {
  extID :{
    userID : [userID, userName, reviewText, reviewTimestamp],
    ...,
         },
  ...,
             }
```

Figure 4.1: Structure of *Efficient Maps*.

returning, ensuring that the users are removed from that map and extensions that strictly contain reviews from single-users are removed.

4.3 Classification and Presentation

In the classification and presentation modules, we complete the last processing steps needed to present a result from the process. Some calculations might have to be done depending on what is being presented as a result. Even though some processing might be present, the final goal is to achieve some output that can be manually inspected and evaluated. An example is a module that renders graphs to describe review length data. It is first supplied with information from a processing module that calculates all the crucial data to render such graphs. The module's output is then graphs which can be directly used as results. This category has many modules since it is the final step, which produces all the human-readable results. All the results presented in the results section are created from this category of modules. Some more examples are the final steps of ATW, COA, *Spam Detection*, *Written Ratio*, and *Flag Merge*.

4.4 Aggregated Time Window

The Aggregated Time Window module is designed to identify temporally correlated review patterns among extensions by analyzing review timestamps. The first module used in the method is the *Efficient Maps*, which supplies indexed data to the next module, called *Static Time Window* (STW). STW aims to divide all reviews into a timeline facilitating iterating in later steps. The ATW method could be used on any given time threshold supplied by the user before running. Customizability exists to

enable the user to analyze different time thresholds smoothly.

For example, if a user enters a value of 1 hour, we want to match every review with any other review within a 30-minute window, either before or after. The ATW method now divides the wanted threshold into a tenth of the user-supplied value before calling STW. The reason for dividing the time window is to increase the accuracy and decrease the margin of error. If the window is kept at one hour, there could be edge cases where reviews are located at the end or beginning of their time windows, resulting in reviews not being matched despite being in close temporal proximity. With shorter windows in ATW, the method prevents skewed matching results by vastly lowering the margin of error.

With the lookup maps in both directions and the timeline divided into appropriately sized windows, the next processing module iterates through the timeline using a sliding window approach. Initially, two indexes are used, with one at the start of the list and the other 11 indices before that (in this case, -10). The initial window does not contain any data, and with each iteration, the window slides one position forward in the chronological timeline. Reviews in the newly added window are included in the sliding window data.

Once the midpoint between the two sliding indices is within the array (in this case, when the lower index is at -5 and the higher index is at 5, giving a midpoint of 0), the module starts analyzing the reviews in the middle window. For each extension with reviews in the middle window, the module iterates through the reviews and adds a connection in a graph between the current review and any other review within the larger sliding window (containing 11 smaller windows). However, reviews within the same extension are ignored when looking for connections.

As the lower sliding index moves into the list, the module removes data from the windows that leave the larger sliding window, ensuring that the larger sliding window always contains 11 smaller windows. Connections continue to be added for reviews in the middle window.

Upon completing the iteration through the timeline, connections have been established between all potential pairs of temporally correlated reviews. Due to the presence of duplication and the fact that each review can only be matched to one other review, the next step is to apply discrete optimization to find the optimal solution. As mentioned in the method section, since the graph is bipartite and thus TUM, we take the initial graph, which would be an ILP, and relax it to an LP. Since the graph is TUM, we save computational complexity by doing this while retaining the optimal value when solving the LP.

Now that the correct and optimal values are calculated, filtering is done, and the results are finally presented in an output file, as described in the methods section.

4.5 Co-author

The Co-author module aims to identify groups of users who collaborate to write reviews for multiple extensions. Detecting these user clusters can uncover coordi-

nated attempts to manipulate extension reputations. The first method used is the *Efficient Maps* module that provides an index which is then narrowed down in the next module, as described in the methods section, to exclude single-users.

Next, a graph is generated with nodes representing users and edges representing the number of shared extensions for which the connected users have written reviews. The module iterates through all users, comparing each user's reviewed extensions with those of all other users. If a pair of users have reviewed the same extension, an edge is created between them in the graph if no edge exists. If the edge exists, the edge weight is incremented.

Various metrics to characterize the discovered user groups are calculated upon identifying the communities. These metrics include the size of each community, the average number of shared extensions among users within the community, and the proportion of user reviews relative to the total reviews in the community. These metrics provide valuable insights into the extent of potential review manipulation and allow for further analysis of the detected communities.

Finally, an output file is generated that includes the identified user communities, their associated metrics, and the extensions affected by these co-author groups. The information in this log could either be manually inspected as a lookup for communities or supplied to the *Flag Merge* method for further analysis.

5

Results

This chapter delves into the various presentation modules that produce results and offers insights into their findings. As described in the introduction, some details such as usernames, profile pictures, emails, and links are left out from certain results for ethical reasons.

5.1 Overview stats

The *Overview Stats* module calculates overview statistics that help determine the plausibility of observed events, differentiating between natural occurrences and potential reputation manipulation. Tables 5.1 and 5.2 present a snapshot of the Chrome Web Store as of February 2023.

Metric	**Value**
Total Extensions with reviews	55,107
Total Users	1,536,506
Total Reviews	1,782,702
Average review length (All users)	104.87
Average review length (Single-users)	85.18
Average review length (Multi-users)	311.25
Average reviews per multi-user	2.84
Median reviews per multi-user	2

Table 5.1: Overall data on February 2023 snapshot (Part 1)

Metric	**Single-users**	**Multi-users**
Users	1,402,687	133,819
Subset of total users	91.29%	8.71%
Reviews	1,402,687	380,015
Subset of total reviews	78.68%	21.32%
Extensions covered	49,411	29,161
Subset of total extensions	89.66%	52.92%

Table 5.2: Overall data on February 2023 snapshot (Part 2)

The metrics provided in Tables 5.1 and 5.2 give valuable insights into different user subsets, single-users and multi-users. The data also highlights review lengths, coverage of different user subsets, and average metrics for each user group. An interesting note is that 91.29% of users are single-users, which further emphasizes the performance gain from removing such users from indexing files if we know they are unimportant.

5.2 Review length

The *Review Length Graphs* module generates plots for different user subsets, as shown in Figure 5.1. The top left quadrant represents the length of reviews for all users, the top right quadrant represents single-users, and the bottom left quadrant represents multi-users. All results are displayed as percent due to the vast difference in the total number of entries for each user subset. The bottom right quadrant displays the combined results for easier comparisons.

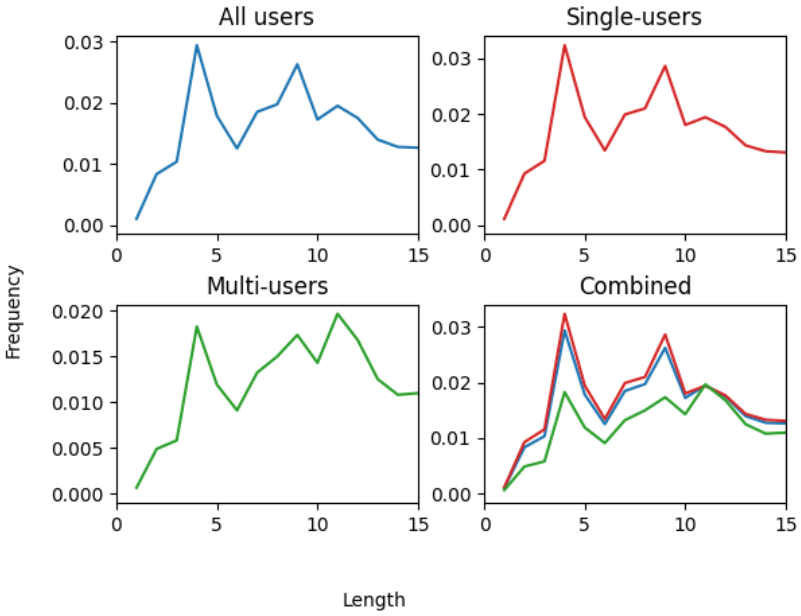

Figure 5.1: Length of review text by subsets of users, 0 to 15 on x-axis.

An intriguing pattern emerges from these plots, with spikes at very short reviews (4 and 9 characters). The spikes exist because many reviews contain short, simple positive words like “good”, “nice”, and “great”. However, the spikes appear at the same lengths in all groups, although they are not as pronounced in the multi-users graph, indicating that multi-users tend to write these short common words less. The fact that multi-users tend to write longer reviews is also supported by the data in Table 5.1.

5.3 Common messages

The frequent messages module extracts common messages based on user-defined filters. Table 5.3 lists the most common reviews.

Message	Occurrences
good	20,088
nice	8,759
very good	4,875
ok	4,832
great	4,810
love it	3,867
awesome	3,822
excelente	2,731
super	2,542
cool	2,508

Table 5.3: Top ten occurrences of common messages

While this information may not be highly beneficial, it does confirm the earlier-mentioned spikes in review length caused by these short, frequently written messages. A more informative approach is to filter out messages below a specific character count, such as 25, which is a value where all types of messages seem to flatten according to our review length plots (Figure 5.2).

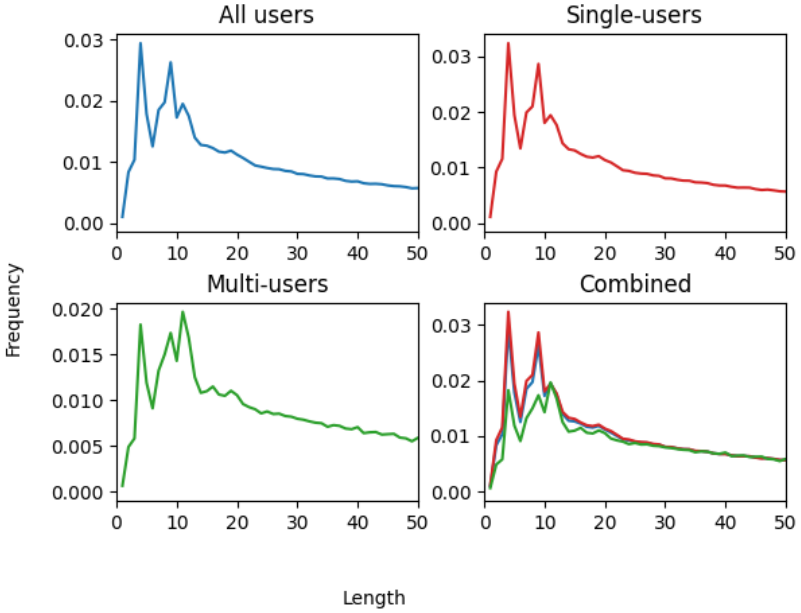

Figure 5.2: Length of review text by subsets of users, 0 to 50 on x-axis.

By applying this filter, we can identify obvious spam messages distributed across the web store, such as offers to buy extensions for monetization, contact requests,

and highly misspelled messages that are unlikely to have appeared that many times by chance, examples shown in Figure 5.3.

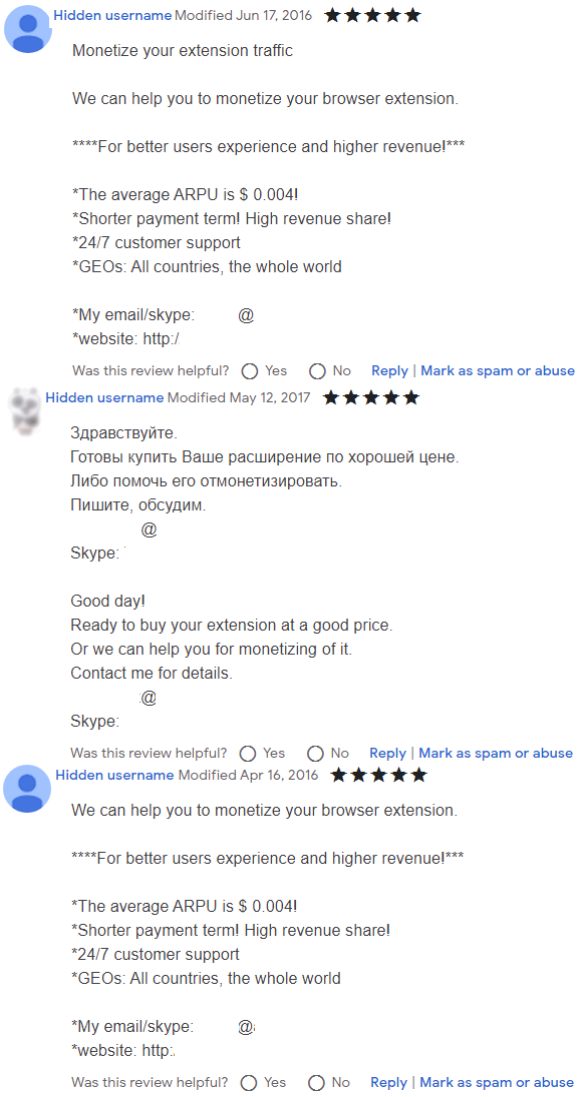

Figure 5.3: Monetization spam written on the extensions *ZeratoR*, *Blackfire Profiler* and *Ciuvo Price Comparison* in order from the top down. Emails and websites are removed from the review messages.

5.4 Written ratio

The *written ratio* method yields intriguing results, analyzing the proportion of written reviews compared to the total number of reviews for an extension. Table 5.4 presents some findings, showing the total ratings, written reviews, and written ratio for selected extensions.

The global statistics related to the *written ratio* offer intriguing insights. Table 5.5 displays the average written ratios for various subsets of extensions and percentile subset statistics. The first column indicates the threshold of reviews the extension

Name	Ratings	Written Reviews	Ratio
Opened or Not - Free Email Tracker	705	705	100%
TwitterScan - Find NFT Gems & Trending Tokens	384	384	100%
D365-UI-Test-Designer	141	141	100%
DigiNovo screen sharing for A1 shop	136	136	100%
AliExpress Search By Image | Rovalty	126	126	100%
Cashback beruby	116	116	100%
RippleHouse	103	103	100%
Jetstream	103	103	100%
BROSH for LinkedIn and Gmail	102	102	100%
Marucast Desktop Capture	99	99	100%

Table 5.4: Written ratio results

has to be strictly above to be included in the subset. One example is the first row with the threshold of 0, meaning that every extension is included in the subset, with an average of the entire subset in column two. The remaining columns now calculate an average on percentiles of this subset; looking at the data, for example, the 90'th percentile of the subset, which is the 90'th percentile of the entire set of extensions in our case, are all 100% written. The data reveals that extensions with a 100% written review ratio are highly improbable to occur naturally, especially in the subsets of extensions with above five reviews.

Threshold	Avg. subset	Avg. of 90^{th}	Avg. of 95^{th}	Avg. of 99^{th}
0	72.73%	100.0%	100.0%	100.0%
5	62.70%	93.33%	100.0%	100.0%
10	61.82%	91.67%	96.15%	100.0%
25	59.92%	89.25%	94.66%	100.0%
50	57.62%	87.10%	93.48%	99.17%
100	55.45%	84.66%	91.81%	99.03%
500	50.96%	78.84%	89.96%	99.07%
1000	48.96%	77.11%	89.48%	99.11%

Table 5.5: Table showing the relationship between thresholds and written ratios in different percentiles. The threshold is the number of reviews the extensions have to have more than be in the subset.

If we now consider the data shown in Table 5.5 and look back at Table 5.4, every single extension in Table 5.4 is at least in the top sub-one percent of their respective subset, which is extraordinary.

5.5 Aggregated Time Window

The ATW method uncovers numerous clusters with exceptionally high amounts of temporally shared reviews. One example of a cluster is shown in Table 5.6. This cluster contains many “New Tab” extensions, which are notoriously malicious and often involve query stealing.

Extension Name	Total Reviews	Connected Reviews
SimpleTab	11	11
TopTab	10	8
NWTab	10	7
Handy Tab	9	6
Summer Tab	10	6
Amazing Tab	10	6
ToDoTab	10	6
Charming Tab	11	6
AmTab	10	5

Table 5.6: ATW cluster containing "New Tab" extensions

The malicious nature of these extensions explains why someone would create accounts to promote them in hopes of stealing queries from more users. In this example, the cluster shown is confirmed to be query stealing. If we instead look at Figure 5.4, the examples depict visual examples of high-scoring clusters that are still under investigation. Notice the similarities in amount of reviews, which are closely related in amounts. They also share a temporal pattern of when the reviews were submitted, which is why ATW flags them.

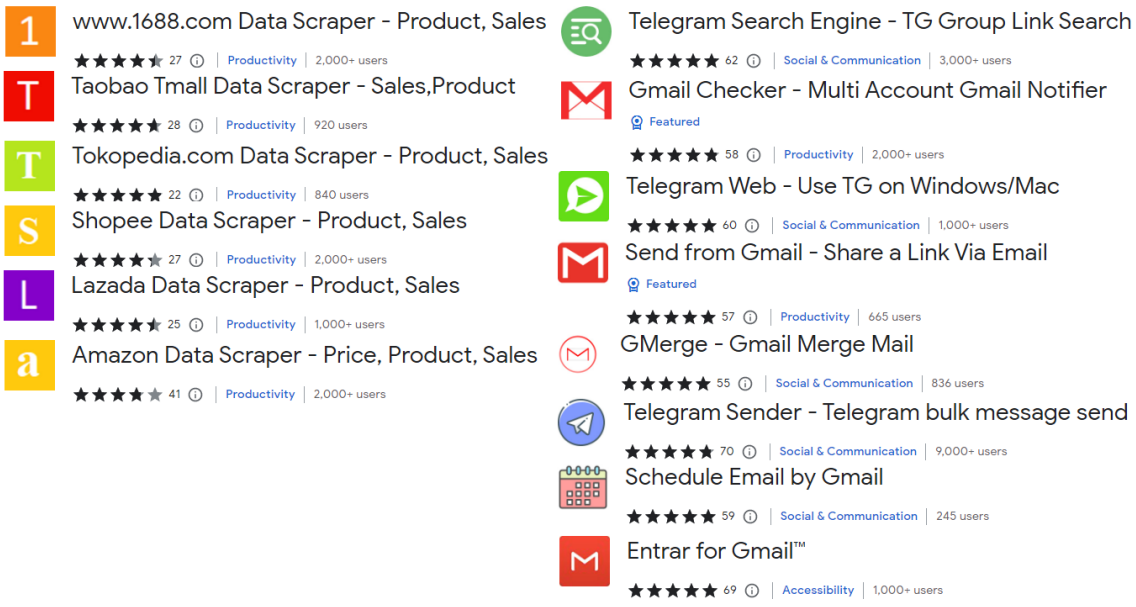

Figure 5.4: Visual example of two clusters found by the ATW method.

Figure 5.5 also compares reviews inside two extensions from the Gmail cluster. Notice that they have gotten reviews at the same time between extensions. In this case, it also happens to be the same users, resulting in this particular cluster being flagged by COA too.

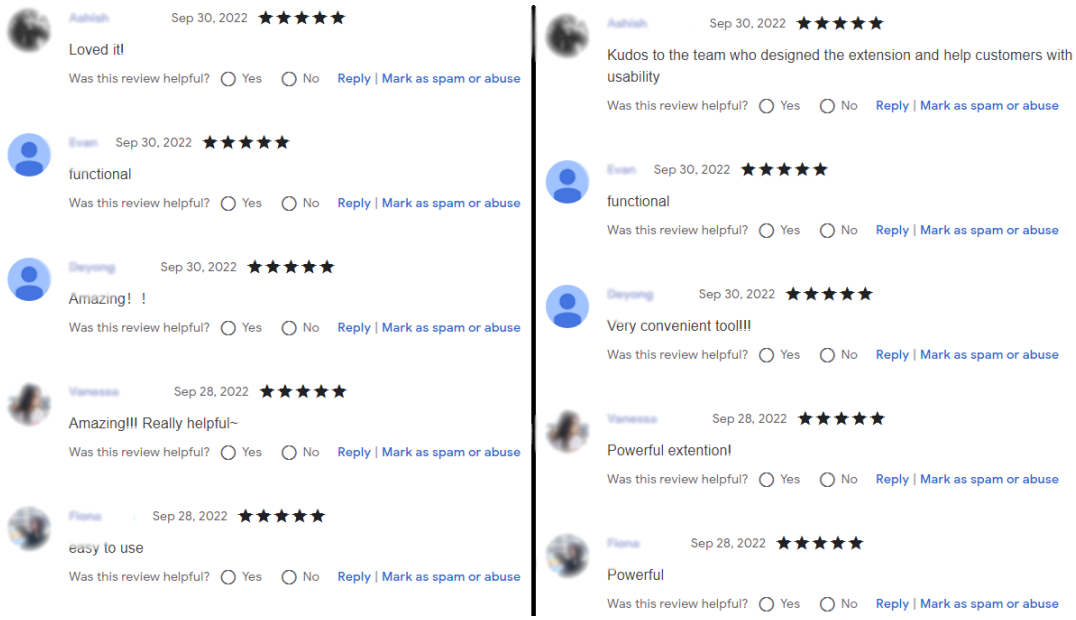

Figure 5.5: Temporally shared reviews between extensions, last names are removed from users for ethical reasons.

To help visualize the meaning behind temporally shared reviews, Table 5.7 shows the timestamp and the username from each review on two separate extensions. Note that there are some shared users even in this example, even though it would flag even if the users are entirely different, which some of them are. The two extensions in the example share one hundred percent of their reviews temporally, with deltas less than 3 minutes on every single one.

Date	**Ninja Cut Unblocked**		**X-Trial Racing Unblocked**		**Delta**
	Time	**Author**	**Time**	**Author**	
05/01	15:47:21	Caden Lee	15:49:37	Patricia Hall	2m 16s
16/01	10:32:32	Tracey Walsh	10:33:46	Monika Hernandez	1m 14s
17/01	15:53:06	Lea Marty	15:54:21	Ahsan Hoseen	1m 15s
23/01	15:23:44	Hobart Hernandez	15:24:36	Hobart Hernandez	52s
24/01	19:02:13	Claire Metcalfe	19:03:34	Bernadette Lambert	1m 21s
02/02	17:35:35	Mason Lewis	17:36:58	Mason Lewis	1m 23s
07/02	15:14:24	Aroni Ortize	15:15:36	Aroni Ortize	1m 12s

Table 5.7: Visualization of extensions with temporally shared reviews. The second and third column represents the time and author of each review on Ninja Cut Unblocked, and the same goes for column four and five, respectively, for X-Trial Racing Unblocked.

5.6 Co-author

As expected, the co-author analysis results reveal some overlap with the ATW method, as coordinated reputation manipulation on the same accounts across extensions creates co-author flags and ATW flags. The ATW flags occur when users manipulate the reputation simultaneously, creating a temporal correlation between their accounts. Table 5.8 presents one high-ranking cluster from the results with 76 users and four extensions.

Extension name	**Reviews from cluster**	**Ratio**
Search by Image on Aliexpress	73	96.05%
Just vpn	71	93.42%
Search by Image on Alibaba	70	92.11%
Product search by image	69	90.79%

Table 5.8: High scoring COA cluster with 76 users, showing the 4 top reviewed extensions in the cluster and the ratio of how many users from the cluster that reviewed that specific extension.

5.7 Spam Detection

The method of Spam Detection iterates through every extension and then their reviews, calculating the temporal gaps between the reviews to determine the reviews' density. If many reviews are close to each other temporally, the extension will be flagged by spam detection. In Table 5.9, an example log is shown where the method was run with the threshold of 5, meaning that every time the extension receives a review within 5 minutes after the last review was submitted, the spam count is increased by one. These numbers are incredibly high, supported by the vast top density. Also, notice that in the top ten, there are almost strictly crypto-related extensions besides Adobe, which has extraordinary amounts of spam reviews, especially compared to other top extensions on the Chrome web store. For a more visual example of how spammed reviews look on the Chrome web store, including the timestamp of the reviews, look at Figure 5.6.

Ranking	Name	Reviews
1	Ethos Sui Wallet	11479
2	Glass wallet | Sui wallet	4549
3	Sui Wallet	4542
4	Swash	4035
5	Price Tracker - Auto Buy, Price History	3942
6	Adobe Acrobat: PDF edit, convert, sign tools	3667
7	Fewcha Move Wallet	3067
8	FlipShope - Price Tracker Extension	2154
9	Morphis Wallet	2123
10	Bitfinity Wallet	1801
50	Sender Wallet	227
100	Get My Receipts by cloudHQ	65
250	Screence screen recorder	18
500	Taplio Stats	7

Table 5.9: Spam detection results with the threshold of five minutes.

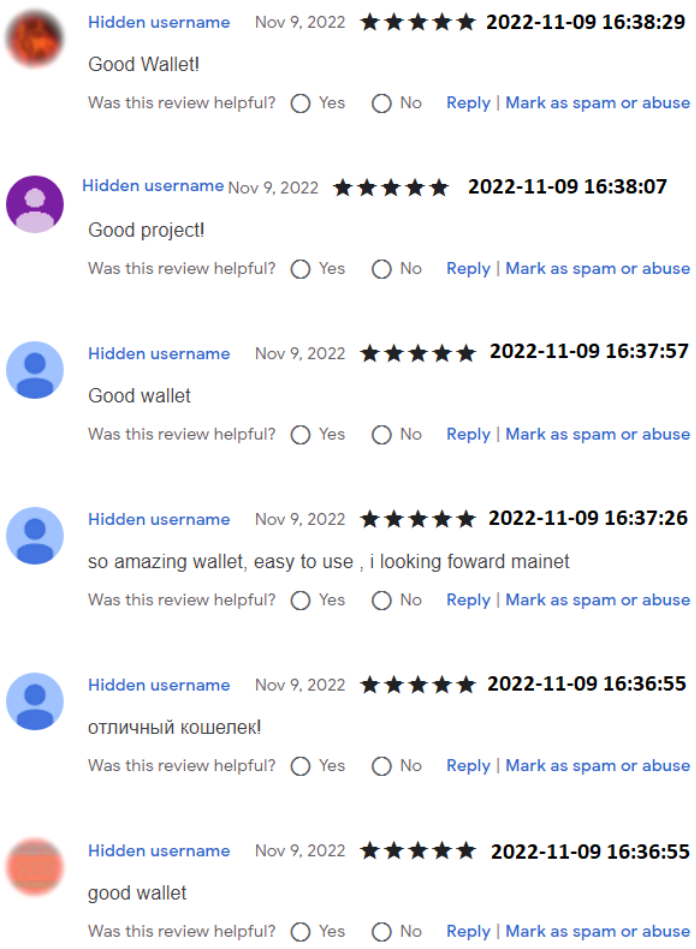

Figure 5.6: Spammed reviews with timestamps on the extension *Ethos Sui Wallet*.

5.8 Flag Merge

Flag Merge was created to merge the results from several methods to find extensions being flagged by several methods, thus increasing the certainty and filtering out false positives. The method loads the output from several other methods, namely ATW, COA, Written Ratio, and Spam detection. It creates a superset of all extensions contained in all the results and maps out what extensions are flagged by what methods and some statistics about the methods. The current version of the Flag Merge results contains the stats shown in Table 5.10. The top extensions, all flagged by all four methods, are shown in the Table 5.11.

Method	Total	New	ATW	COA	Written ratio	Spam
ATW	485	35	485	287	257	336
COA	1136	488	287	1136	439	403
Written	3187	1918	257	439	3187	1039
Spam	3394	2105	336	403	1039	3394

Table 5.10: Overlap of flagged extensions.

Extension Name	Reviews	Flags			
DigiNovo screen sharing for A1 shop	136	X	X	X	X
Search by Image on Aliexpress	104	X	X	X	X
Search by Image on Alibaba	100	X	X	X	X
Discordmate - Discord Chat Exporter	82	X	X	X	X
dmsave - Dailymotion video downloader	71	X	X	X	X

Table 5.11: Results from Flag merge containing the number of reviews as well as marks on which flags that specific extension got, the flags are ATW, COA, Written Ratio, and Spam Detection in order from left to right.

5.9 Maliciousness

To identify malicious extensions, we primarily employ two strategies. First, we investigate the extensions that rank prominently in the output generated by *Flag Merge*. Secondly, we capitalize on the capabilities of the *COA* and *ATW* methods, which generate clusters of extensions exhibiting similar patterns indicative of reputation manipulation. By doing so, we can pinpoint new malicious extensions based on the characteristics of known culprits.

One illustrative example of implementing the first approach involved examining the extensions at the top of the flagged list for “query stealers" - notorious extensions that slyly extract users’ search queries without their consent. One particular breed of extensions known for indulging in this is the “New Tab" extensions, which hijack the browser’s home tab, replacing it with an alternative that modifies its functionality and appearance.

Currently, there are 111 extensions flagged by all four algorithms in *Flag Merge*. Among them, 13 were classified as “New Tab" extensions and, consequently, subjected to a thorough manual inspection. Astoundingly, each of these 13 turned out to be query stealers. This discovery led to a further exploration of the clusters these culprits were associated with, according to the ATW and COA methods. Following an exhaustive analysis of these clusters, the tally of malicious query stealers swelled to 26. Interestingly, these extensions were dispersed across four distinct clusters, with the largest cluster harboring 16 of the 26 extensions.

Turning to the second strategy, which involves leveraging a list of known malicious extensions, we sought assistance from Wladimir Palant, a security blogger who frequently discusses malicious extensions on his blog. Utilizing a list extracted from one of his articles [26], we compared it against the results from *ATW* and *COA*.

The extracted list featured 18 malicious extensions, of which 16 were flagged by at least one of our methods and subsequently included in the *Flag Merge* output list. On delving into the clusters in *COA* and *ATW*, we found that the union of clusters having at least one of our 16 extensions comprised 40 extensions in total. We presented this list to Wladimir for further analysis. Following an in-depth investigation, he determined that 18 of the 40 extensions contained the same malicious code. Furthermore, an additional eight were found to contain an improved version of the code, compatible with the new manifest v3 a subject Wladimir tackled in another post [27].

6

Related work

The landscape of browser extension security has witnessed significant advancements thanks to numerous studies exploring various aspects of this domain. These studies have made substantial strides in their respective areas, each providing unique insights and expanding the understanding of browser extension security.

Picazo-Sanchez et al. [8] leveraged download patterns as a critical signal for analyzing browser extensions to find malicious extensions. Even though this paper focuses on an entirely different aspect, namely the downloads, the critical insight is that malicious actors care about the extension's reputation on the centralized repository. The amount of downloads certainly is one form of reputation, another obvious one, which we focus on in this study, is the review system.

Similarly, Pantelaios et al. [9] delivered a comprehensive analysis of browser extensions with their two-stage system that leveraged anomalous extension ratings, code changes, and keyword pattern matching. They successfully uncovered a broad range of extension abuses. However, their methods were limited to strictly include extensions with 50 reviews or more, which is not a limitation we have.

Aggarwal et al. [11] focused on a different aspect, developing a robust machine-learning model to detect spying extensions. This approach relied on the sequence of browser API calls, demonstrating high precision and recall in detecting spying extensions. Methods like these are perfect candidates for the follow-up part of this study, finding malicious extensions that could be used when traversing the result clusters to find more malicious extensions.

Benjamin et al. [12] conducted an in-depth empirical study uncovering 4410 query stealing "New Tab" extensions, or as they call it, traffic stealing extensions. Their research provided a foundational understanding of query stealing and how it can be detected, which is how query stealers were manually detected in this study.

Despite the significant progress in browser extension security, the focus on user reviews and reputation manipulation in the Google Chrome Web Store, as highlighted in our research, represents a new and necessary direction. Our work complements these prior studies, filling a gap in the literature by utilizing novel methods to detect reputation manipulation.

7

Conclusion

In this study, we proposed a comprehensive framework of novel methods for detecting reputation manipulation among browser extensions. The framework leverages various methods, including the ATW, COA, Written Ratio, and Spam Detection, to identify extensions that may have been the target of reputation manipulation, willingly or unwillingly. The methods used to detect such behavior answer our first research question, "*Does the Chrome Web Store contain reviews meant to manipulate reputation? what are efficient techniques to detect such reviews?*", which it certainly does.

While detecting reputation manipulation does not necessarily imply malicious intent, it is essential to scrutinize such extensions further to determine if they pose any security risks. The Flag Merge method serves as an effective tool for identifying top candidates with a high likelihood of being malicious. Flag Merge results are then used to answer our second research question, "*Could methods that detect reputation manipulation be used to identify malicious extensions?*"

For instance, the "Primary Tab" extension was found to be a malicious query stealer that was flagged on all methods. Upon identifying a confirmed malicious extension, our framework can be utilized to trace back and identify other related extensions through cluster analysis. In the case of "Primary Tab", the investigation revealed a cluster of "New Tab" extensions with manipulated reputations, which all were later confirmed malicious upon manual inspection. The cluster of extensions based on the "Primary Tab" extension contained 16 malicious "New Tab" extensions, with no false positives in the cluster.

In conclusion, the Flag Merge method efficiently pinpoints potential malicious extensions, which should be inspected. After inspection, the other methods in the framework serve as powerful tools for identifying similar extensions and their relationships. We mainly looked at query stealers, but any analysis could be done on the potentially malicious extensions, and then the other methods would help uncover similar extensions. As a result, our framework enables the swift discovery of numerous malicious extensions by flagging the most suspicious ones. It then allows the user to quickly find similar extensions after the first malicious extension is identified, ultimately enhancing the security of browser extensions for all users.

Bibliography

[1] *Chrome web store*. [Online]. Available: https://chrome.google.com/webstore/category/extensions.

[2] [Online]. Available: https://chrome-stats.com/t/extension.

[3] C. Gartenberg, *Google announces changes to chrome web store policies to help fight spammy extensions*, Apr. 2020. [Online]. Available: https://www.theverge.com/2020/4/30/21242597/google-chrome-web-store-new-spam-policy-extensions.

[4] *Keeping spam off the chrome web store*, Apr. 2020. [Online]. Available: https://blog.chromium.org/2020/04/keeping-spam-off-chrome-web-store.html.

[5] C. Summerson, *How to make sure a chrome extension is safe before installing it*, en, https://www.howtogeek.com/347429/how-to-make-sure-a-chrome-extension-is-safe-before-installing-it/, Accessed: 2022-12-12, Apr. 2018.

[6] C. Lurey, *Are all chrome extensions secure?* en, https://www.keepersecurity.com/blog/2022/10/19/are-all-chrome-extensions-secure/, Accessed: 2022-12-12, Oct. 2022.

[7] *Browser extensions: How to vet and install safely*, en, https://security.berkeley.edu/education-awareness/browser-extensions-how-vet-and-install-safely, Accessed: 2022-12-12.

[8] P. Picazo-Sanchez, B. Eriksson, and A. Sabelfeld, "No signal left to chance: Driving browser extension analysis by download patterns," in *Proceedings of the 38th Annual Computer Security Applications Conference*, ser. ACSAC '22, Austin, TX, USA: Association for Computing Machinery, 2022, pp. 896–910, ISBN: 9781450397599. DOI: 10.1145/3564625.3567988. [Online]. Available: https://doi.org/10.1145/3564625.3567988.

[9] N. Pantelaios, N. Nikiforakis, and A. Kapravelos, "You've changed: Detecting malicious browser extensions through their update deltas," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 477–491.

[10] S. Arshad, A. Kharraz, and W. Robertson, "Identifying extension-based ad injection via fine-grained web content provenance," in *International Symposium on Research in Attacks, Intrusions, and Defenses*, Springer, 2016, pp. 415–436.

[11] A. Aggarwal, B. Viswanath, L. Zhang, S. Kumar, A. Shah, and P. Kumaraguru, "I spy with my little eye: Analysis and detection of spying browser exten-

sions," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2018, pp. 47–61.

[12] B. Eriksson, P. Picazo-Sanchez, and A. Sabelfeld, "Hardening the security analysis of browser extensions," in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '22, Virtual Event: Association for Computing Machinery, 2022, pp. 1694–1703, ISBN: 9781450387132. DOI: 10.1145/3477314.3507098. [Online]. Available: https://doi.org/10.1145/3477314.3507098.

[13] *Reviews*, en, https://getreviews.buzz/, Accessed: 2022-12-12, Aug. 2020.

[14] *Free or buy google reviews, amazon reviews & more*, en, https://www.reviewsub.com/, Accessed: 2022-12-12.

[15] *Buy positive reviews online at cheap prices on review community*, en, https://reviewgg.com/, Accessed: 2022-12-12, Jan. 2022.

[16] *Architecture overview*. [Online]. Available: https://developer.chrome.com/docs/extensions/mv2/architecture-overview/.

[17] *Content scripts*. [Online]. Available: https://developer.chrome.com/docs/extensions/mv2/content_scripts/.

[18] *Manage events with background scripts*. [Online]. Available: https://developer.chrome.com/docs/extensions/mv2/background_pages/.

[19] *Ad injection*. [Online]. Available: https://sourcedefense.com/glossary/ad-injection/.

[20] D. R. Staff, *Malicious chrome extensions plague 1.4m users*, Aug. 2022. [Online]. Available: https://www.darkreading.com/vulnerabilities-threats/1-4m-users-running-malicious-chrome-extensions.

[21] B. N, *Hundreds of malicious chrome browser extensions with used for stealing user sensitive data - 32 million users affected*, Jun. 2020. [Online]. Available: https://cybersecuritynews.com/hundreds-of-malicious-chrome-extensions/.

[22] *Warning: Fake one-star reviews amp; ratings are bombarding freeaddon extensions!* Sep. 2020. [Online]. Available: https://freeaddon.com/fake-1-star-ratings-reviews-attack-by-hackers/.

[23] J. Matouek and B. Gärtner, *Understanding and Using Linear Programming* (Universitext). Springer, 2007. [Online]. Available: https://link.springer.com/book/10.1007/978-3-540-30717-4.

[24] J. Matouek and B. Gärtner, "What is it, and what for?" In *Understanding and Using Linear Programming* (Universitext), Universitext. Springer, 2007, pp. 1–10. [Online]. Available: https://link.springer.com/book/10.1007/978-3-540-30717-4.

[25] J. Matouek and B. Gärtner, "Integer programming and lp relaxation," in *Understanding and Using Linear Programming* (Universitext), Universitext. Springer, 2007, pp. 29–40. [Online]. Available: https://link.springer.com/book/10.1007/978-3-540-30717-4.

[26] W. Palant, *More malicious extensions in chrome web store*, https://palant.info/2023/05/31/more-malicious-extensions-in-chrome-web-store/, Accessed: June 10, 2023, 2023.

[27] W. Palant, *How malicious extensions hide running arbitrary code*, https://palant.info/2023/06/02/how-malicious-extensions-hide-running-arbitrary-code/, Accessed: June 10, 2023, 2023.