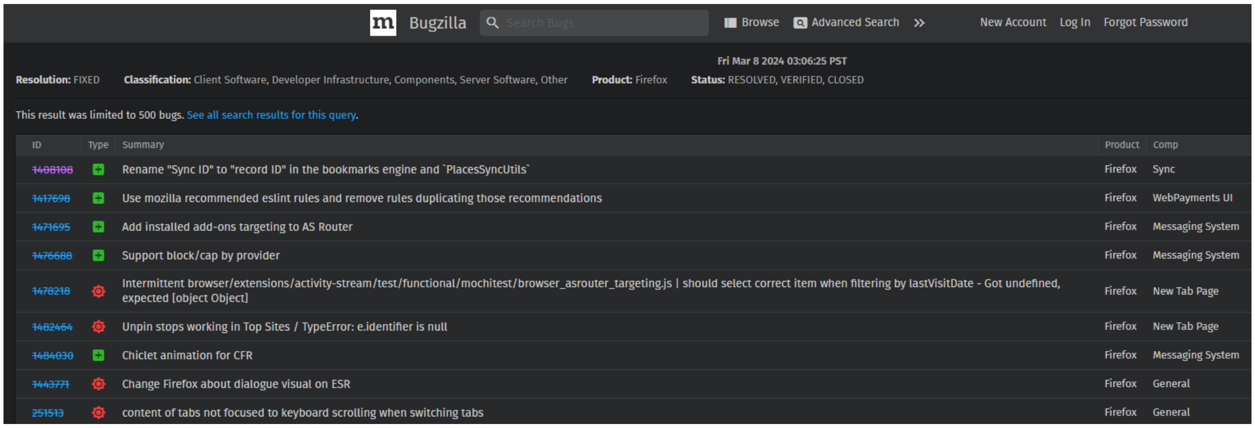
Mozilla Firefox Bug Analysis

Mario-Mihai Mateaş, 4th year student, University Politehnica of Timişoara

I. Data mining through Bugzilla, Mozilla's unique and powerful issue-tracking system

Firefox, together with all of the other Mozilla products are keeping track of their issues in a mature, stable and detailed bug tracking system, called Bugzilla. Bugzilla offers several REST API endpoints that allow us to collect important data from the reported issues.



Fri Mar 8 2024 03:06:25 PST

Resolution: FIXED **Classification:** Client Software, Developer Infrastructure, Components, Server Software, Other **Product:** Firefox **Status:** RESOLVED, VERIFIED, CLOSED

This result was limited to 500 bugs. See all search results for this query.

ID	Type	Summary	Product	Comp
~~1408108~~	⊞	Rename "Sync ID" to "record ID" in the bookmarks engine and `PlacesSyncUtils`	Firefox	Sync
~~1417698~~	⊞	Use mozilla recommended eslint rules and remove rules duplicating those recommendations	Firefox	WebPayments UI
~~1471695~~	⊞	Add installed add-ons targeting to AS Router	Firefox	Messaging System
~~1476688~~	⊞	Support block/cap by provider	Firefox	Messaging System
~~1478218~~	⚙	Intermittent browser/extensions/activity-stream/test/functional/mochitest/browser_asrouter_targeting.js | should select correct item when filtering by lastVisitDate - Got undefined, expected [object Object]	Firefox	New Tab Page
~~1482464~~	⚙	Unpin stops working in Top Sites / TypeError: e.identifier is null	Firefox	New Tab Page
~~1484030~~	⊞	Chiclet animation for CFR	Firefox	Messaging System
~~1443771~~	⚙	Change Firefox about dialogue visual on ESR	Firefox	General
~~251513~~	⚙	content of tabs not focused to keyboard scrolling when switching tabs	Firefox	General

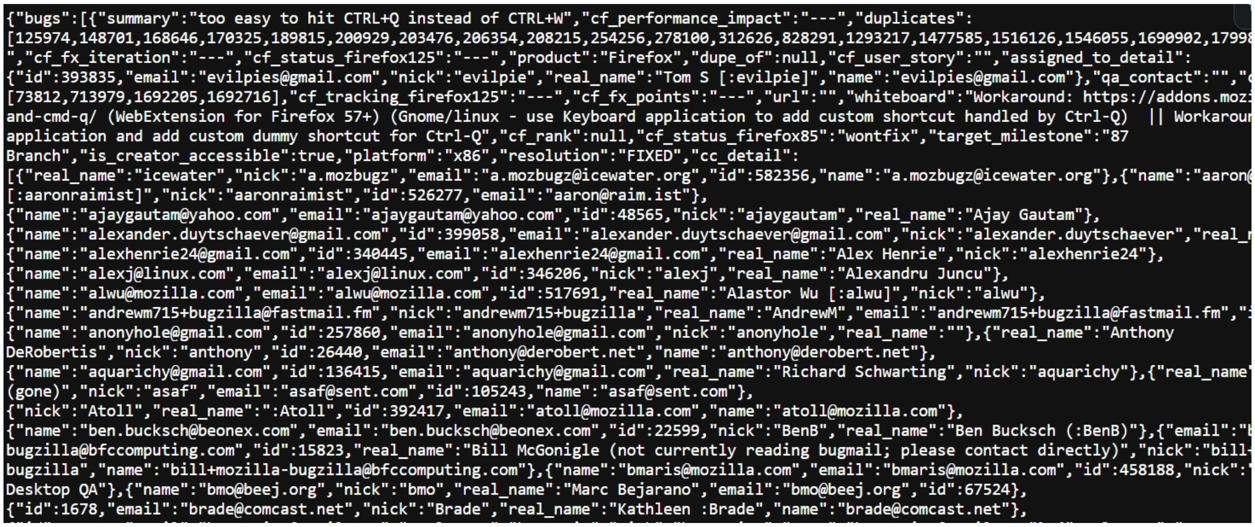
We are going to use Python in order to send requests to Bugzilla and generate easy-to-read graphs out of advanced data structures. The endpoint we are going to use is:

```
https://bugzilla.mozilla.org/rest/bug?resolution=FIXED&bug_status=RESOLVED&bug_status=VERIFIED&bug_status=CLOSED&classification=Client%20Software&classification=Developer%20Infrastructure&classification=Components&classification=Server%20Software&classification=Other&query_format=advanced&product=Firefox&limit=HERE_GOES_THE_LIMIT&offset=HERE_GOES_THE_OFFSET
```

… meaning that we are going to analyze the issues reported for the **Mozilla Firefox** component.

We have introduced several parameters (such as **bug_status**, in order to make sure that the API will return only fixed issues). Without mentioning the last parameter (**limit**), Bugzilla's API will try to return the whole collection of issues reported to Firefox. Such an action is going to overload the server, and our request will, after some seconds, be rejected. We will send multiple requests with a limit of 400 and with increasing values of the **offset** parameter.

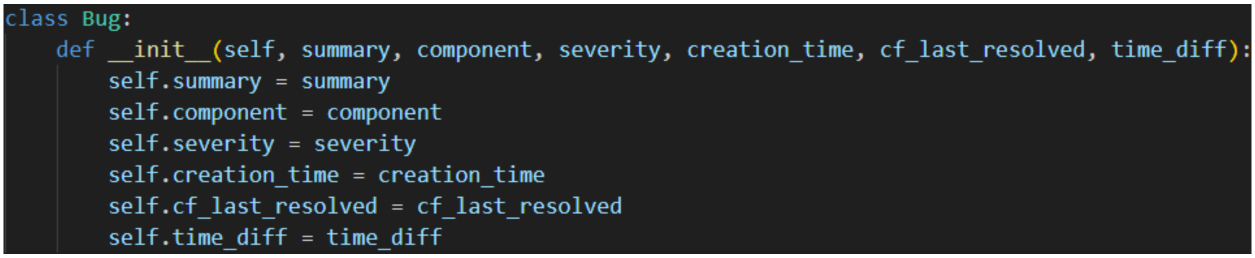
The answer to such a query may look very complicated though, meaning that it is pretty hard to grasp data out of a raw JSON response that looks like this:



```
{"bugs":[{"summary":"too easy to hit CTRL+Q instead of CTRL+W","cf_performance_impact":"---","duplicates":
[125974,148701,168646,170325,189815,200929,203476,206354,208215,254256,278100,312626,828291,1293217,1477585,1516126,1546055,1690902,17998
","cf_fx_iteration":"---","cf_status_firefox125":"---","product":"Firefox","dupe_of":null,"cf_user_story":"","assigned_to_detail":
{"id":393835,"email":"evilpies@gmail.com","nick":"evilpie","real_name":"Tom S [:evilpie]","name":"evilpies@gmail.com"},"qa_contact":"","c
[73812,713979,1692205,1692716],"cf_tracking_firefox125":"---","cf_fx_points":"---","url":"","whiteboard":"Workaround: https://addons.mozi
and-cmd-q/ (WebExtension for Firefox 57+) (Gnome/linux - use Keyboard application to add custom shortcut handled by Ctrl-Q)  || Workarour
application and add custom dummy shortcut for Ctrl-Q","cf_rank":null,"cf_status_firefox85":"wontfix","target_milestone":"87
Branch","is_creator_accessible":true,"platform":"x86","resolution":"FIXED","cc_detail":
[{"real_name":"icewater","nick":"a.mozbugz","email":"a.mozbugz@icewater.org","id":582356,"name":"a.mozbugz@icewater.org"},{"name":"aaron@
[:aaronraimist]","nick":"aaronraimist","id":526277,"email":"aaron@raim.ist"},
{"name":"ajaygautam@yahoo.com","email":"ajaygautam@yahoo.com","id":48565,"nick":"ajaygautam","real_name":"Ajay Gautam"},
{"name":"alexander.duytschaever@gmail.com","id":399058,"email":"alexander.duytschaever@gmail.com","nick":"alexander.duytschaever","real_r
{"name":"alexhenrie24@gmail.com","id":340445,"email":"alexhenrie24@gmail.com","real_name":"Alex Henrie","nick":"alexhenrie24"},
{"name":"alexj@linux.com","email":"alexj@linux.com","id":346206,"nick":"alexj","real_name":"Alexandru Juncu"},
{"name":"alwu@mozilla.com","email":"alwu@mozilla.com","id":517691,"real_name":"Alastor Wu [:alwu]","nick":"alwu"},
{"name":"andrewm715+bugzilla@fastmail.fm","nick":"andrewm715+bugzilla","real_name":"AndrewM","email":"andrewm715+bugzilla@fastmail.fm","i
{"name":"anonyhole@gmail.com","id":257860,"email":"anonyhole@gmail.com","nick":"anonyhole","real_name":""},{"real_name":"Anthony
DeRobertis","nick":"anthony","id":26440,"email":"anthony@derobert.net","name":"anthony@derobert.net"},
{"name":"aquarichy@gmail.com","id":136415,"email":"aquarichy@gmail.com","real_name":"Richard Schwarting","nick":"aquarichy"},{"real_name"
(gone)","nick":"asaf","email":"asaf@sent.com","id":105243,"name":"asaf@sent.com"},
{"nick":"Atoll","real_name":":Atoll","id":392417,"email":"atoll@mozilla.com","name":"atoll@mozilla.com"},
{"name":"ben.bucksch@beonex.com","email":"ben.bucksch@beonex.com","id":22599,"nick":"BenB","real_name":"Ben Bucksch (:BenB)"},{"email":"b
bugzilla@bfccomputing.com","id":15823,"real_name":"Bill McGonigle (not currently reading bugmail; please contact directly)","nick":"bill+
bugzilla","name":"bill+mozilla-bugzilla@bfccomputing.com"},{"name":"bmaris@mozilla.com","email":"bmaris@mozilla.com","id":458188,"nick":"
Desktop QA"},{"name":"bmo@beej.org","nick":"bmo","real_name":"Marc Bejarano","email":"bmo@beej.org","id":67524},
{"id":1678,"email":"brade@comcast.net","nick":"Brade","real_name":"Kathleen :Brade","name":"brade@comcast.net"},
```

A solution for this would be to filter some of the data and use several popular Python libraries in order to make it more readable.

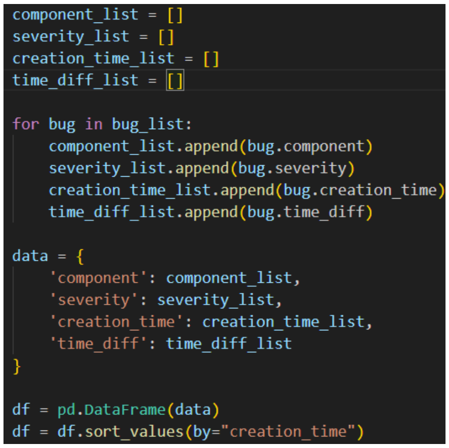
As stated above, we will only extract the fields that are the most important for us. We'll save the **summary** (equivalent to the title of the issue), the **component** the issue is associated with, the **severity** (normal, medium, critical, blocker) of the issue, the **creation_time** of the report, the **cf_last_resolved** (which is the resolution time of the issue) and the difference between those two timestamps, stored in the **time_diff** field. We have created a specialized class (**Bug**), in order to collect the data in a more organized manner:



```python
class Bug:
    def __init__(self, summary, component, severity, creation_time, cf_last_resolved, time_diff):
        self.summary = summary
        self.component = component
        self.severity = severity
        self.creation_time = creation_time
        self.cf_last_resolved = cf_last_resolved
        self.time_diff = time_diff
```

The **time_diff** data will be converted into hours for better readability.

The collected data will be stored in a **pandas** dataframe, ordered by the issue creation date.



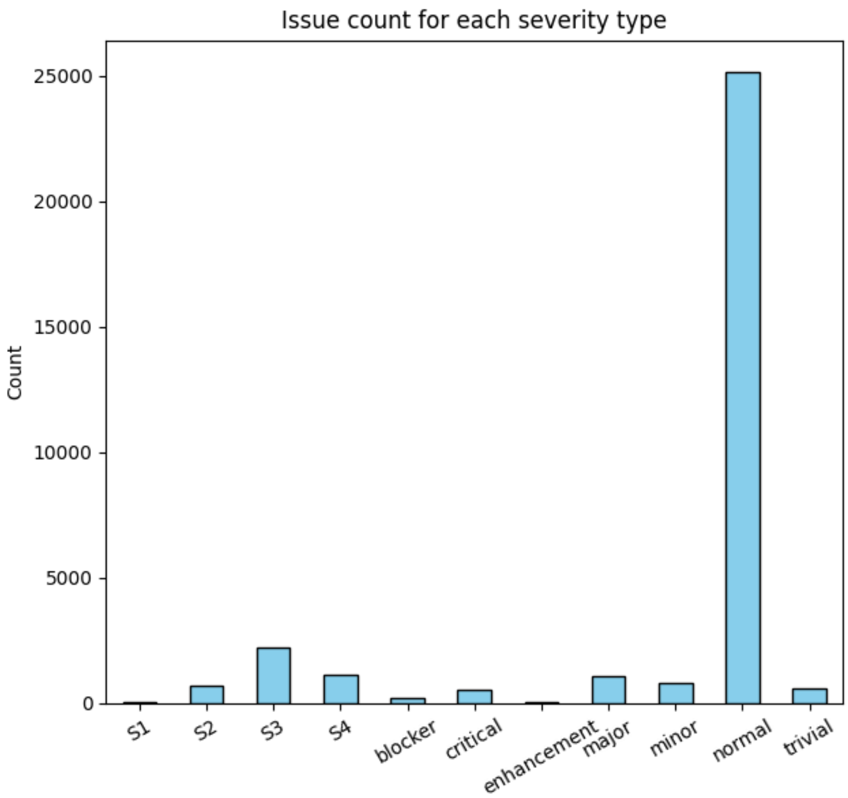
```python
component_list = []
severity_list = []
creation_time_list = []
time_diff_list = []

for bug in bug_list:
    component_list.append(bug.component)
    severity_list.append(bug.severity)
    creation_time_list.append(bug.creation_time)
    time_diff_list.append(bug.time_diff)

data = {
    'component': component_list,
    'severity': severity_list,
    'creation_time': creation_time_list,
    'time_diff': time_diff_list
}

df = pd.DataFrame(data)
df = df.sort_values(by="creation_time")
```

We will work with this dataset in order to create insightful graphs through the **matplotlib** module for Python.

We have also done some clearing among the data collected, meaning that blank data has been removed. Also, insignificant Firefox components have been removed in order to improve readability of the plots.

II. The actual analysis process

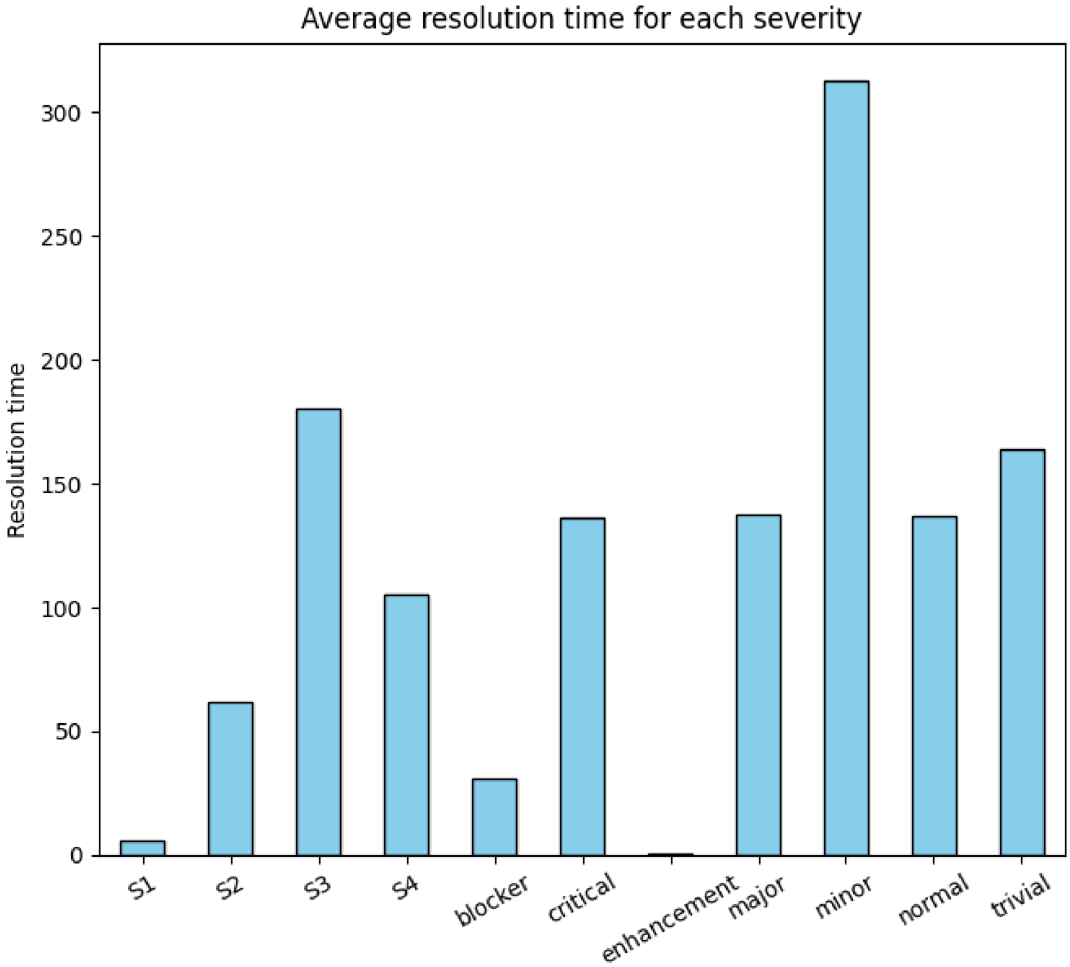
We will start by looking at the severity types of the issues reported. We will plot a graph of the severities, together with the number of bugs with the corresponding severity.



This is a normal distribution. Nothing seems wrong here. From a total of over 45000 bugs collected, most of them are of normal severity. Severities labeled “S1”, “S2” and “S3” might be newer severity types that correspond to certain priority levels. “S3” might be equivalent to the “normal” severity[1].

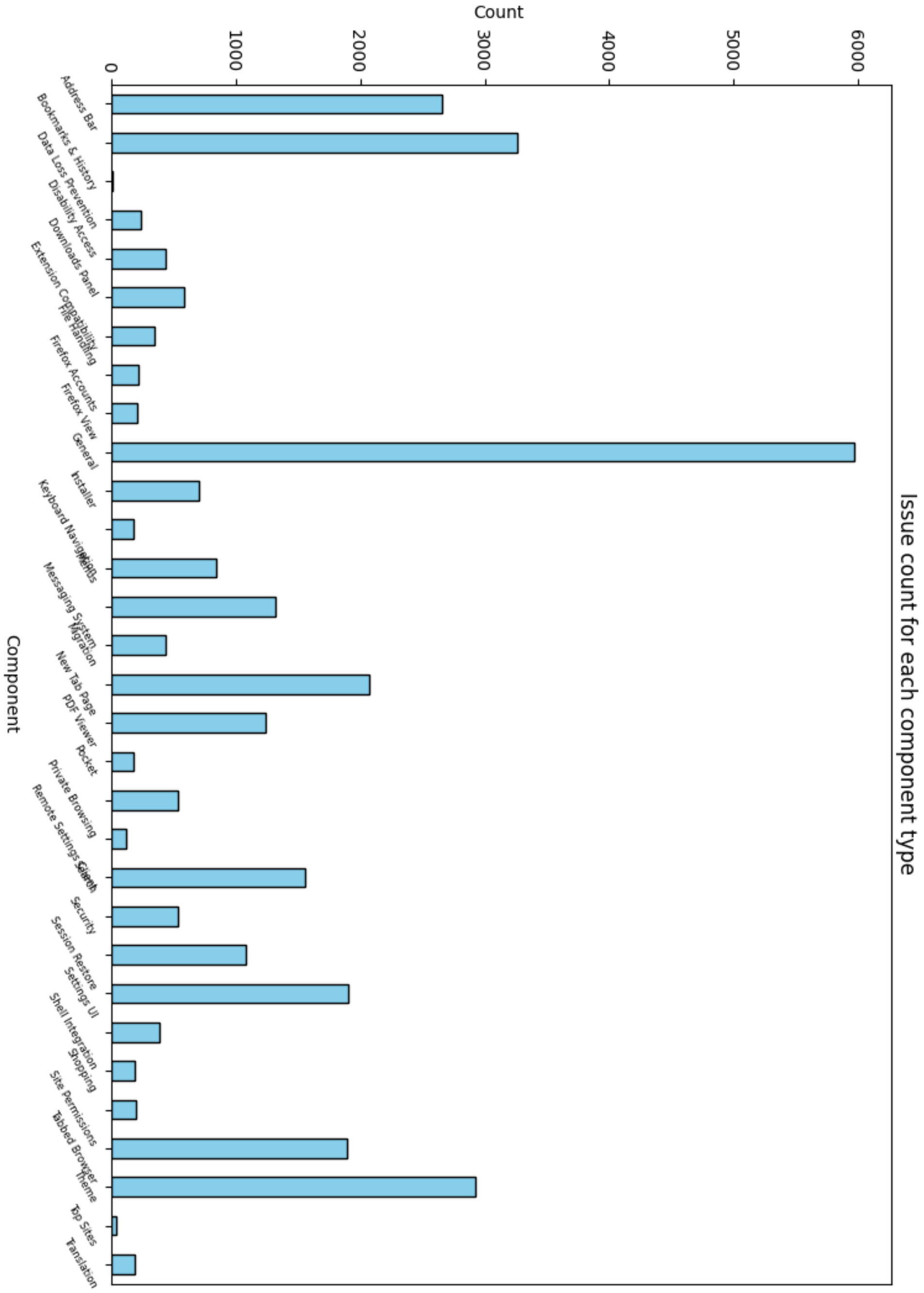
Another factor that we should analyze in relation with the severity of the issues is the number of hours spent on resolving the bugs.

[1] https://wiki.mozilla.org/BMO/UserGuide/BugFields, last accessed 08.03.2024

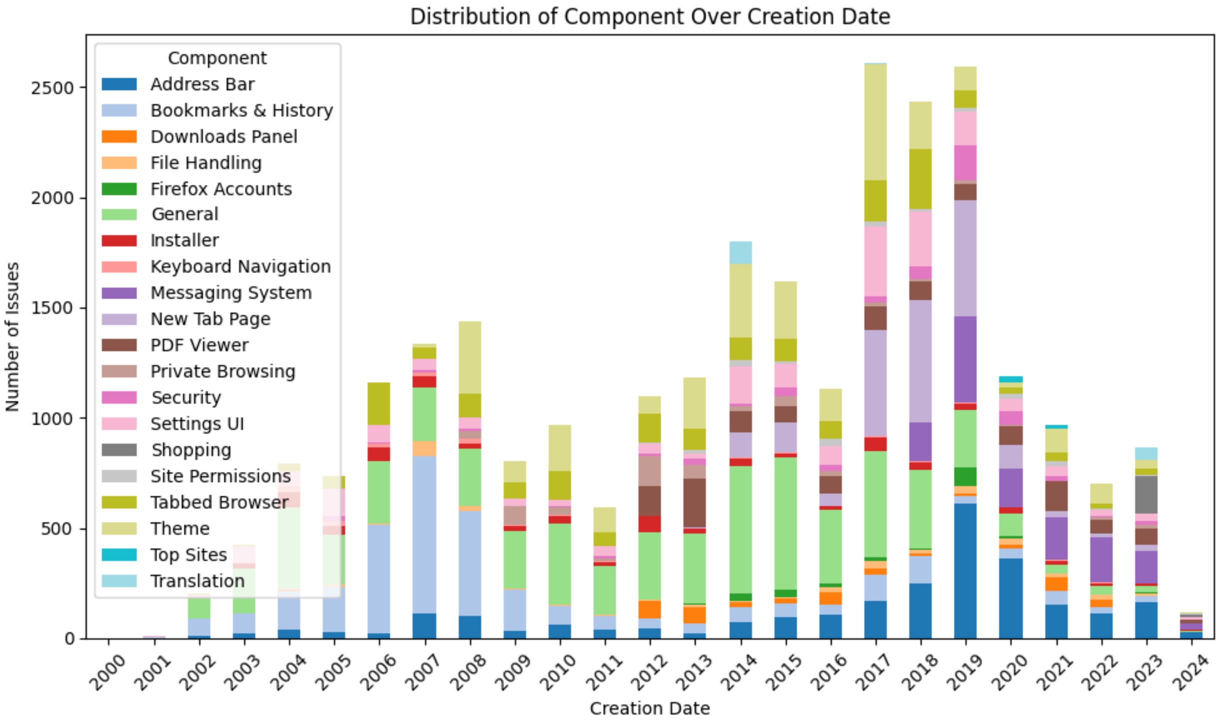


The above graph depicts the average resolution time for each recorded severity. The time has been converted into days. We can observe that the "blocker" issues take less time to solve as they are prioritized over the other issues. This seems normal. The issues marked as "minor" seem to take more time to be solved, compared to the "normal" ones. The average resolution time of the "trivial" severities seems to be close to the time calculated for "normal", meaning that there is a possibility of bad estimation or categorization of certain severities to their corresponding levels. Also, during our analysis, we have noticed that there are several bugs that have been reported at the beginning of Mozilla Firefox's development stage (2003-2004), and closed several years ago (2022-2023). Most of these issues have been reported as "minor". This means, such defects could be classified in one of the two:

1. the defect had been so insignificant, it has been moved for over 20+ years in the backlog;
2. the defect was reported through another system (version), and the issue has been fixed in the past, but it was only marked recently as "closed".



The above graph depicts the number of issues associated with each type of “component”. A component is usually a feature integrated inside Firefox. We can see that this graph is displaying expected statistics: issues related to the “General” component are the most frequently reported, while features that are not so common for a Web Browser (such as “File Handling”, “Migration” or “Shell Integration”) have a lesser number of reports.

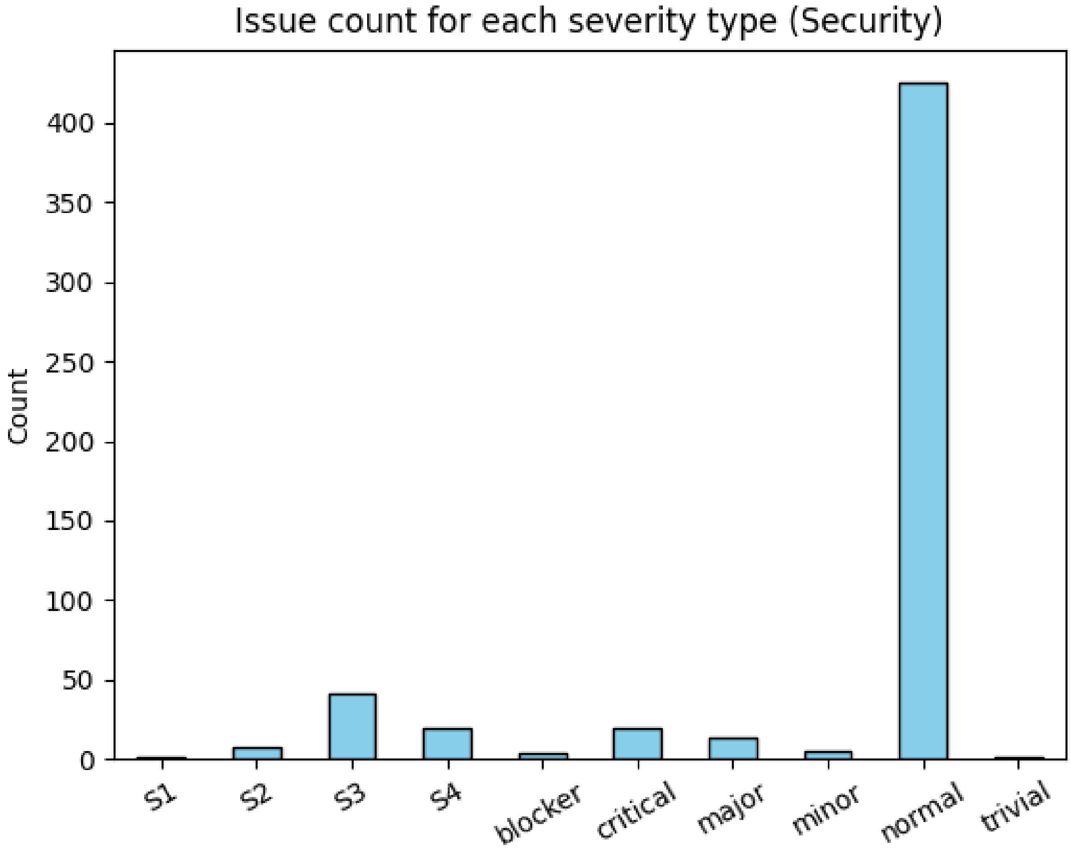


The distribution above displays Firefox’s focus on components over a span of 24 years. 2024 features a lesser number of issues, because the data has been collected on March 9, only 3 months after the start of the year.

We can see that there was a spike of issues reported in 2017, compared to its predecessor. Increasing issues have been reported related to the “New Tab Page” component, and this is, indeed, what we expected, because Firefox revamped its new tab page in the same year[2]. 2017 also represents the year of Firefox’s Quantum Project release[3]. 2018 and 2019 also had a large number of reports, and most of them also featured the “New Tab Page”.

We can also notice that most of the security features have been reported in 2019, although security issues weren’t common during the years. We can do further analysis, in order to check how the security features have been prioritized.

[2] https://www.mozilla.org/en-US/firefox/57.0/releasenotes/, last accessed 09.03.2024
[3] https://www.mozilla.org/en-US/firefox/browsers/quantum/, last accessed 09.03.2024



There is not a huge difference between this distribution and the distribution of all the issues reported. However, we can notice that security issues are less likely to be reported as "trivial" or "minor", which is also good.

III. Conclusions

From our analysis, we can conclude the following things:

1. There wasn't a huge number of security issues reported regarding Firefox over a 20+ year span. This could mean that the developers took great care of the security aspects during the implementation of new features or defect fixing.
2. There was a huge increase of reported issues in years 2017, 2018 and 2019, compared to the other years, and this was mostly related to several important decisions regarding Firefox (e.g. important updates, new projects, browser redesign).
3. Issue severity levels are, indeed, taken very seriously at Mozilla. "Minor" fixes are indeed considered as minor and can be postponed even for 20 years[4].
4. There might be some problems regarding marking issues with a certain severity level ("trivial" issues tend to have a very high resolution time – but this can also be due to postponing).

[4] https://bugzilla.mozilla.org/show_bug.cgi?id=52821, last accessed 09.03.2024