



Assignment of master's thesis

Title:	Archival tool for the Discord communications platform
Student:	Bc. David Labský
Supervisor:	Ing. Milan Dojčinovski, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2022/2023

Instructions

Discord is a widely used communication platform where users exchange large amounts of information. Discord chat logs are very valuable as they often contain useful information. However, the existing access mechanisms do not provide easy and full access to these chat logs, and access to them may be withdrawn at any time. The main goal of the thesis is to design and implement a tool for creation of backups of chats and media from the Discord social platform.

- Analyse existing web archival tools and best practises.
- Analyse the current access mechanisms for the Discord platform.
- Design and implement a tool for the backup of Discord discussion servers. The tool should enable backup creation and search capabilities.
- On a selected Discord server validate the tool and perform anonymized analysis over collected data.
- Evaluate the functionalities of the tool.



**FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE**

Master's thesis

Archival tool for the Discord communications platform

Bc. David Labský

Department of applied mathematics

Supervisor: Ing. Milan Dojčinovski, Ph.D

May 4, 2022

Acknowledgements

I would like to thank my supervisor, Ing. Milan Dojčinovski, Ph.D., for taking on this topic with me, for his feedback and guidance.

Thank you, Jiří, for the mutual support during the making of this thesis and throughout our studies, we make a good team and I couldn't have done it without you.

Thank you, dear Clanga, for moral support as well as proofreading. Thanks to Tiido for proofreading as well.

Thank you to all the friends and colleagues who have been patient with me in the process of writing this thesis.

A final thank you goes to my parents for continuously encouraging and backing my studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No.121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46 (6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the “Work”), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prague on May 4, 2022

.....

Czech Technical University in Prague

Faculty of Information Technology

© 2022 David Labský. All rights reserved.

This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis

Labský, David. *Archival tool for the Discord communications platform*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2022.

Abstrakt

Discord je populární chatovací platforma která v současnosti neumožňuje uživatelům exportovat všechna data, ke kterým mají přístup. Možnost tvořit zálohy online dat je důležitá jak pro osobní důvody, tak za účelem umožnění dlouhodobého uchování dat. Cílem této práce je vytvořit open-source nástroj pro archivaci Discordových chatů ke kterým má daný uživatel přístup. Využíváme strategii zachycování síťové komunikace prováděné headless webovým prohlížečem. Tato metoda je široce aplikovatelná pro archivaci single page aplikací jiných než Discord, se kterými současné nástroje obtížně fungují. Funkcionalita je prokázána skrze provedení analýzi dat stažených ze zvoleného Discord serveru.

Klíčová slova Discord, chat logy, webová archivace, uchování digitálních dat, Puppeteer

Abstract

Discord is a popular instant messaging platform which currently does not allow its users to export all data which they can access. The ability to create backups of online data is important for personal reasons as well as to enable long term preservation. The goal of this thesis is to create an open-source tool for the archival of Discord chats to which one has gained access. We use a strategy of capturing network traffic performed by a headless web browser. This method is broadly applicable to archiving single page applications other than Discord which current tools have difficulty working with. Functionality is demonstrated by performing analysis of data downloaded from a chosen Discord server.

Keywords Discord, chat logs, web archival, digital preservation, Puppeteer

Contents

1	Introduction	1
1.1	Thesis goals	5
1.2	Thesis structure	5
2	Background and Related Work	7
2.1	Motivation	7
2.1.1	Digital preservation	8
2.1.2	Online content is at risk	9
2.1.3	Archival of online content	10
2.1.4	Difficulty archiving modern websites	12
2.2	Applications for archived chat logs	13
2.2.1	Search engines	13
2.2.2	OSINT	14
2.2.3	Machine learning	15
2.3	Data liberation	16
2.3.1	Digital Markets Act	16
2.4	Related work	17
2.4.1	Third party Discord clients	17
2.4.2	Discord Archivers	18
2.4.2.1	DiscordChatExporter	19
2.4.2.2	Discord History Tracker	21
2.4.2.3	Pullcord	22
2.4.2.4	Discard	23
2.4.3	Comparison	24

2.5	Findings	25
3	Analysis and Specification	27
3.1	Discord architecture	27
3.1.1	Users	28
3.1.2	Servers	29
3.1.3	Channels	30
3.1.4	Messages	30
3.2	Means to access Discord	32
3.3	Tool specification	33
3.3.1	Objects	33
3.4	Legality	34
3.5	Ethics	35
3.5.1	Requirements	36
4	Technologies	39
4.1	Development of Discord bots	39
4.2	Archiving web content	40
4.2.1	Formats	40
4.2.2	Software	42
4.3	Capturing network traffic	43
4.3.1	Proxy software	44
4.3.2	Packet capture software	44
4.4	Headless browsers	45
4.4.1	Selenium	46
4.4.2	Puppeteer	46
4.4.3	Playwright	46
4.5	TypeScript and Node.js	46
4.6	Linux containers	47
4.7	Elasticsearch	47
5	Implementation	49
5.1	Crawler	50
5.1.1	Browser	51
5.1.2	Jobs	51
5.1.3	Discord project	52
5.1.4	Capture tools	54

5.1.4.1	mitmproxy	54
5.1.4.2	Wireshark	54
5.2	Reader	54
5.2.1	Output formats	55
5.3	Command-line interface	57
5.4	End-to-end tests	58
5.5	Findings	59
6	Validation	61
6.1	Analysis	62
7	Conclusion	67
7.1	Future work	68
	Bibliography	69
	A Acronyms	79
	B Supplemental Material	81

List of Figures

1.1	A screenshot of the mIRC IRC client displaying a chat room[1]. mIRC logs received messages automatically.	2
1.2	A screenshot of the Discord client displaying a chat room[2].	3
2.1	Internet Archive’s Wayback Machine displaying the 1998 version of the website of the Czech Technical University in Prague.[3].	11
2.2	Screenshot of DiscordChatExporter.	20
3.1	Discord architecture from the perspective of a user.	28
3.2	A variety of sample messages.	31
4.1	Marketing screenshot of Elastic’s Kibana, demonstrating its ability to create data dashboards.	48
5.1	A diagram describing the architecture of Discard2.	50
5.2	Processing of a job in Discard2’s crawler.	51
5.3	Hierarchy of tasks in the Discord project.	53
5.4	Screenshot of Discard2 output during a crawling job.	57
5.5	Diagram describing Discard2 architecture during end-to-end tests.	58
6.1	Activity over time.	62
6.2	Heatmap of activity over the course of a week.	63
6.3	20 most active channels.	64
6.4	50 most active users.	64
6.5	Messages sent per minute (blue) against average sentiment (green).	65

6.6 Word cloud of thread names. 66

List of Tables

2.1	Overview of popular Discord archivers ¹	19
6.1	Number of results for various search strings	62

Introduction

Ever since its inception, the Internet has transformed the way humans communicate. E-mail, instant messengers, and social media have had an impact on culture worldwide. However, the landscape of online services is constantly changing, and has been a shift from open, federated solutions like e-mail to proprietary platforms backed by for-profit corporations. Instant messaging has also undergone this shift. The earliest chat protocol to achieve wide adoption, IRC, was fully decentralized, with numerous large networks. The open model resulted in a proliferation of clients suitable for various hardware and users. However, full decentralization also brings downsides. Without an authority or consensus on evolution of the protocol, changes in the computing landscape can leave a platform out of date. IRC's usability suffers in the era of mobile devices, as it is not designed for spotty broadband connection.

Discord is an instant messaging and VoIP platform introduced to the public in May 2015. Owing to its ease of use and readily available voice over IP services, it quickly garnered adoption in the e-sports and streaming audiences[4]. Although continuously favored by gamers, Discord has successfully broadened its audience and is now targeting student groups¹, hobbyists, fandoms, and more. As of 2021, Discord has over 350 million registered users, nearly half of which are active monthly[5]. Every minute, Discord users are collectively sending 668 thousand chat messages[6].

Traditionally, providers for instant messengers (such as IRC, AIM, ICQ,

¹As of April 2022, the Discord server for the Faculty of Informatics of the Czech Technical University in Prague has 3,547 members in it, primarily students.

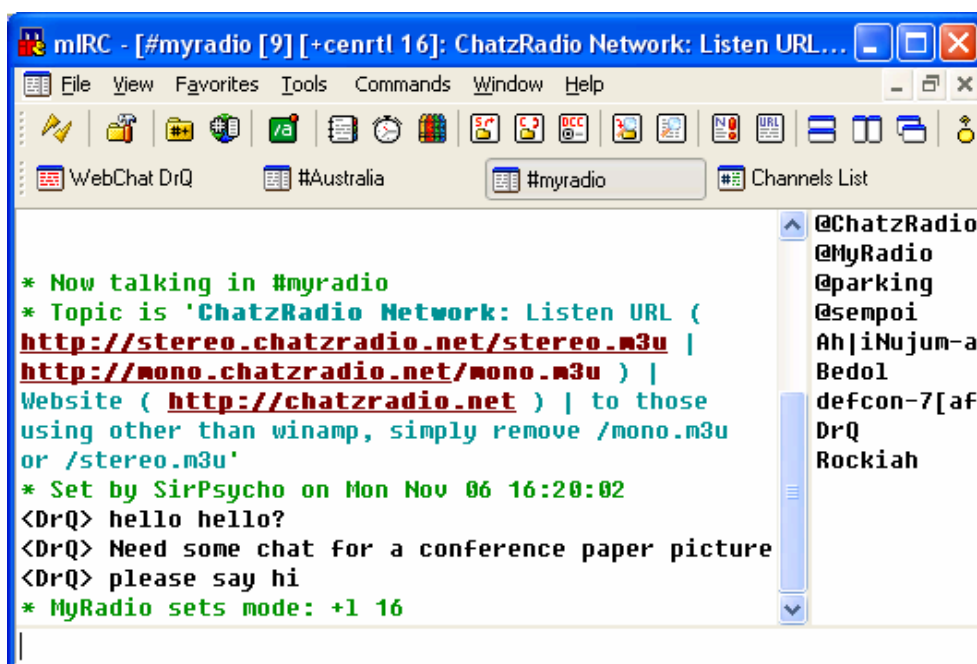


Figure 1.1: A screenshot of the mIRC IRC client displaying a chat room[1]. mIRC logs received messages automatically.

or MSN) acted only simple gateways. Messages would be passed from user to user immediately without the option to access them again. Instead, clients typically had a logging feature, keeping a local record of all personal communications involving the user. The advent of cloud services has upended this principle. Modern social platforms like Discord store all user data server-side. This provides practical utility as users no longer have to stay connected in order to read backlog (i.e. conversation that happened when they were disconnected), and it also allows the server to index chat logs for faster search. On the downside, however, this means the user is at the mercy of the service provider for their data. Should the user lose access to their account for any reason, including being banned at the discretion of the service provider, they lose access to their historical records. In fact, since users can delete messages in private chats without leaving a notice, there is a real risk of gaslighting over past conversations².

²The term *gaslighting* refers to interpersonal manipulation resulting in the victim questioning their sense of reality. For instance, a person may end up in a situation where they have to decide whether the second party had deleted a message and is lying about the fact,

The Discord client does not perform any client-side logging, and Discord forbids the use of third party clients[7], which may provide such functionality. While a competing chat platform, Telegram, offers its users a powerful chat export tool[8], Discord has no such feature. As a for-profit company, Discord may find itself lacking motivation to allow users to download complete chat logs, in part because it would ease migration to other platforms. It is hoped that the work done in this thesis will benefit communities interested in moving from Discord to open, decentralized chat platforms such as Matrix[9].

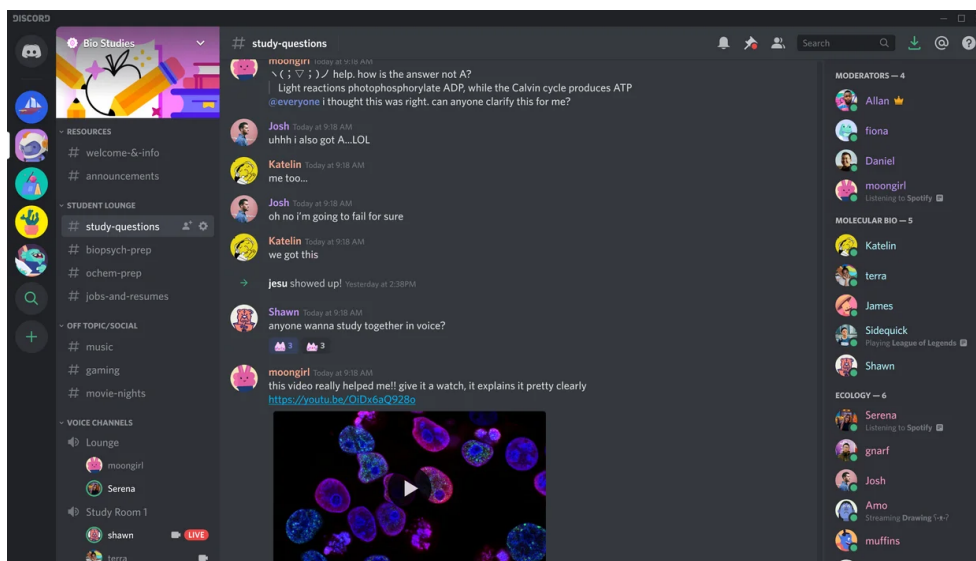


Figure 1.2: A screenshot of the Discord client displaying a chat room[2].

As closed services like Discord continue to displace traditional platforms of the open web like forums³ and personal homepages, the Internet is shifting its look. Discourse is mounting around Google search result quality deteriorating[12]. Savvy researchers know to scourge communities for a given topic on platforms such as Reddit and Discord to find answers to their questions. Invite links to many Discord servers are public, and after joining, all history is there to read, however a Discord account is required and content or if they are misremembering the past conversation.

³Although Discord is primarily a chat platform, communities which close forums often justify it by claiming Discord is simply a more popular place for people to talk, albeit this is not without pushback[10]. Additionally, at the time of writing, Discord is noted to be in the middle of developing a "forum" feature, the description of which can already be found in the API documentation[11].

1. INTRODUCTION

from the server cannot be discovered by search in the first place. Discord's status can be described as *deep web*, as chats are unable to be indexed by conventional search engines, even if the intention is for them to be public.

The ability to create backups of online data is essential for reasons of keeping a personal record, data liberation, and long term preservation. As such, the goal of this thesis is to create an open-source tool for archival of Discord chats to which one has gained access. We will utilize the strategy of capturing network traffic performed by a headless browser running in a container. This method will be broadly applicable to archiving single page applications other than Discord which current tools have difficulty working with.

1.1 Thesis goals

The goals of this thesis are:

- analyze best practices in digital archival and the existing tools for archival of web applications,
- inspect the current state of the art in the field of archiving Discord servers,
- create a tool for archival of Discord chats to which one has gained access,
- allow exporting archived data to a variety of formats for further processing, including enabling search,
- demonstrate the tool's functionality by downloading data from a chosen Discord server and performing simple analysis.

1.2 Thesis structure

This thesis is structured as follows. In Chapter 2 - **Background and Related Work**, we discuss the motivations for archiving chat logs, the background of archiving online content, tools that are currently available which partially solve this problem, and other developments in the area. In Chapter 3 - **Analysis and Specification**, we discuss the architecture of the Discord platform and come up with suitable requirements and design for the tool we're developing. In Chapter 4 - **Technologies**, we describe the technologies that were considered and used to implement the tool. In Chapter 5 - **Implementation**, the architecture and capabilities of the tool are explored in detail. Chapter 6 - **Validation** has us demonstrate the functionality of the tool by analyzing the chat logs of a chosen Discord server. Finally, in Chapter 7 - **Conclusion**, we discuss the results of the research, the implications of the tool, and future work.

Background and Related Work

In this chapter, we go over the important place chat logs hold in the modern world, discuss the motivations for archival of online data, the challenges associated with data preservation, and the myriad applications for information gathered from Discord. We also go over the existing tools that may solve this problem for us today, making note of their limitations.

2.1 Motivation

The goal of archiving is to ensure the continued availability of valuable records for long-term use, to allow for their efficient retrieval and to prevent loss.

Archiving is important because it ensures the availability of records which can tell stories about events, people, or organizations. Individuals may have practical and sentimental reasons to ensure the longevity of their personal data, and businesses just as well have motivation to prevent loss of data which may be critical. Materials properly archived can also be used as evidence during legal procedures to ensure justice. Over a period of decades and centuries, archived materials become of interest particularly to researchers and historians.

Chat logs, in contrast to written publications or grey literature, offer a window into the minutiae of life. They may contain trivial information which becomes signal over time or in larger quantities, knowledge the long term importance of which was not known at time of writing, and more often than ever before photos and attachments of value. Archived chat logs allow us

to document the birth of ideas or to study contemporary reactions to major world events, such as the September 11 terrorist attacks[13].

The Discord platform has been available since 2015[4] and has garnered hundreds of millions of users worldwide. But no online service can guarantee the continued availability of data hosted on the platform. That is why it is important to develop tools allowing users to download their data and retain it themselves.

2.1.1 Digital preservation

In case of archiving digital materials, preservation is of particular importance. Unlike physical objects such as books, which, when stored under stable conditions can last nearly forever, regular effort must be put to ensure digital objects don't become inaccessible. There are multiple layers to this risk. The term *bitrot* typically refers to the phenomenon in which digital media, such as optical disks or memory chips, loses data over time. This can be due to improper storage conditions or handling, but broadly speaking, no digital media has a lifespan compatible with long term preservation, and so the most secure and economical way to store data in the long term requires redundancy (i.e. backups) to ensure data integrity, and processes in place which regularly check and migrate the data onto new devices in cases of failure.

However, even when data integrity is ensured, it may still become more difficult to work with it in time. While data can be kept constant in time, the world changes around it. The software necessary to work with it doesn't exist in a vacuum, it's a part of a complex digital ecosystem, and systems can change or become obsolete and eventually unavailable. Common examples include image or text data stored in obsolete file formats, or expired software licenses barring access. Thus, in contrast to physical artifacts which wear out with use, it can be said that digital artifacts wear out when work is not put into maintaining them[14].

There are two main solutions to sustain the capacity to understand digital objects. The first is to continuously re-interpret the data into newer formats and onto newer platforms. The second is to ensure reproduction of the old platforms by means of emulation, for example of an older operating system which supports the relevant software. During the design phase of a new

format intended for long term preservation, both methods should be carefully considered and enabled.

2.1.2 Online content is at risk

The era of so-called cloud services has changed the manner most people interact with their data. Cloud services have their advantages: the user no longer has to think about files and data storage, and their data is available anywhere with an internet connection. That is convenient. However, when data is only available over the network, the situation concerning its availability can change at any time. When the user goes offline for any reason, the data will be unavailable to them, however, that is typically only a temporary inconvenience. Service providers carry the risk of short and prolonged outages of their own. As an example, at time of writing, an estimated 50 to 400 thousands users of Atlassian Cloud services, including the popular issue and project tracker JIRA, were impacted by an outage which barred them from accessing critical company data[15]. This outage has lasted for two weeks[16] and one researcher discovered few of the companies had any backups[15]. It is likely cloud data backup practices among individuals are worse yet than those of businesses⁴.

Going beyond outages, every year, scores of websites become unavailable and online services get shut down[18]. Product Hunt is a website reporting on new products in tech. 22 % of the websites of products featured on Product Hunt since December 2013 are no longer reachable[19]. Major companies such as Google and Yahoo are infamous for sunsetting products which are still popular with their users, in some cases mere years after acquiring them[20][21]. When notice is given, which is not universal, these services occasionally offer their users to download an export of their content. Sometimes these exports are reported to be unreliable or incomplete, such as in the case of Yahoo! Answers[21]. Even when user data export is offered, it has to be requested and downloaded, which a minority of users will do, and it contains only data authored by a given user. Most information that was previously

⁴As of 2021, according to backup provider company Backblaze, 80 % of U.S. adults who own computers make backups at least once a year[17]. This is an encouraging number, but it can be understood to concern data stored on their personal computers. Indeed, 61 % of respondents indicated their primary backup method was "the cloud". Those users are unlikely to back up data stored on the cloud in the first place. 31 % also indicated they do not understand how the cloud works.

2. BACKGROUND AND RELATED WORK

available publicly becomes difficult to locate or lost forever, contributing to a phenomenon known as *link rot*[22].

Hyperlinks are references in text pointing to other documents which can be followed with a click or a tap. Hyperlinks were integral to the creation of the World Wide Web which transformed the way people use the Internet. The pervasion of broken links which no longer point to their original intended destination, that is link rot, erases important context and hurts the Internet’s ability to preserve information. Estimates of the rate of link rot varies between studies and categories of links, but the half-life of a link is reported as anywhere between two years[23] to about 14 years in academic corpora[24]⁵.

2.1.3 Archival of online content

Prevention and remedy of link rot is a major concern for digital archivists. This typically involves the creation of backups of publicly available content which is suspected or known to be at risk or is deemed culturally significant. The **Internet Archive**[25] is the principal player in this space. It is an American digital library which provides public access to collections of digitized materials, including books, music, video, and software. As of 2022, the Internet Archive hosts over 100 petabytes of data[26]. A major endeavor developed by the Internet Archive is the **Wayback Machine**[27], which hosts captures of 681 billion web pages. Internet Archive develops and runs its own crawler infrastructure on the World Wide Web. They expose web captures through the interface of the Wayback Machine, which enables everybody to view captured historical versions of webpages, including those which have gone offline.

Journalists, academics, and regular users alike have an interest in preserving online content they care about. They may desire to reference it in the future or wish to ensure public availability. Lacking proper methods, however, they resort to crude methods like copy-pasting text or taking screenshots. These low fidelity methods are problematic, because they only capture fractions of content, lose valuable metadata, and are more difficult to verify the authenticity of. When people have the means to make higher quality captures, they do make use of them. Besides automated captures of websites

⁵Note that research may not account for *content drift*, which refers to significant changes of the linked content which is nonetheless online.



Figure 2.1: Internet Archive’s **Wayback Machine** displaying the 1998 version of the website of the Czech Technical University in Prague.[3].

by means of a spider, the Wayback Machine offers an interface called **Save Page Now** which allows anybody to make copies of web pages important to them. In a process that has been described as *participatory web archiving*[28], in 2018, 100 pages were being saved per second using the Save Page Now interface[29].

While Save Page Now is excellent at empowering users to save content online, it only allows for saving a single page at a time. When websites announce shutdowns, dedicated groups come together to help preserve the content. Volunteer groups of archivists such as **Archive Team**[30] monitor services at risk, or rely on requests from users, especially for smaller or local sites where the language barrier prevents news from reaching the group. With the help of a distributed network of machines also provided by volunteers, Archive Team has been preserving content at risk online since its founding in 2009⁶. Owing to a versatile arsenal of tools and a strategy that can be

⁶Archive Team describes itself as a “loose collective of rogue archivists, programmers, writers and loudmouths dedicated to saving our digital heritage”. The author has been an active member of the group beginning with 2015.

described as "archive first, ask questions later", Archive Team has been able to act quickly in face of tight deadlines to preserve copies of online content.

2.1.4 Difficulty archiving modern websites

It is difficult to overstate the speed at which technologies evolve and change. The stack behind the World Wide Web, developed at its inception with a basic capability for hypertext documents, serves today as a backbone for complex real-time applications. Single-page applications (SPAs) are websites which interact with the user by dynamically rewriting parts of the web page using JavaScript. The added complexity over traditional web pages proves troublesome for many tools developed for digital preservation in the past. For instance, GNU Wget, which is commonly used to recursively download websites lacks the capabilities to execute JavaScript completely. The WARC format currently standard for archives of web content has no support for capturing data transferred over the WebSocket protocol, as used by applications such as Discord.

Increasingly, headless browsers are used to support the full range of interactions possible with modern websites. The **Webrecorder** group employs this technique in a suite of open source tools they are developing to enable capture and replay of interactive websites in a high fidelity manner[31]. In cases when user data in an online service is only available behind login or through an application, specialized tools have to be developed to download it. There are always risks associated with storing personal data in such applications. Even if the product stays online, companies can withhold access to it without any justification or notice. Whether done because of a legitimate user error or due to some other cause, communication over lost accounts is often poor⁷. Among those suites which specialize in helping users retrieve personal data from online silos are **HPI** (Human Programming Interface)[35] and **Dogsheep**[36].

The tools currently available for archiving data from Discord are dis-

⁷Google is among those companies known to disable user accounts without communication or clear paths to a solution, as a result of which users lose access to their email and documents permanently, as in [32] or [33]. The Google account of a historian and military expert was locked after his research files were flagged as "terrorism-related content"[34]. As noted in Section 2.4.1 (Third party clients), Discord users have also suffered from data loss resulting from account suspensions.

cussed in Section 2.4.2.

2.2 Applications for archived chat logs

Inter-human communication is a keystone to intelligence and much of the information passed this way is never processed into formal literature like articles, white papers, and other documents. In the era of instant communications, companies like **Slack**⁸, which provides a chat platform geared towards businesses and workplaces, encourages team members to treat chat logs as a knowledge base and make liberal use of the search feature[38]. The skills and tools to work with chat history are critical for research today.

In this section, we briefly cover some of the fields where a tool to archive Discord chat logs is applicable and useful.

2.2.1 Search engines

The Internet is a vast resource and the amount of information available is beyond comprehension. Traditionally, search engines have served an important role in assisting navigation of the Web by providing automatic curation and surfacing results relevant to a query. **Google Search**, by far the most popular search engine, has become so entrenched in daily lives that "to google something" functions as a readily understood verb. Nevertheless, in 2022 numerous sources observed that Google's search quality has been deteriorating[12].

The problem is partly caused by the prominence Google has started giving to advertisements, its primary source of revenue[39]. Not only have ads become less clearly marked over time, for some search queries, sponsored results cover the whole screen[40]. However, even the quality of organic results has perceptively been on the decline. Good understanding of search engine optimization (SEO) techniques has enabled the practice of *spamdexing*[41], which pushes artificial, low quality content (often known as *blogspam*) to the top of the search results.

When searching on the Internet, people often seek the opinions of other people, not commercialized content. It was noted that many people have

⁸Underscoring this point is the origin of the name Slack, it being an acronym for "Searchable Log of All Conversation and Knowledge"[37].

started suffixing "reddit" to their Google search queries to find authentic discussions from real people and enthusiasts as opposed to paid reviews[42]. Reddit is a popular social news aggregation and discussion website[43]. Keen to capitalize on this trend, on 2022-04-14, Reddit has introduced a new search engine for comments on their website[44].

To researchers looking for genuine conversations on a given subject, **Discord** is another avenue. With over one million publicly listed Discord *servers*[45], many of which are centered about various technological, educational, and hobby topics, it is a treasure trove of information. However, searching through Discord servers requires the user to join them, even when invite links are public. Moreover, the Discord client allows for searching in only a single Discord server at a time. Tools which enable users to create backups of Discord servers also empower users to employ local tools to search across many at a time and enable integration into personal search engines.

2.2.2 OSINT

Open-source intelligence (OSINT) is broadly defined as the collection and analysis of data gathered from publicly available sources in order to produce actionable intelligence. OSINT has applications in national security, law enforcement, business intelligence, as well as citizen science. The Internet is a vast resource of user submitted data on a variety of topics in real time. Discussion fora, social media, and indeed chat platforms can produce a valuable insight into local events. Such information can help keep the public informed about current events and keep governments and other groups accountable when they release information which can be shown to be false. Groups such as **Project OWL**[46] which perform OSINT covering geopolitical events organize themselves on Discord and similar platforms, with the Disboard server aggregator listing 19[47].

Open source intelligence has recently risen into prominence in the wake of the war in Ukraine. Given a population with smartphones in hand and access to the Internet, social media has become awash with images and videos of military vehicles on the move, missile strikes, and interactions of citizens with occupiers of towns and cities. At the same time, we are not yet at the point where it would be easy to routinely create forged material (*deepfakes*) that skilled people cannot tell apart from reality. Thus, unclassified citizens

are able to collect data from a variety of resources to derive information which helps clear the fog of war. A few examples include the use of satellite imagery to geolocate recordings of military hardware posted on TikTok or Twitter, or the use of Google Maps' crowdsourced traffic information to determine the location of military convoys[48].

One prominent OSINT and forensic analysis organization, **Bellingcat**, demonstrates how the online media environment regarding the war is volatile: content can be deleted by users and channels can be removed by platform moderators. They encourage users to archive communication platforms such as Telegram[49].

Information gathered from public sources such as open Ukrainian and Russian Discord servers, particularly those discussing the current situation, may affect the course of the war. It is possible that nation states and other major actors already have the capabilities to gather data from Discord and other platforms (such as Twitter or Instagram) at scale. Democratizing this ability can put power in the hands of people as well as remind individuals about the possible impact of the information they post publicly.

2.2.3 Machine learning

The rise in machine learning and artificial intelligence in the past few years has been staggering. With new techniques, progress has been made particularly in the areas of language comprehension and image manipulation.

OpenAI's GPT-3 language model from 2020[50] is one of the largest models of its type. The quality of text produced by this model is so good, it is difficult to distinguish it from text written by humans[51]. Among a myriad uses, GPT-3 has been used to produce various kinds of chatbots[52]. The full version of GPT-3 constitutes 175 machine learning parameters and of particular interest to us is the corpus of text used to train the model. Although a part of this training data was sourced from books and Wikipedia, the majority of it – over 429 billion tokens, 86 % of total tokens – came from sources on the open Internet (the *Common Crawl* and *WebText2* datasets)[50]. The WebText series of datasets are comprised of URLs harvested from Reddit submissions with a minimum score of 3 as a proxy of quality[53].

It is clear that for continued growth of these generative language models and artificial intelligence at wide, larger datasets will be needed. Continuing to harvest texts available on the open web will eventually yield diminishing returns, and chat logs such as those found in large, public Discord servers are one possible avenue for future datasets of human text particularly suitable for training chatbots.

2.3 Data liberation

Data liberation is the principle that users have the right to download and migrate their data from online services[54]. It's aimed at reducing vendor lock-in and protecting user freedoms. Some companies independently take measures to enable data liberation, for instance, **Google Takeout** enables users to download a copy of their data hosted on Google services. Regulation in the area of data liberation has been slowly arising, with the European Union spearheading this space with the infamous General Data Protection Regulation, or **GDPR**[55]. The GDPR ensures that individuals have control and right over their personal data held by companies. In particular, Article 15, the right of access, gives people the right to learn how their data is processed and request a copy of the actual data. Article 20, the right to data portability, ensures that the data is provided in a "structured, commonly used and machine-readable format".

The GDPR is a powerful instrument, and Discord offers a GDPR export. However, as further elaborated on in Section 3.2, Discord's GDPR export only includes messages sent by the user (arguably the "personal data"). Fragmented, one-sided records of conversations are of little use for archival. When companies don't offer useful and complete exports of data, independent users will develop custom archiving and "downloader" solutions.

2.3.1 Digital Markets Act

On 2022-03-24, The European Union Parliament and Council has agreed to a new set of rules regarding so-called *tech gatekeepers*, the **Digital Markets Act**[56]. This regulation, the text of which is at time of writing only provisionally agreed, is set to force the largest messaging services, such as Whatsapp, Facebook Messenger, or iMessage, to document their APIs and open them to

the public in order to enable alternative clients and bridges to connect.

The definition of a *tech gatekeeper* according to the DMA is a tech company with a market capitalization of at least 75 billion euro or an annual turnover of 7.5 billion. The company must also provide services such as browsers, messengers, or social media, with at least 45 million monthly end users in the EU. In 2020, Discord was valued at only 3.5 billion U.S. dollars[57], and as such falls outside of the scope of this regulation. Nonetheless, this act shows how the dynamics of instant messengers is changing, and there will be additional pressure towards open messaging.

The DMA does not specify any particular standard the messengers should adopt, only that they should document and publish their current APIs. At first, it is expected that messengers will implement interoperability with other providers using dedicated bridges for each service. In the future, the DMA may expand to include group chats in its scope. At that point, it may be effective for service providers to agree on an open standard like Matrix or XMPP to enable interoperability[58]. In addition, the DMA is unlikely to force service providers to allow communities to migrate their chat history, only allow new messages to be bridged over. As such, a tool to enable migration of data from Discord to the Matrix protocol would be especially useful for communities making use of such a bridge.

2.4 Related work

As a large social platform, Discord has attracted a lot of attention from tech savvy users. These users have developed frameworks, bots, and even third party clients. Some of these which are relevant to the topic of archival are described in this section.

2.4.1 Third party Discord clients

Third party clients are programs which are not officially developed and sanctioned by the developer of a platform. They provide a custom interface to the platform, and may provide additional features and plug-ins for user convenience. Typically, they rely on a re-implementation of the platform's protocol and API achieved by reverse-engineering. This is similar to how an archiver may be implemented. A common motivation for using a third party Discord

client is performance. The Discord desktop application is implemented in Electron[59], and has been noted to be heavy on system resources like CPU and RAM. Users with weaker computers who wish to communicate on Discord may want to use a client which is native and thus lighter on resources.

Discord has publicly stated in a tweet that the use of third party clients or *client modifiers* is against their Terms of Service and can result in account termination[7]. However, neither the Terms of Service at time of posting, nor the version at times of writing state this explicitly; the closest come the provisions against reverse engineering and modifying Discord’s software[60], which are difficult to enforce.

Users of third party clients such as **Ripcord**[61] and **cordless**[62] have reported their accounts being suspended for violating the Discord Terms of Service[63][64]. In response to one of the cases, a Discord developer has stated that they are not specifically banning people for using third party clients, but rather accounts which use their API in ways the official client doesn’t as a part of their spam detection strategy[65]. Albeit this particular user got unbanned after multiple appeals, the use of third party clients is going to continue being a risky proposition, since it can be difficult for third party clients to keep up with the changes made to the protocol and act in a manner that doesn’t trigger spam protection.

Rather than re-implement the Discord API in order to fetch data from Discord, the tool developed in this thesis will make use of the official client, which should provide correct behavior and minimize the risks of account suspension.

2.4.2 Discord Archivers

The idea of exporting, archiving, or otherwise downloading data form Discord is not particularly novel. Users of any platform which provides limited ability to export data will seek to implement the ability to do so themselves. To accomplish this task, users can make use of one of the several open source tools available today. In this section, we examine their implementation and output formats.

In addition to these broadly used tools, the author of this work has

Name	Int.	Upd.	Stars	License	Language
DiscordChatExporter	2017	2022	3,624	GPLv3	C#
Discord History Tracker	2016	2022	277	MIT	C#, more
pullcord	2017	2021	45	Unlicence	Go

Table 2.1: Overview of popular Discord archivers¹

previously attempted to create a Discord archiver called *Discard*. A section will be dedicated to its design and lessons learned.

2.4.2.1 DiscordChatExporter

With 3.6k stars on its GitHub repository **DiscordChatExporter**[66] may be the most popular Discord archiver in use. It has been under active development since 2017.

DiscordChatExporter provides both a command line interface and a straightforward GUI (only available for Windows). After providing a bot or a user token, the user can choose which channels and servers they wish to export. When given a user token, the tool also supports exporting direct messages with other users. Message logs can be output messages in one of the following formats: HTML, TXT, CSV, JSON.

DiscordChatExporter is written in C# and re-implements the Discord API from scratch. As such, using it with a user token – which, while discouraged, can be the only way to download most chats – use carries a similar risk to using third party clients in that behavior which deviates from the official client’s behavior can result in account suspension. Reports of people getting banned for use of the tool are scant but not nonexistent[67].

DiscordChatExporter is an excellent tool for end users, owing to its GUI and easy to view HTML export format. However, from the position of a digital archivist, it is difficult to accept some of the choices made resulting in a lower fidelity capture. While most of the Discord API uses the JSON format, the JSON files exported by DiscordChatExporter are processed and differ in capitalization and wording. For instance, in message objects, the official Discord API uses `edited_timestamp` and `pinned` field names, while

¹*Int.* stands for year introduced. *Upd.* stands for year last updated. *Stars* refers to the number of stars on the project’s GitHub repository.

2. BACKGROUND AND RELATED WORK

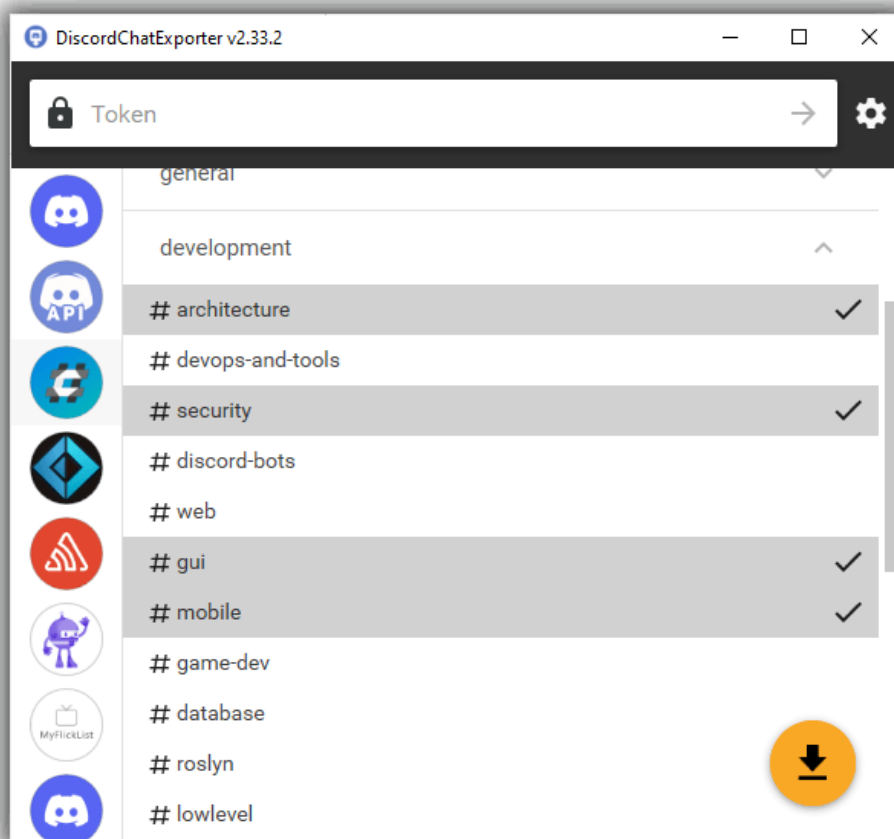


Figure 2.2: Screenshot of DiscordChatExporter.

DiscordChatExporter opts for `timestampEdited` and `isPinned`. This may seem trivial, but it hampers efforts to work with this data and ultimately the longevity of the format. Unfortunately, the author of the tool has stated this is a design choice that won't be fixed[68].

Recording API responses with minimal data processing allows for certainty that no data is missed, even for exotic types of content, or in case Discord changes its API. For instance, when Discord introduced the *reply* feature[69], DiscordChatExporter did not include the data in its exports for a month until the author of this work implemented it[70]. All data exported during this period has thus undergone *silent corruption* possibly affecting the semantics of the conversations when later interpreted. Saving raw message

data as received from the server would prevent cases such as this. Unfortunately, a pull request to add this functionality to `DiscordChatExporter` was rejected[71].

Even though the author of this work considers `DiscordChatExporter` to be a fine tool and respects the continued work put in by its maintainer, the previous issue, combined with other omissions marked as out of scope including the ability to fetch the list of users in a chat[72] or the authors of reactions to messages[73] motivate the creation of a more thorough archival tool designed around the idea of raw packet capture.

2.4.2.2 **Discord History Tracker**

Discord History Tracker[74] is another popular open-source Discord archiver. First released in in 2017, it was originally implemented as a bookmarklet, running in the browser. Beginning with 2021, a desktop app companion (implemented in `C#`) is also available.

As a browser script, `Discord History Tracker` interacts directly with the official Discord client. It listens to changes to the UI and when a new message is received, it transmits it to the desktop app, which saves it in an `SQLite` database. In this way, it can record incoming messages in real-time as well as scroll through the chat history of a channel in order to fetch all messages. `Discord History Tracker` also comes with a GUI for reading archived messages in the database.

Technologically, building the archiver in part as a browser script is a very interesting choice. It subverts the need to reimplement the Discord API, which, as determined earlier, risks account suspension. Instead, the tool scrolls the chat and clicks on buttons in a manner similar to a human. Although unlikely to be of consequence, it should be noted that Discord could easily detect the presence of this script.

Since `Discord History Tracker` reads message data from the DOM of the browser, it is not able to reconstruct the original JSON object as transmitted over the network. While arguably a nitpick, this method makes it unfeasible to archive data from certain exotic types of messages. For instance, `Discord History Tracker` does not support archiving messages which contain buttons, and

the author of the program has admitted that *"Discord has so many features now that I don't have the time to figure out how everything works, faithfully replicate it, and fix it every time they change something."*[75]. Both parsing the DOM and updating the database schema to keep up with the changes is a complicated undertaking. Storing raw data received from the server during archival helps preclude these situations, since extraction can always be implemented in the future as long as the data is there.

2.4.2.3 Pullcord

Pullcord[76] is a Discord archiver written in Go. At its core, it uses the **discordgo** library. It's a command line application.

Pullcord can fetch channel history, server history, and all related files including attachments, user avatars, or emoji. Pullcord's output format is a directory of TSV (*tab separated values*) files, one for each channel. It functions in an append-only mode where newer messages are stored following the previous ones. Although this is a custom, non-standard format, it is well specified in the `FORMAT.md` file of the program's repository.

There are some advantages and limitations to this format. The advantage is that the format is relatively simple and compact. The disadvantage is that as more fields or object variants get introduced, the fields have to become more complicated with amendments to the end to avoid losing backwards compatibility. It suffers from the same problem as other archivers in that processing received objects can result in lost or inadvertently mangled data.

Pullcord also currently downloads file attachments at time of processing, which may be undesirable because attachments can have large file sizes, which can slow down the download and put pressure on disk space. In the case of failure of downloading an attachment, the entire fetch is stopped with a fatal error. It may be desirable to leave attachments downloads for separate processing, as they can be downloaded from Discord servers without user authentication. A "light mode" along with the possibility to make certain kinds of errors non-fatal are planned[77][78].

At time of writing, Pullcord was last updated on 2021-02-13. Pullcord

is functional, but may no longer under active development. It does not support newer Discord features like threads[79].

2.4.2.4 Discard

Discard is the author’s first attempt at creating an archival tool for Discord. Developed in early 2021, it is written in Python and works as a command line application as well as a library. It is released under the MIT license and available on GitHub[80].

Discard supports archiving the chat history from channels and entire servers in given text ranges. It is compatible with both bot tokens and user tokens.

The idea behind Discard was to use the **Discord.py** library to communicate with Discord servers as a bot. The Discord.py library is popular among bot developers⁹, it has been in development since 2015, it has 10.3 thousands stars on GitHub[82] and excellent developer documentation[83]. As opposed to working with the data fetched using Discord.py directly, Discard monkeypatched methods performing HTTP and WebSocket requests in order to capture and log the raw traffic with the Discord server. This was meant to avoid the pitfalls that come with processing the data which other archivers encounter.

The term *selfbotting*, also called *userbotting*, refers to the practice of using the Discord API in a manner that is intended for bots, but with a user account. Since sanctioned Discord bots only have access to those servers administrators add them to, in order to access the messages in servers the user can access, but does not have the permission to add bots to, archivers have to make use of the selfbotting practice, which is prohibited according to Discord’s support[84], albeit the Terms of Service are not clear on this¹⁰.

Following the release of Discard (but independently of it), the discord.py developers have announced that support for selfbots will be removed from the discord.py library[85]. This is an understandable move, since Dis-

⁹According to the developer, it is the second most popular library in the ecosystem[81].

¹⁰The closest comes the provision against "auto-messaging" other users on the platform. However, archivers operate in a read-only mode and have no intention to message other people at all.

cord provides no support for selfbots whatsoever, and users of them risk a ban from the service. Although users have since created a fork of the library called `discord.py-self` which restores support for selfbots[86], it was not immediately evident whether use of the `discord.py` library is sustainable for the development of an archiver.

In August 2021, the core maintainer of the `discord.py` library announced that he is stepping down as the maintainer of the project[87]. Among the reasons to end his involvement in the project, he cited the Discord developers' neglect of the official bot API and overall poor communication between Discord employees and the developers of popular bots and libraries, who are volunteers. A particular point of contention was the introduction of a "privileged intents" feature, which limits the actions a bot can perform, including reading message contents, once it's present in more than 75 servers[88]. Motivated by prevention of abuse on the platform, Discord required developers of popular bots to provide verification with a government issued ID. This privacy-invasive verification system proved unpopular among bot developers. Although the `discord.py` developer announced a return to development six months after their departure (in March 2022)[81], they stated that their concerns over the lack of communication, as well as uncertain evolution of the API remain.

The fact that the bot API looks tumultuous and uncertain has contributed to the author's conclusion that building an archiver which interfaces the official client in a manner that a human does may be more future-proof.

2.4.3 Comparison

We have taken a look at four different Discord archiver tools. What follows is a comparison of the strategy each one uses to access Discord and process data.

- **DiscordChatExporter** - Interfaces with the API directly, exports processed data into a variety of formats.
- **Discord History Tracker** - Interfaces with the client using a browser script, exports processed data into SQLite.

- **Pullcord** - Interfaces with the API using a library, exports processed data into a custom format.
- **Discard** - Interfaces with the API using a library, exports raw data into a custom format.

The primary limitation of interfacing with the API directly is the risk of account suspension due to improper handling. Libraries exist primarily to enable bot development and support for *selfbotting* is waning. In addition, processing data at capture time can result in incomplete archives, especially given the fast pace Discord adds new features at. A new tool which manipulates the official client while recording raw interactions with the server may provide a solution for long term archival of data on Discord.

2.5 Findings

Digital archival is an important topic, yet many obstacles stand in the way of long term preservation of data. As online services grow more complex, so too it becomes more difficult to extract data from online silos, although legislation in regions such as the EU is on the right track to enable interoperability. Discord's stance on third party clients interacting with its service is ambiguous at best, and improper usage of APIs can result in account termination. While tools exist to download data from Discord, they have limitations and do not reach the level of fidelity of archival the author would like to see.

Analysis and Specification

In this section, we take a look at specifying the problem the design of the new Discord archiver tool. We start by going over the architecture of the Discord platform and the resources available on it. Next, we proceed by defining the desired characteristics of the archiver tool. We consider issues such as longevity of the chosen format, resilience of the program against network issues, scalability, et cetera. The non-technical requirements of legality and ethics are also considered.

3.1 Discord architecture

On the surface, the Discord platform is not very complicated. The majority of interaction happens within the context of the relation between a user of the platform and a three-level structure. A Discord server (also called internally a *guild*) is a collection of channels and users present in it. Channels within a server are set up by a server's moderators and represent chat rooms centered around a given topic. Textual channels consist of a stream of messages by users who are present in them.

Under a section of the application called Direct messages, users can also create one-on-one chats with other users. Here, it is also possible to create "group DMs", or group chats, with up to 10 members, outside of the server structure.

On the technical side, every object on the Discord platform can be uniquely identified with a 64-bit ID containing a timestamp in Snowflake for-

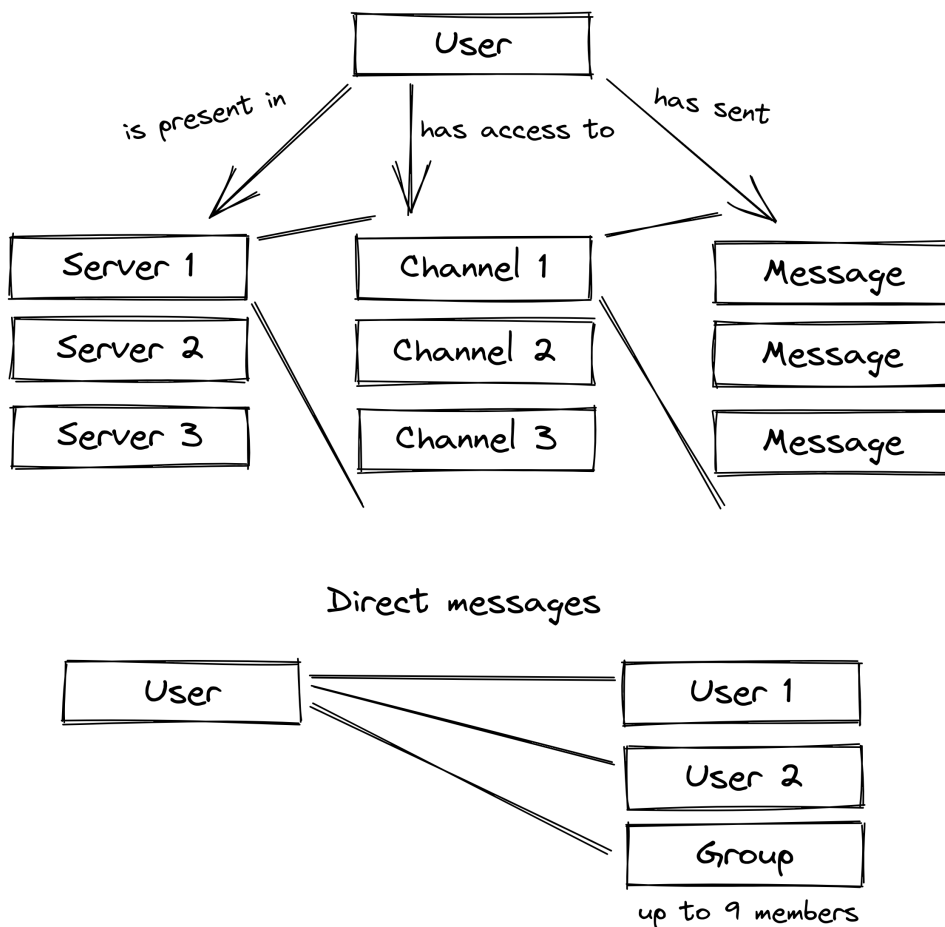


Figure 3.1: Discord architecture from the perspective of a user.

mat pioneered by Twitter[89]. When a user enables "Developer Mode" in the settings of the Discord client, it becomes possible to right-click objects such as channels and servers and copy their IDs.

3.1.1 Users

User accounts on Discord are registered using an e-mail or a phone number as a login method along with a password. The general user identifiers are in the Username#1234 format, where 1234 is a discriminator, also called a tag, a four-digit number present to make differentiate users with the same username. Usernames can be changed at any time. Users also have an avatar and can

provide a short public bio for their profile.

Bots on the platform are also deemed as users, however, messages delivered using webhooks do not have a user associated with them.

3.1.2 Servers

Discord enables any user to create a **server** (or *guild*). Typically, they are communities centered around a given topic and come in sizes ranging from a few members (such as small friend groups) to 100s of thousands (in large public servers). Servers have a name, an icon image, and an optional description. Within a server, administrators can create text and voice channels where members then interact.

Server admins can create **roles**, which are a set of permissions that can be assigned to users. Typically, roles are used to control which subset of channels a given group of users can view or write in. Roles can also serve a cosmetic or informational purpose, as they can change name colors and are visible on a given user's server profile.

Servers can have up to 100 **custom emoji**, small icons which can be used in messages and reactions on the server, or in any server if the user has the paid Discord Nitro upgrade. Similarly, it is possible to configure up to 60 **stickers**, which are larger images that can be attached to messages.

Depending on permissions, the admins of a Discord server or regular users can create **invite links**, which are short links which enable their recipients to join the server. Invite links can be set to expire after a certain amount of time, or to have a limited number of uses. Another method to join Discord servers is through the "Discover" tab, where public servers are listed and can be browsed. Discord allows users to join up to 100 servers at a time, unless they purchase the Discord Nitro upgrade to their account, in which case the number grows to 200. Server moderators can kick or ban users from the server at any time.

Discord offers a **search** feature, which can be used to find messages fulfilling specified criteria in the scope of a single server. It is possible to filter by user or channel, and clamp the results within a given date interval. The text search is fulltext and stems words.

3.1.3 Channels

Discord offers two types of channels: text channels and voice channels. Text channels behave like a chat room in which users may send messages and exchange files. Voice channels resemble a conference call which can be joined and left at any time. Voice channels also offer video and screen sharing capabilities.

A user who joins a Discord server has access to full history of the channels they can see, except for channels with the seldom denied "Read Message History" permission¹¹.

Moderators can *pin* any number of messages in a channel to bring attention to them and make them easier to access.

When a channel is open, the user can see a list of users who are also present in it. This list sorts users who are online first and can be grouped by roles. The list is lazy-loaded, meaning that in order to fetch all users in a channel, it's necessary to scroll through the list until the end is reached. In channels with more than 1000 members, those who are offline won't be shown.

3.1.4 Messages


Messages, sent by users or bots, appear in channels and private chats. Messages can be made in response to a previous message (as a reply), in which case they contain a reference.


Message text is formatted in a Markdown-like syntax and can contain mentions of other users, roles, and channels. It can also contain custom emoji. Attached to messages can be images and files up to 8 MB (100 MB with Nitro), GIFs, and stickers.

Messages can have a number of reactions, which are default or custom server emoji chosen by other users in response to the message. The list of users who responded with a certain emoji is made available in a modal dialog.


Messages, including those made in the past, can be edited by the author

¹¹A user lacking this permission for a given channel can only see the messages that were received during the current client session. Many users have noted that the feature is essentially broken, as it e.g. prevents mobile users from holding a conversation while switching applications[90].

 **test_ahcae** 20/03/2022
test 0
emoji 😊 🦎 🖌️ ❤️
image attachment







edited msg (edited) (edited)
formatting: **bold** *italics* ~~crossout~~ `code`

 **test_ahcae** 20/03/2022
thread from this msg

thread name [4 Messages >](#)
There are no recent messages in this thread.

29 March 2022

 **test_ahcae** 29/03/2022
msg
this message has 3 reactions
 1  1  1
msg2


 **test_ahcae** 29/03/2022
this message is made in response to a previous message

Figure 3.2: A variety of sample messages.

at any time. The date and time a message was edited last can be seen. Messages can be deleted by their authors or server moderators at any time without leaving a notice.

Messages can indicate the start of a **thread**. Threads as a feature were introduced in July 2021 and they allow users to branch off a conversation into an isolated channel. All threads started in a channel can be listed.

When a channel is opened, the client loads 50 messages at a time. More can be reached by simply scrolling the chat log.

3.2 Means to access Discord

Discord offers the following sanctioned methods to access data on the platform.

Web browser. The Discord client can be accessed using modern web browsers such as Chrome or Firefox.

Desktop application. Discord offers a desktop application for Windows, Mac, and Linux. The application behaves largely like the browser client, but offers better integration with the operating system. The desktop application is implemented using Electron[59].

Mobile app. Discord offers a mobile application for iOS and Android available to download on their respective app stores. It offers similar functionality to the desktop application, but is designed to be used on mobile devices. The iOS version is implemented using React Native, while the Android port is done using an in-house solution[91].

Bot API. Discord allows users to create bot accounts on the Discord Developer Portal. With a bot token, a script can interface with the HTTP Discord API in a standard and documented manner. Discord enforces some restrictions on bots, notably the ability to read messages when in more than 100 servers without additional verification. While there are bots which implement message archival functionality, and tools such as DiscordChatExporter can make use of bot tokens to work, since only administrators can add Discord bots to a server, it is not a viable option for most people to get a copy of their messages. Additionally, a bot cannot be added or granted access to private

messages.

GDPR export. As required by law in the European Union, Discord offers its users to export personal data which Discord stores about them. This export can be requested once a month and comes as a .zip package on the requester's email within days. This export includes information on the user account, the names of servers joined, user activity (including details about sign-in times and program navigation), and the contents of all messages sent by the user, as long as they have not been deleted[92]. While certainly better than nothing, a one-sided record of conversations is of limited use for archivists.

3.3 Tool specification

To fulfil the functionality of an archiver, the program will provide a basic user interface. In order to access Discord, a user account is needed, so the archiver will require one to be provided. Using provided credentials, a specified task (such as archiving a single channel) will be performed by the program. While working, the program should continuously output progress information to the user. Once finished, the program should save all recorded information to a directory. The data should either be in a human-readable format, or a method must be provided to transform it.

3.3.1 Objects

In order to specify the functionality of the archiver, we have to first define the objects of interest that we want to download from the Discord platform.

We focus on the following objects: users, DMs, servers, channels, threads, and messages.

For given objects, these are the contexts we encounter them in and the attributes we are interested in capturing:

Users. We want to know which user has sent a given message. For a given user, we want to know their name and avatar.

Servers. We want to be able to choose a single server to archive or multiple. For the archived servers, we want to save the name, icon, and list

of channels. We want to fetch a list of emoji the server offers.

DMs, Channels, and Threads. For a given channel, we want to save the topic and messages sent within a specified timeframe.

Messages. We want to capture messages in their entirety, including the text, the date and time, the author, and the reactions. We want to be able to download images and attachments.

3.4 Legality

Although not a lawyer, the author would like to share their understanding of the legal aspect of this work. Since Discord Inc. as a company is founded in the United States, attention has been focused on the legal aspects there.

While violating the Terms of Service of a service is not generally considered a crime[93], it can have undesired consequences, like account termination. Albeit the legal status of bypassing technological barriers imposed by a website intended to enforce the terms of service, such as IP address blocking, may appear contested, the author concedes with the **Electronic Frontier Foundation**, which has stated[94]:

There’s nothing inherently wrong or unlawful about avoiding IP address blocking, and there are valid reasons why someone might choose to do so, including to sidestep anticompetitive behavior by other Internet services. As long as an end user is authorized to access a computer and the way she chooses doesn’t cause harm, she should be able to access the computer any way she likes without committing a crime.

Software intended to download content from online services has been successfully defended multiple times. `youtube-dl` is a command line program which enables users to download videos from the YouTube platform. In 2020, GitHub, which provides hosting for the repository where developers collaborate as well as release downloads, received a DMCA letter requesting a takedown of the repository. The DMCA notice made the claim of anticircumvention: that the code was designed to circumvent technical measures which

control access or copying of copyrighted material, in violation of Section 1201 of the DMCA. GitHub initially complied with this request, but later deemed it unwarranted and reinstated it, noting that[95]:

As we explained, the key claim in the youtube-dl takedown is circumvention. Although we did initially take the project down, we understand that just because code can be used to access copyrighted works doesn't mean it can't also be used to access works in non-infringing ways. We also understood that this project's code has many legitimate purposes, including changing playback speeds for accessibility, preserving evidence in the fight for human rights, aiding journalists in fact-checking, and downloading Creative Commons-licensed or public domain videos.

Despite emulating the behavior of a browser in order to receive video data from YouTube's servers, `youtube-dl` was shown to not "circumvent" an access control as was claimed in the takedown request. Similar purposes and circumstances apply in the case of downloading content from Discord.

Most recently, on 2022-04-18, a landmark ruling by the U.S. Ninth Circuit of Appeals involving LinkedIn has shown that scraping publicly accessible data is legal in the U.S.[96].

Please note that while creating a personal backup of data available online is broadly admissible, much content can be considered creative work and fall under the realm of copyright law. Reproduction and distribution may be restricted. The implications of copyright in the digital age are a heavily discussed matter and beyond the scope of this thesis.

3.5 Ethics

Ethics are an often overlooked aspect of the software development discipline. Like any tool which can be used to gather data, the user can choose to use it for good or to do harm. The author would like to stress that the software developed in this thesis is intended as an archival tool. The software, or any data gathered from it, should not be used for unethical purposes, including but not limited to selling data to third parties, or researching and publicly

broadcasting private or identifying personal information (commonly called "doxxing").

Occasionally, archivists may find themselves in a dilemma with regard to privacy. Respecting the privacy of individuals while curating a public archive can be difficult. This is not a new problem: archivists have been working with materials concerning personal conversations such as letters, tapes, or transcripts for centuries. There are legal and ethical requirements which can often be met with legal aid as well as, in cases of research organizations, pre-existing sets of ethical guidelines[97]. While sometimes there is no right answer as to what should be publicly reproduced, ultimately, since collections can stay private, privacy does not have to stand in the way of preservation.

In the Discord **Privacy policy**[98], distinction is made between private and public servers. The author would like to encourage users of any archiver software to take this into consideration when working with data collected using archival tools.

3.5.1 Requirements

Based on the previous analysis, we specify **functional requirements** for the application as follows:

- Log in to a Discord account using provided credentials.
- Perform tasks according to user choice.
 - Download direct messages received from other users.
 - Download messages from a channel posted within a given time-frame.
 - Iterate over all channels available in a server.
 - Iterate over all servers available in a user's account.
- Show a basic summary in output (i.e. number of messages recorded).
- Take a screenshot on error.

The **non-functional requirements** are:

- Capture novel message types without changes to program code (forward compatibility).
- Resilience to network failure - the possibility to resume a failed job.
- Operate headlessly (i.e. without a desktop environment and human interaction).
- Output in JSON format.
- Export into a database capable of fulltext search.

The program will **not support**:

- Running continuously in order to archive messages arriving in real time.
- Recording or otherwise capturing information from voice channels.
- Joining servers without user intervention.
- Sending messages, using reactions, or otherwise interacting with other users.

A real-time mode may be considered in the future. Voice channels are widely understood to be ephemeral, and consent should be received for recording other people. Easy to use bots exist which can record voice channels[99], alternatively, system audio and video can be recorded using desktop tools such as OBS. Performing any user interactions is undesirable behavior for an archiver and automating messaging is explicitly forbidden by the Discord Terms of Service.

Technologies

In this section, we discuss the techniques and technologies that were considered for the project.

4.1 Development of Discord bots

Discord offers a bot API which can be used directly or via a library. There are more than ten libraries for creating Discord bots available for various languages[100].

The option to pick up a Discord bot development framework like `Discord.py` or `Discord.js` to interface with the Discord API in a standard way is tempting. However, Discord bots can only access messages in servers they have been added to by server admins. This requirement makes bots too limited to be useful to most archivists, who need to download messages they can access in any server they have joined.

Albeit some of the bot libraries allow for use of user tokens, they provide no guarantees that the interactions with the Discord server will adhere to official client behavior. Running a *selfbot* in this way is forbidden by Discord and can result in account termination.

Nonetheless, since bot libraries provide constants and deserialization methods for objects used in the Discord API, they may prove useful for working with data downloaded from Discord, even if they are not used for network interactions.

4.2 Archiving web content

There are many methods to capture data from websites. A widely recognized method is called *web scraping*. Web scraping is the process of using a script to repeatedly request and extract information from websites. Since Discord is a web application, many of the techniques used in web scraping are applicable. However, most approaches to web scraping yield low fidelity output, as the goal is not preservation, but gathering of specific information. Processing and reducing data during capture is a recipe for losing data which is not expected to be necessary. We look at techniques which can save as much data and metadata as possible.

The techniques employed in web scraping vary considerably depending on the complexity of the website. We first go over the **formats** suitable for long term archival of websites, and then over the **programs** which can produce them.

4.2.1 Formats

A single webpage can be thought of as an **HTML** file together with its linked dependencies: typically **CSS** stylesheets, possibly **JavaScript** scripts, images, fonts, and other. For many purposes, a copy of these files may be deemed sufficient. The files may be packaged together using a format like **ZIP** or **tar.gz** to ease transfer and reduce file size. The **HTMLZ** file format takes this approach one step further by being a set of HTML files compressed with ZIP. Many browsers and command line tools allow users to save webpages directly in these formats.

While commonly done, there are multiple downsides to this technique. Since files in this case get stored to the filesystem, it must be stressed that URLs cannot be mapped to filenames directly. URLs inside the page's files must be adjusted to point to the files located relative for the files to render correctly in browsers. In archival, altering the file that is being preserved should already be a big red flag. Dependencies that link to other domains have to be collapsed, losing important information about the source of files. Any characters in filenames that are not supported by the filesystem have to be mangled, and URLs with different cases, although uncommon, will cause problems on case-insensitive systems like Windows.

In general, **dynamic websites** which make use of query parameters (such as `?page=2`) are unsuitable for this type of archival. Importantly, this technique completely disregards the context in which the files are served in. HTML files are not isolated, they are most commonly served over a protocol called **HTTP**. In HTTP, the **request** made by the client contains a lot of important metadata: the version of the HTTP protocol used, the URL requested, the name of the client (or User-Agent) that made the request, **cookies** which may be required to be presented with the response, and so on. Similarly, the **response** provides crucial information such as whether the request succeeded, the file type and encoding of the data, and more.

To solve these challenges, the **WARC** format was created. A 2008 revision of an earlier ARC format created by the Internet Archive, it is today the most widely used format for archival of web pages. The WARC format supports capturing HTTP request and responses with headers and all relevant data, and provides space for additional **metadata** such as the date and time of the capture, or IP addresses of the relevant parties. The specification of the WARC format in version 1.1 is formalized and available from [101].

The WARC format powers the Wayback Machine and is also in use by many national libraries around the world. Although there is a number of known ambiguities and issues with regard to the standard[102]¹², it is widely supported and the tooling surrounding it is relatively mature. The most important omission for our case is the complete lack of support for **WebSockets**. The WebSocket protocol, which enables full-duplex communication over TCP as an upgrade over HTTP and was standardized in 2011[103], does not currently appear to be under consideration for a future version of the WARC standard.

Since Discord makes use of Websockets for some (but not all) of its data transmission, we are forced to look further. Among formats which are designed to capture HTTP traffic and also support WebSocket is **HAR** (HTTP Archive). HAR files are JSON-encoded archives which support logging the interaction between browsers and websites. It is primarily designed for purposes of studying the performance of web pages as they load, as opposed to

¹²The author considers the lack of support for storing certificates exchanged during secure connections a particularly disappointing omission.

long term preservation. While a draft standard of the format by the W3C is available, it is marked as abandoned and cautions against its use[104]. In spite of that, the format is alive and major browsers like Chrome and Firefox can export HAR interactions with web servers from their respective developer tools. Chrome 76 (released in 2019) added support for WebSockets to its HAR export[105]. Unlike WARC files, HAR does not support storing some supplemental data such as DNS lookup results.

While the HAR format appears promising, one major stumbling block is the lack of support for streaming onto disk. Browsers are able to capture traffic in memory and save a HAR archive on demand. Lack of support for streaming means workarounds must be employed such as creating an export at regular intervals to prevent the memory from filling up.

4.2.2 Software

Wget[106] is an example of a tool which can handle many typical cases of web archival. When provided with a list of URLs, it can download HTML and other files, and it also supports outputting in the WARC format, albeit Archive Team notes the implementation is incomplete and immature compared to other specialist archiving systems[107]. Wget does not support the execution of JavaScript.

Frequently employed by Archive Team is **Wpull**, a Wget-compatible downloader implemented in Python. In an attempt to support archiving modern sites, Wpull had implemented PhantomJS support, enabling it to execute JavaScript on websites. However, PhantomJS development has ceased in 2018[108], demonstrating the difficulty of keeping up with the fast pace of evolution of the web.

To handle interacting with complex modern websites, the author concludes the only tool that can keep up with the ever evolving landscape of the web is web browsers themselves. Recent versions of major browsers such as Chrome and Firefox offer APIs which allow them to be driven headlessly using programming language APIs. Those will be discussed individually in Section 4.4.

4.3 Capturing network traffic

If we find ourselves dissatisfied with the formats and techniques available to capture websites, it is possible to go one level higher. Since all interaction between an HTTP client and server happens in the context of packets sent over a network, capturing **network traffic** is a valid and interesting strategy to save all relevant data. Since formats of network protocols are widely known (indeed, they are the perfect representation as no processing is done from the original), in theory, as long as it is contained in the packet capture, the web data can always be derived.

There are two primary methods to network traffic capture: **proxying** and **packet capture**.

A **proxy** is a server application which behaves as an intermediary between a client making a request and the server providing a response. Proxies are effective for capturing network traffic and can even dissect and modify data transmitted in real time. However, some concessions have to be made. Since many important protocols today employ encryption to protect data confidence over the line, proxies have to act as a MITM (man-in-the-middle). This requires support and awareness of the client software interfacing with the proxy, or modifications to program code. By necessity of the observer effect, since proxy servers have to process the packets that they pass through, the behavior of the client and server may not be equivalent to their behavior without the proxy server.

Software which employs **packet capture** does not have to interact with the client program, instead, it makes use of operating system APIs to listen to network adapters and register packets that pass through it. This enables high fidelity capture of the interactions between the client and server on the network layer, including minute details such as routing decisions and packet retransmissions. Since most traffic today is encrypted, this capture alone cannot suffice to read the data. Conveniently, client programs such as browsers or wget support logging the session keys used during TLS traffic to a file, typically using an environmental variable called `SSLKEYLOGFILE`¹³. The packet capture together with the keylog file allow for decrypting the data.

¹³The previous version of the protocol widely known as TCP today was called SSL.

4.3.1 Proxy software

warcprox[109] is a MITM HTTP proxy developed by the Internet Archive. In conjunction with a client, it can capture HTTP traffic and store it in a WARC file. It is a part of the wider infrastructure that powers the Wayback Machine. However, warcprox does not handle WebSocket since the underlying WARC format does not support it.

mitmproxy[110] is a free and open-source interactive HTTPS proxy. It provides a command line interface as well as a web interface, and it offers a Python API for scripting. As opposed to a classic proxy server it has functionality to intercept HTTPS traffic, decrypting it and re-encrypting it as it travels between the client and the server. This naturally requires cooperation from the client, which must be configured to accept mitmproxy's root certificate.

mitmproxy supports saving captured HTTP and WebSocket traffic to a file. An interesting feature mitmproxy offers is server-side replay of server responses from saved HTTP conversations. This feature is useful for debugging and testing, as it sidesteps the need for repeated interactions with a server and improves reproducibility.

The file format of mitmproxy dumps is not particularly well described. It uses a nonstandard implementation of an obscure serialization format called TNetStrings. This format has its problems: it's append-only, meaning streams aggregate in memory before being written, and the fact that it postfixes data types *after* data makes it difficult to index and stream. There have been talks to replace the flow storage format with SQLite[111][112], Protocol Buffers[113] and PCAP[114]. The author will be following these developments with vested interest.

4.3.2 Packet capture software

Packet capture software largely makes use of **libpcap**, which provides a system-independent interface for packet capture.

tcpdump[115] is a free and open source command-line packet analyzer from the libpcap developers. It supports saving packets in the **PCAP** format.

PCAP is a relatively simple binary format which stores a series of packets captured on a single network interface. PCAP uses 32-bit timestamps with up to nanosecond precision¹⁴. To solve some problems with the PCAP format, **PCAPNG** was created. Although not as widespread as PCAP yet, PCAPNG timestamps are now 64-bit timestamps and the format supports making storing captures made from multiple network interfaces simultaneously. Arbitrary comments and metadata may also be stored alongside packet data. These improvements come at the cost of random seeking, but it may still be possible to create indexes to PCAPNG files if they do not involve hotplugging network interfaces. Rust[116] and Node.js[117] PCAPNG parsers are available.

Wireshark[118] is a powerful free and open source packet analyzer which comes with a graphical user interface. The command line tool is called **tshark**. Wireshark also supports saving traffic in the PCAP and PCAPNG formats. Notably, when processing packet captures, Wireshark supports following and reconstructing TCP stream data even when the packets arrive out of order or are retransmitted. Wireshark supports dissecting many protocols, including HTTP2 and Websockets, and outputting their information and contents in structured formats such as XML and JSON. When provided with a TLS keylog file, it can decrypt TLS traffic. In addition, Wireshark allows users to write Lua scripts to work with parsed packet data directly.

While the packet capture capabilities of tcpdump and Wireshark are similar, support for parsing hundreds of protocols directly makes Wireshark more useful than tcpdump for in-depth protocol analysis.

4.4 Headless browsers

In the recent years, it became possible to control browsers such as Chrome and Firefox headlessly using automation APIs. The primary motivation for this is automated testing of websites, web applications, and extensions; but additional uses include screenshot generation, pre-rendering content of websites (i.e. SSR, server-side rendering), inspecting performance, crawling web content, and more. There are many applications for this: automated testing, scraping, and automation of web applications.

¹⁴This will make the PCAP format fall victim to the Year 2038 problem.

4.4.1 Selenium

Selenium[119] is one of the earliest projects in this space, with its development beginning in 2004[120]. Initially implemented as a browser extension injecting JavaScript into the page, today it is implemented using browser APIs. Selenium offers libraries for many projects like Python, Java, and C#. It is primarily designed to enable testing websites with a wide variety of browsers as opposed to general automation.

4.4.2 Puppeteer

Puppeteer[121] is a Node.js library capable of driving a browser headlessly. It is maintained by the Chrome DevTools team and as such primarily designed for Chrome, but experimental Firefox support is available. It is primarily designed for testing and automation. There are many helper libraries available for Puppeteer, such as puppeteer-extra offering convenience features like ad-blocking and stealth[122]. Puppeteer can also be used to directly interact with Electron apps.

4.4.3 Playwright

Playwright[123] is a Node.js web testing and automation framework from Microsoft. It supports Chromium, Firefox, and WebKit. Besides Node.js, it also provides libraries for Python, C#, and other languages. The API is similar to Puppeteer and the two are highly comparable¹⁵, but Puppeteer is still broadly better supported by add-ons such as puppeteer-extra.

4.5 TypeScript and Node.js

JavaScript is the standard programming language of the Web. Single page web applications use JavaScript by necessity and are either written in it or are transpiled into it. As such, for a project which interfaces with a web application, JavaScript is a natural fit.

TypeScript is a programming language developed by Microsoft which is a syntactic superset of JavaScript and adds support for static typing to the language. TypeScript is compiled into JavaScript. TypeScript makes

¹⁵In fact, much of the original developers of Puppeteer essentially moved to Microsoft to work on Playwright.[124]

it possible to define types and interfaces and makes writing code targeting JavaScript environments overall safer and more pleasant. Strong typing is useful when developing a program which relies on external input and where edge cases may not be easily reproducible.

Node.js is a JavaScript runtime environment is designed for executing JavaScript outside of a web browser, but permits the use of many of the same libraries and similar APIs. Frameworks which provide APIs to control browsers headlessly, such as Puppeteer, are frequently implemented for Node.js.

4.6 Linux containers

Containers refer to technology which enables OS-level virtualization, typically in terms of Linux systems. Containers isolate the environment a given application runs in and can simplify the deployment of software by ensuring all dependencies are installed[125]. For archival, it's especially important to ensure a stable, reproducible environment, so any archival tool which relies on multiple software with different versions (browsers, libraries, etc.) benefits from being possible to set up and run in a container. Additionally, since containers isolating the network stack, packet capture software such as Wireshark can be used without the need for special permissions.

Docker[126] and **Podman**[127] are representative in this space.

4.7 Elasticsearch

Elasticsearch[128] is a free and source-available search and analytics engine. It offers a REST-ful HTTP interface and operates on schema-free JSON documents. This is particularly useful when working with data which may have an uncertain shape and attributes, such as chat message objects received through an API. Elasticsearch is able to index these objects automatically and provides a fulltext search interface. **Kibana**[129], another part of the Elastic stack, is data visualization dashboard software which allows the user to quickly explore and visualize the data.

4. TECHNOLOGIES

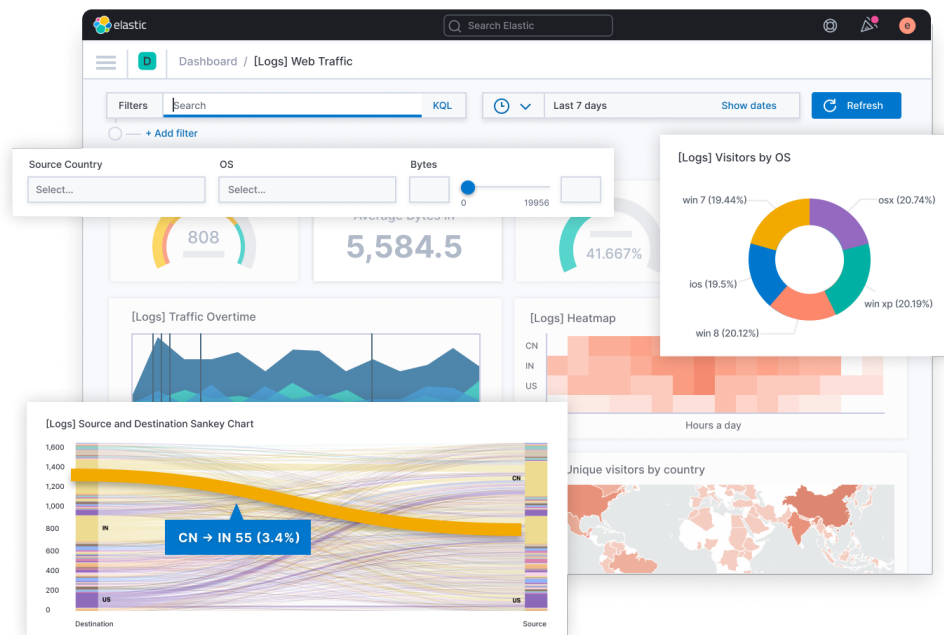


Figure 4.1: Marketing screenshot of Elastic's Kibana, demonstrating its ability to create data dashboards.

Implementation

In this section, we look at the implementation of the main project of this thesis, the archival tool for the Discord chat platform. For simplicity, the project will henceforth be dubbed **Discard2**.

Discard2 is written in **TypeScript** using **Node.js**. It consists of two main components: the **crawler** and the **reader**. The crawler is responsible for connecting to the Discord servers and downloading the requested data into a specified directory in a format suitable for archival. It uses a **capture tool** to accomplish the task of saving the client-server traffic. The reader is responsible for reading the data from the archive and converting it to other usable formats.

Discard2 is designed to run on Linux and can either run as a regular program when all dependencies are installed, or operate inside a Docker or Podman container. Running Discard2 in a container makes installation easier and establishes a more consistent, reproducible environment, and is thus encouraged for production uses.

Discard2 is made available as an open-source project and is licensed under the MIT license. The repository is hosted on GitHub at <https://github.com/Sanqui/discard2>.

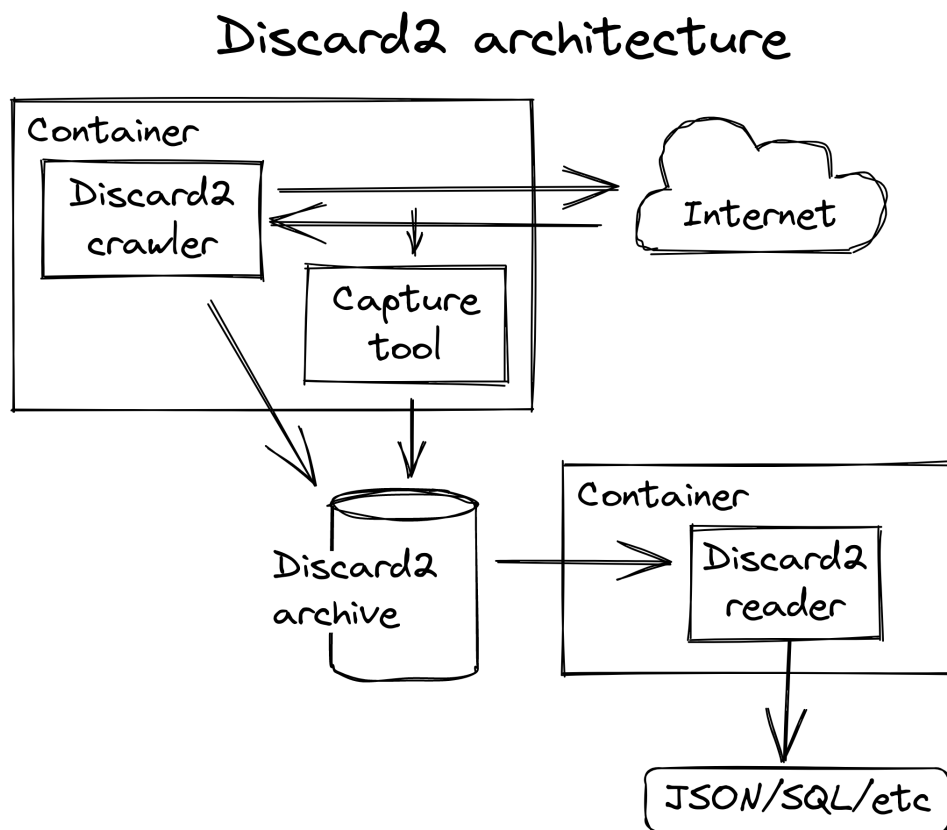


Figure 5.1: A diagram describing the architecture of Discard2.

5.1 Crawler

The **crawler** part of Discard2 is implemented as a headless browser controlled using **Puppeteer**. It operates on a task-oriented job system. The user (or another script) assigns the crawler a **job**, which consists of any number of **tasks**, such as downloading specified channels. Meanwhile, a **capture tool** is deployed which captures traffic between the client and server to the output directory.

When working with dates, Discard2 operates in UTC to eliminate ambiguity.

5.1.1 Browser

The browser in the crawler is controlled using **Puppeteer**. Puppeteer experimentally supports Firefox, but is primarily designed for Chrome. Since Chrome is the more widely used browser, to ensure the best compatibility it is also used by Discard2.

In deployment and in containers, the browser is launched headlessly, i.e. with no user interface. However, for debugging purposes, it is also possible to launch the browser with a GUI and observe the behavior of the crawler. Even when running headlessly, when an error is encountered during the crawl, a screenshot is taken to help diagnose the issue.

The crawler also enables the user to provide a data directory for use by the browser. This enables cookie re-use between sessions and can alleviate the need to re-authenticate the account every time a job is run.

Finally, to reduce traffic and improve performance, the browser can be configured to block requests to images. Because images can be accessed from Discord servers without authentication, other standard tools can be used to archive them after the fact without the overhead of a browser.

5.1.2 Jobs

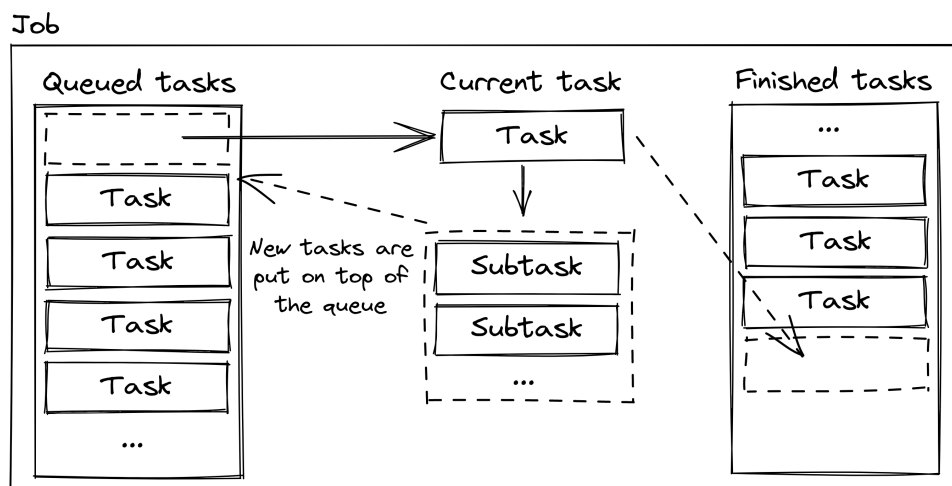


Figure 5.2: Processing of a job in Discard2's crawler.

A **job** is a single run of the crawler with predetermined settings and tasks. The crawler performs these tasks in sequential order, in some cases breaking a task down into smaller tasks as information is received from the server. Completed and pending tasks are stored to a file in the output directory as a part of the **job state**. This enables resumption of a job if the crawler crashes or is terminated.

The current state of the job is saved to a state file after the completion of every task. Should a job be suspended or fail, it is possible to resume it from the failed task using this file, so that long-running tasks don't have to be repeated.

The user specifies a starting task using the CLI. It is possible to provide multiple tasks by providing a starting JSON state file.

5.1.3 Discord project

In Discard2, the **project** refers to the website or platform being archived. For the sake of this thesis, only a single type of project has been implemented, that being **Discord**, but the crawler is written to support projects with other types of tasks.

The Discord project implements the following main tasks:

Login. The login task is automatically performed on the start of a Discord project. It navigates to the Discord login screen. If presented with a login form, it fills it with the account e-mail and password, which must be provided. Should it encounter a CAPTCHA, it informs the user to log-in manually in a separate browser from the same IP, which clears the account flag.

Profile. The profile task opens the user settings in order to fetch additional information about the current user as well as verify the account e-mail to prevent use with an unexpected account.

DM. The DM task is provided with four parameters: the ID of the DM and optionally, "before" and "after" dates. It opens the specified DM and performs a search for messages between the specified dates. It chooses the earliest message and proceeds to scroll the chat until reaching the latest

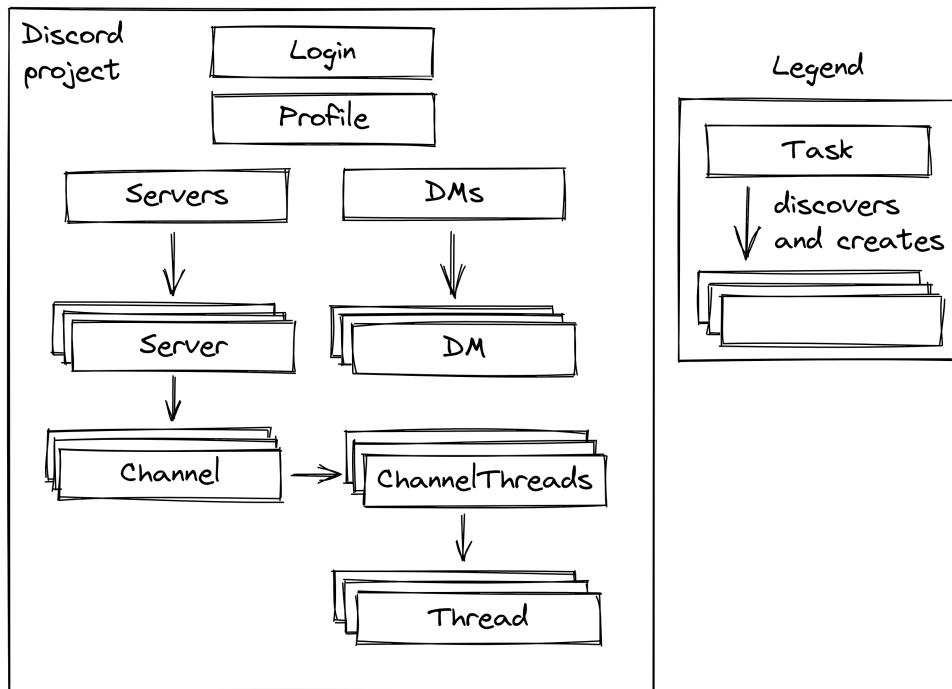


Figure 5.3: Hierarchy of tasks in the Discord project.

message.

Channel. The channel task is provided with four parameters: the server ID, the channel ID, the same "before" and "after" dates. It opens the server, locates the channel, and proceeds to download the chat history similarly to the DM task.

Thread. The thread task requires the server ID, the channel ID, and the thread ID. It opens the server, locates the channel, in the list of threads, finds and opens the requested threads, and proceeds to download chat history in the thread. Because Discord doesn't offer searching within threads, it's not possible to fence the download by dates.

Server. The server task is provided with a server ID and the same "before" and "after" dates as the channel task. It opens the specified server and creates a new task for each channel in the server.

5.1.4 Capture tools

While the Discard2 crawler does by necessity parse some data displayed in the application, its main purpose is to access the data and no effort is made to register and save it. Instead, a **capture tool** is employed to save the traffic between the client and the server. Besides the "dummy" capture tool, Discard2 supports two primary capture tools: **mitmproxy** and **Wireshark**.

5.1.4.1 mitmproxy

The mitmproxy capture tool, also called **mitmdump** by the command-line program employed, is the best functional capture tool. It behaves as a proxy between the browser and the server, intercepting traffic and logging HTTP and Websocket *flows* to a file (mitmproxy does not specify a name for its capture format; Discard2 uses `capture.mitmdump` for the filename).

5.1.4.2 Wireshark

The Wireshark capture tool, also called **tshark** by, again, the command-line program employed, is the second capture tool implemented. Wireshark is able to capture network traffic using the pcap interface without the need for a proxy. The capture is more high fidelity, because includes DNS requests, unmodified TCP/IP packet headers, and the handshakes for encrypted communication. Wireshark saves traffic in the **pcapng** format.

Wireshark requires the user account on the host machine to have permission to capture traffic using the operating system's pcap interface. Since Wireshark cannot differentiate or filter traffic from a single application, it captures all traffic made on the system, some of which may be sensitive. For these reasons, it is recommended to use Discard2 with the Wireshark capture tool only inside a container.

5.2 Reader

The reader part of Discard2 exists to process the archives from completed jobs containing captured data and output them in formats suitable for further work. In archival, this process is called *derivation*.

When provided with a job path, the reader validates the job state and depending on the capture tool used, reads data from the capture. In cases of both mitmproxy and Wireshark, it's necessary to have the program installed (or run the tool reader in the standard container).

When the mitmdump capture tool is used, it's possible to read all data gathered from the server. However, **due to a bug in Wireshark, it was not possible to implement reliable retrieval of data from the pcapng format.** When parsing the packet data, Wireshark is typically able to follow the TCP stream and provide decrypted and reassembled HTTP data. However, in case of an out-of-order or retransmission event of a TCP packet, Wireshark may lose its ability to follow the stream. This is despite the implementation of TCP reassembly support in 2018[130]. The author has contacted the Wireshark community with regard to the issue.

5.2.1 Output formats

Discard2's reader currently supports the following output formats:

- **raw-print** - plain text overview of requests and responses,
- **raw-jsonl** - machine-readable JSON lines with full request and response data,
- **print** - plain text log of messages (suitable for grep),
- **elasticsearch** - message data in format for import to an Elasticsearch index,
- **derive-urls** - URLs of images and attachments for archival by other tools.

raw-print. The raw print output format is a simple text format showing an overview of HTTP and Websocket requests without their full content. It's intended for developers to quickly understand the data contained in the archive.

Sample output:

5. IMPLEMENTATION

```
HTTP > GET /api/v9/users/@me/billing/country-code
HTTP < 200 {"country_code":"CZ"}
HTTP > GET /api/v9/users/954364925218783293/profile?with_mutua...
HTTP < 200 {"user":{"id":"954364925218783293","username":"test...
HTTP > GET /api/v9/users/@me/billing/payment-sources
HTTP < 200 []
HTTP > GET /api/v9/channels/954365197735317517/messages?limit=50
HTTP < 200 [{"id":"961971917856862238","type":0,"content":"355...
```

raw-jsonl. The raw JSONL format is a plain list of HTTP requests and responses, as well as WebSocket interactions read from the capture, with timestamps. It's particularly suitable for processing by further applications.

Sample output:

```
{"type":"http","timestamp_start":1649614252.9561036,"timestamp_end":1649614254.0228916,"request":{"method":"GET","path":"/api/v9/channels/961993711884062751/messages?limit=50"},"response":{"status_code":200,"data":[]}}
{"type":"ws","timestamp":1649614268.277466,"direction":"recv","data":{"t":"THREAD_LIST_SYNC","s":4,"op":0,"d":{"threads":[],"most_recent_messages":[]},"guild_id":"954365197735317514"}}
```

print. The print format outputs all messages contained in the capture. While such output is not a complete representation of message data (it does not include reactions, embeds etc.), it may be used to get an overview of contents of the archive, and may be useful for searching through archives using standard tools such as `grep`.

Sample output:

```
[2022-03-20T13:14:37.700000+00:00] 954365219411460138:
<test_ahcae#7949> test 0
[2022-03-20T13:15:49.983000+00:00] 954365219411460138:
<test_ahcae#7949> image attachment
[2022-04-08T12:59:45.398000+00:00] 961973542663122944:
<test_ahcae#7949> chat msg
```


5. IMPLEMENTATION

The `--help` option shows all available commands and options. Complete usage instructions may be found in the `README.md` file of the project's repository.

When the crawler is running, Discard2 outputs information about the tasks being fulfilled, allowing the user to track progress. When a channel is being downloaded, a progress bar is shown providing an automatically calculated ETA (Figure 5.4).

5.4 End-to-end tests

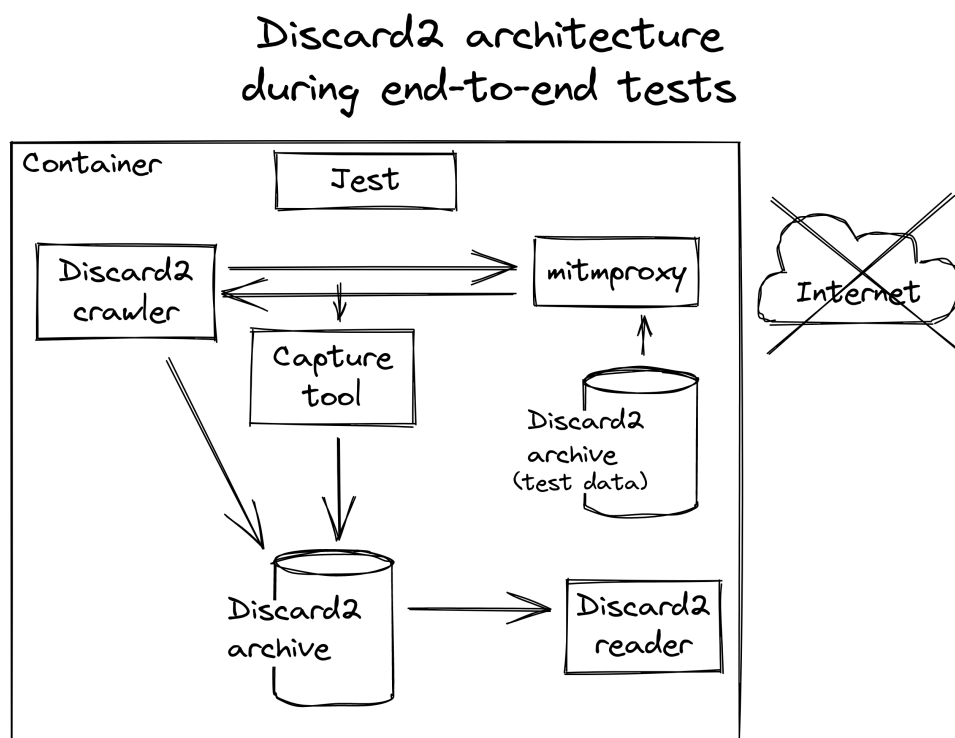


Figure 5.5: Diagram describing Discard2 architecture during end-to-end tests.

In modern software development, tests are of essence to evaluate and verify that the application does what it's supposed to do. For testing Discard2, the *end-to-end* testing technique was employed, since the project has many

components that need to work together: the crawler controls a browser while a capture tool is running, and the reader extracts the data from the resulting capture and verifies that the test data is present. However, while testing, it is undesirable to hit the servers of the target service, since that causes unnecessary load and hampers reproducibility. Instead, mitmproxy's server-side replay feature is used, simulating interaction with the real server from a pre-existing capture, while enabling the tests to re-capture the data and test deriving it. The **Jest** library for Node.js is used to run multiple test scenarios, such as archiving a single channel or server. Thanks to GitHub's free GitHub Actions service, tests are run automatically on every commit (referred to as CI).

5.5 Findings

Discord2 works according to the design specification. Still, some limitations to the approaches chosen were identified.

Overall, implementing the **crawler** was of medium difficulty. Interfacing with the UI of Discord is done mainly by utilizing CSS selectors. For instance, to locate the button which opens the server with the ID `954365197735317514`, it is possible to use the selector `[data-list-item-id=guildsnav___954365197735317514]` and click the found element. Then, it is necessary to wait for the desired content to appear, or retry the click. If a given object is in a list which may be scrolled, it may be loaded dynamically, requiring scrolling of the list to locate it. In one instance, that being the list of threads in a given channel, IDs of items in the list are not exposed through the DOM. This necessitates the capture of the relevant HTTP request and determining the index of the desired object in the list in order to click it.

When compared to work with an API directly, interacting with the UI of the application is more complicated. It is necessary to strategically wait for elements to load and unexpected dialogs may pose problems. However, when identified, these issues can be addressed relatively easily. Whether this strategy will be resilient in face of long term changes to the application will have to be determined in the future, however, the author believes that as Discord adds more features which may be undocumented in the API, implementing

5. IMPLEMENTATION

them by interfacing the UI will be relatively quick and give this archiver an edge.

In terms of the **reader**, interfacing with the capture tool in order to extract data from captured packets was the most difficult. While mitmproxy provides a relatively simple Python API to extract request data, Wireshark's export methods are complicated and messy. With the correct combination of command line flags, it's possible to get JSON output which is not ambiguous (by default, Wireshark has no problem with outputting duplicate keys in objects). However, Wireshark does not provide HTTP request-response pairs, so it was necessary to re-implement parts of the HTTP2 protocol's behavior in order to extract the data. Even then, it was discovered Wireshark cannot reliably reconstruct TCP streams after out-of-order or retransmission events. Although the captures made by Wireshark are most likely complete, it is not currently possible to reconstruct the data from all of them.

Once request-response pairs are attained, parsing the data is a matter of identifying interesting HTTP requests by reading the path. For instance, every request for messages in a channel is in the form `GET /api/v9/channels/962731115729276948/messages?after=956547528042635264&limit=50`, where the first number is the ID of the channel, and the `after` parameter denotes the ID of message to fetch messages following. The response is a JSON serialized list of message objects. Discard2's reader can output these messages directly as JSONL for future processing. Besides these API requests, the capture contains the code for the Discord application at time of capture, which is a benefit for longevity of the archive and future understanding.

Validation

In order to demonstrate the functionality of the tool, a server was chosen to be archived and a simple analysis of the data was performed to verify that the data appears correct. ”FIT ČVUT” is a Discord server ran by students of the Faculty of Informatics on the Czech Technical University in Prague. It was started in 2017 and currently has 3,547 members in it, primarily students. It can be described as a medium-sized server. With the permission of the server’s owner, a Discord user account was created, joined the server, and was granted the roles necessary to see the channels for Bachelor’s and Master’s students.

On 2022-04-23, Discard2’s crawler was used to download approximately 590 thousand chat messages from the FIT ČVUT server. This process took approximately 4.5 hours and the resulting archive totals 345.2 MB. On 2022-04-28, an additional approximately 40 thousand messages were downloaded from threads, which were not covered by the original capture. Discard2’s reader was used to import this data into an Elasticsearch database for further analysis.

On 2022-05-02, searches for test strings were performed on the dataset and in the Discord server using the same account that was used to download the data. The number of search results was compared in Table 6.1 and found to be very similar. A slight deviation can be explained by different fulltext indexing methods as well as messages that have changed in the meantime. We can therefore conclude that the downloaded archive is practically complete and the tool works well.

Search string	Number of results	
	Discord	Archive
"test"	3,267	3,302
"Discord"	1,457	1,601
"FIT"	4,237	4,225
"web"	563	566

Table 6.1: Number of results for various search strings

6.1 Analysis

630,514 total messages in the ČVUT FIT server sent between July 2017 and April 2022 were analyzed with the use of Elasticsearch and Kibana.

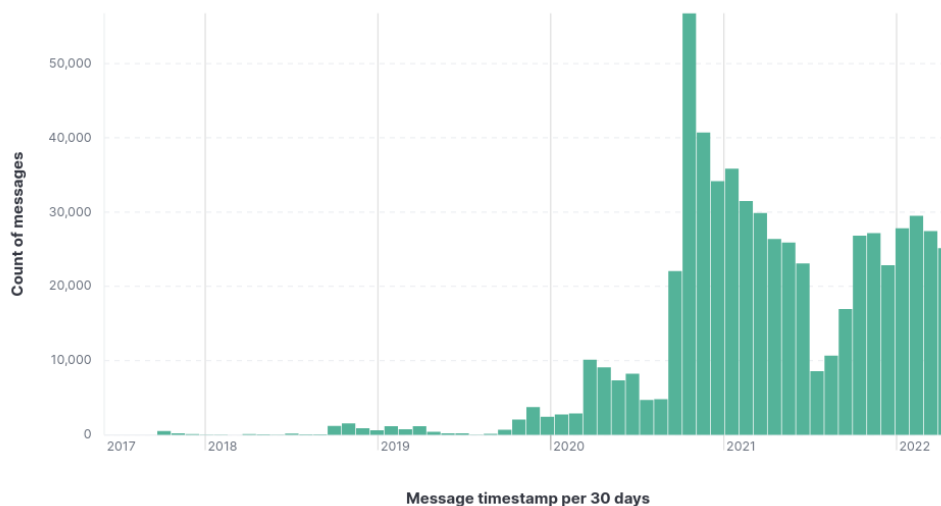


Figure 6.1: Activity over time.

Figure 6.1 displays the amount of messages per month and already reveals interesting facts about activity on the server. We will note how activity sprung up from February 2020 to March 2020, with the amount of messages more than tripling from 2,881 to 10,125. This is easily explained by the worldwide lockdowns caused by the COVID-19 pandemic. Internet use rose greatly during the pandemic, including in the Czech Republic[131]. While decreased activity is noted in the summer months of 2020, beginning with September, activity rose again, and with 56,767 messages, October 2020 yielded peak

activity. This can be explained by the influx of first year students to the server, who, unable to attend classes in person, sought a way to socialize with their new peers. After another drop the following summer, in the semesters of the 2021/2022 academic year, activity on the server has been at a stable approximately 27,000 messages per month.

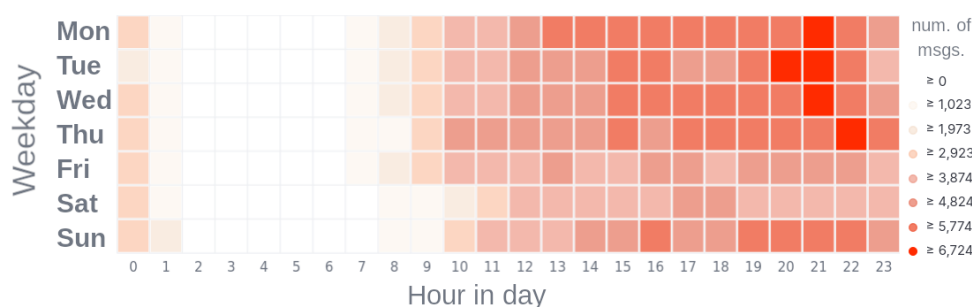


Figure 6.2: Heatmap of activity over the course of a week.

Figure 6.2 displays the accumulated activity in each hour of the week. It allows us to see how people gradually wake up between 7 and 9, with activity being stable from 10:00 onwards. Peaks of activity are seen in the evening, around 21:00. After midnight, activity rapidly drops, with very little people posting messages after 2:00. Activity beings about two hours later on weekends, which is consistent with the fact that most people have no class or work on these days. Saturday is the weakest day overall, with Sunday seeing consistent activity in the evening, which might be correlated with Sunday midnight being a common deadline for assignments.

Figure 6.3 lists the 20 most active channels, or chat rooms by count of messages. The top four channels, about politics, dating, off-topic, and the novel coronavirus respectively, reflect the social nature of the server. However, it makes sense that channels about individual subjects see more spread out activity. Still, two channels about programming subjects, PA1 and PA2, two channels about math subjects, ZMA and LIN, and a channel for the algorithms and graphs subject, AG1, all make it to the top 20. Overall, 55 % of activity takes place outside the top 20 channels.

6. VALIDATION

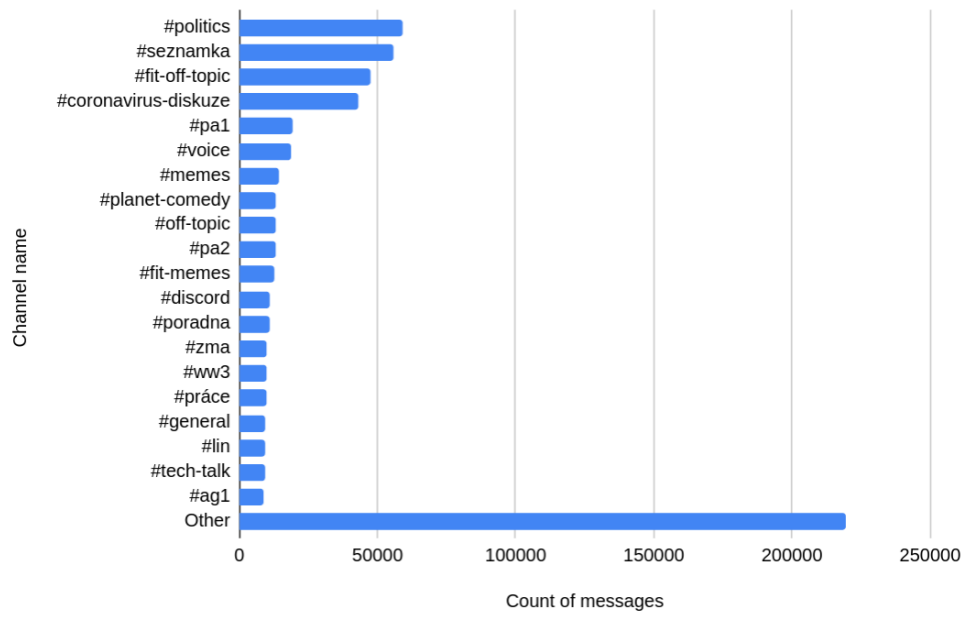


Figure 6.3: 20 most active channels.

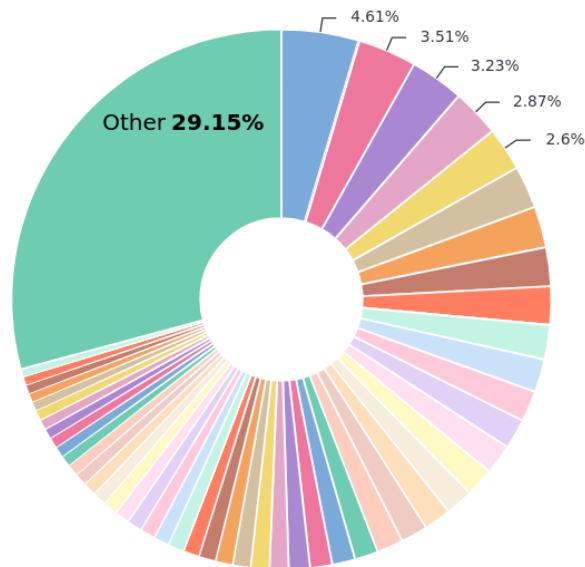


Figure 6.4: 50 most active users.

The pie chart at Figure 6.4, while not particularly enlightening due to anonymization of the data, does show that the majority of messages, about 71 %, are sent by the top 50 users. 2,034 unique users have sent messages in total.



Figure 6.5: Messages sent per minute (blue) against average sentiment (green).

In Figure 6.5, the number of messages per minute beginning with 2020 was plotted against the average message sentiment¹⁶, smoothed over 3 days. Again, activity can be seen to rise in the wake of the pandemic, and further yet once new students come in September. We note that message sentiment does not correlate to the absolute activity. Overall, message sentiment is going down, the author interprets this as messages sent in the server becoming more neutral (perhaps less humorous in tone) over time. A small yet notable dip can be seen at immediately after 2022-02-24, the day of the Russian invasion of Ukraine.

¹⁶Because the majority of text is in Czech, sentiment for a given message was determined using the simple technique of adding together sentiment values of emoji present in the message and in reactions to the message. Sentiment ranking was provided by the *Sentiment of emojis paper*, which determined emoji polarity using tweets in European languages[132].

6. VALIDATION



Figure 6.6: Word cloud of thread names.

Threads are a relatively recent Discord feature (introduced in July 2021) which allows users to create temporary channels for a single topic. Users of the FIT ČVUT server have created 3,231 threads to date. Figure 6.6 shows a word cloud of common words used in thread names. We observe that the most common threads are started in relation to exams, or tests. *Progtest* refers to the faculty platform on which weekly programming homework is served. Similarly, *Marast* is a platform for math exercises.

Conclusion

In this work, we have noted the importance of digital archival, the value in making backups of chat logs, and possible uses for them. We have gone over the situation with regard to archiving content on the Internet, in particular websites. We have learned that existing generalized web archival tools have trouble with modern single page applications such as Discord. While advancements in legislation are being made on the front of personal data freedom, Discord does not offer its users to download complete chat logs they have access to.

Specialized archival tools which assist users in making backups of Discord chats were compared. However, by reimplementing the Discord API, some of them carry heightened risk of account suspension. They also process the data while downloading it, which is poor practice in preservation as it carries the risk of silently losing data and makes it difficult to keep up with changes and updates made to the protocol. Requirements for a new Discord archiver were conceived, one which manipulates the official Discord client using a headless browser while performing a raw capture of the transmitted data.

The new archiver for the Discord platform, named **Discard2**, is implemented using Node.js and consists of two parts. The crawler, which downloads data from Discord's servers, controls a headless Chrome browser using Puppeteer. It uses a flexible task system which allows for downloading messages from channels, threads, servers, and DMs. Mitmproxy and Wireshark were implemented as capture tools, saving traffic between the client and the server

into an archive. Discard2's reader is a separate component which can extract message data from a given archive. It supports outputting to a variety of formats, including Elasticsearch, which can be used to search and analyze the data. Discard2 is licensed under the MIT license and can be downloaded on GitHub at <https://github.com/Sanqui/discard2>.

Functionality of the archiver tool was demonstrated by archiving 630,514 messages from the student Discord server of the Czech Technical University's Faculty of Informatics. The completeness of this archive was validated and simple statistical analysis on the data was performed.

7.1 Future work

During the development of the archiver, it was discovered that Wireshark's support for reassembling TCP streams is incomplete. Therefore, archives made using the Wireshark capture tool cannot be reliably read. Because of this limitation, mitmproxy is the preferred capture tool. The author hopes to cooperate with Wireshark developers to implement this functionality, because it would result in higher fidelity captures.

In general, the tool implemented in this thesis is functional according to the specification. Improvements can be made on the front of reliability. Browsers are complex pieces of software and a variety of situations, such as network errors and out of memory events, can make them fail. Discard2's crawler supports resuming a job from failure, but these restarts do not yet happen automatically, and task resumption is not as fine-grained as it could be (some data is downloaded twice). Discard2's reader can be made to understand more Discord features (such as stickers). Because archives are complete thanks to the traffic capture strategy, more export functionality can always be implemented in the future without blocking current archival efforts.

The tool has been presented to interested Archive Team members and a roadmap for future functionality is underway. The author also intends to re-use the framework for archival projects of other online applications, as it has been built with flexibility in mind.

Bibliography

- [1] Jones, Q.; Mihai, M.; et al. Empirical evidence of information overload constraining chat channel community interactions. 01 2008, pp. 323–332, doi: 10.1145/1460563.1460616. Available from: https://www.researchgate.net/publication/220878890_Empirical_evidence_of_information_overload_constraining_chat_channel_community_interactions
- [2] Discord. What makes Discord different? 2022. Available from: <https://discord.com/why-discord-is-different>
- [3] Czech Technical Unvirseity in Prague. Czech Technical Unvirseity in Prague [Wayback Machine]. 1998. Available from: <https://web.archive.org/web/19981206104022/http://www.cvut.cz/ascii/index.html>
- [4] Pierce, D. How Discord (somewhat accidentally) invented the future of the internet. 2020. Available from: <https://www.protocol.com/discord>
- [5] Curry, D. Discord Revenue and Usage Statistics (2022). 2022. Available from: <https://www.businessofapps.com/data/discord-statistics/>
- [6] Domo. Data Never Sleeps 9.0. 2021. Available from: <https://www.domo.com/learn/infographic/data-never-sleeps-9>
- [7] @discord Twitter account. All 3rd party apps or client modifiers are against our ToS, and the use of them can result in your account being disabled. I don't recommend using them. 2022. Available from: <https://twitter.com/discord/status/1229357198918197248>
- [8] Telegram Team. Chat Export Tool, Better Notifications and More. 2018. Available from: <https://telegram.org/blog/export-and-more>
- [9] The Matrix.org Foundation C.I.C. Matrix.org: An open network for secure, decentralized communication. 2021. Available from: <https://matrix.org/>
- [10] Plunkett, L. Please Stop Closing Forums And Moving People To Discord. 2021. Available from: <https://kotaku.com/please-stop-closing-forums-and-moving-people-to-discord-1847684851>

BIBLIOGRAPHY

- [11] Discord. Discord API Change Log. 2022. Available from: <https://discord.com/developers/docs/change-log>
- [12] Surge AI. Is Google Search Deteriorating? Measuring Google's Search Quality in 2022. 2022. Available from: <https://www.surgehq.ai/blog/is-google-search-deteriorating-measuring-search-quality-in-2022>
- [13] Keeton, J. September 11th, 2001, through the eyes of IRC channels. 2022. Available from: <https://www.dailydot.com/debug/september-eleventh-through-eyes-irc-channels/>
- [14] Hochstein, L. Bitrot. 2022. Available from: <https://surfingcomplexity.blog/2022/01/23/bitrot/>
- [15] Orosz, G. The Scoop: Inside the Longest Atlassian Outage of All Time. 2022. Available from: <https://newsletter.pragmaticengineer.com/p/scoop-atlassian>
- [16] Viswanath, S. Update on the Atlassian outage affecting some customers. 2022. Available from: <https://www.atlassian.com/engineering/april-2022-outage-update>
- [17] Pusin, Y. The State of Backups: Who's Most at Risk. 2021. Available from: <https://www.backblaze.com/blog/the-state-of-backups-whos-most-at-risk/>
- [18] Archive Team wiki editors. Deatchwatch - Archive Team Wiki. 2022. Available from: <https://wiki.archiveteam.org/index.php/Deathwatch>
- [19] Wootten, I. Are Product Hunt's featured products still online today? 2022. Available from: <https://www.scrapingbee.com/blog/producthunt-cemetery/>
- [20] Ogden, C. Killed by Google. 2022. Available from: <https://killedbygoogle.com/>
- [21] Archive Team wiki editors. Yahoo! - Archive Team wiki. 2021. Available from: <https://wiki.archiveteam.org/index.php/Yahoo!>
- [22] The Free Dictionary. "link rot". American Heritage® Dictionary of the English Language, Fifth Edition, Houghton Mifflin Harcourt Publishing Company. 2011. Available from: <https://www.thefreedictionary.com/link+rot>
- [23] ZOMDir. The half-life of a link is two years. 2017. Available from: <https://blog.zomdir.com/2017/10/the-half-life-of-link-is-two-year.html>
- [24] Weblock. All-Time Weblock Report. 2015. Available from: <https://web.archive.org/web/20160304081204/https://weblock.io/report?id=all>
- [25] Internet Archive. Internet Archive: Digital Library of Free & Borrowable Books, Movies, Music & Wayback Machine. 2022. Available from: <https://archive.org/>
- [26] Internet Archive. Used Paired Space. 2022. Available from: <https://archive.org/~tracey/mrtg/du.html>
- [27] Internet Archive. Internet Archive: Wayback Machine. 2022. Available from: <https://archive.org/web/>

-
- [28] Jessica Ogden, S. W., Ed Summers. Participatory Web Archiving: Opening the Black Box of 'Save Page Now'. 2019. Available from: <https://jessogden.github.io/assets/pdf/Ogden-Summers-Walker-RESAW19-Slides.pdf>
- [29] Kahle, B. 651,621,510,000 web URL's now in the Wayback Machine by @internetarchive. ... 2018. Available from: https://twitter.com/brewster_kahle/status/994380510011928578
- [30] Archive Team. Archive Team. 2022. Available from: https://wiki.archiveteam.org/index.php/Main_Page
- [31] Webrecorder. Webrecorder. 2022. Available from: <https://webrecorder.net/>
- [32] redigit (@Demilogic). @Google my account has now been disabled for over 3 weeks. ... 2021. Available from: <https://twitter.com/Demilogic/status/1358661840402845696>
- [33] haistakaavittukaikkisaatana. My Google account got suspended because of NewPipe · Issue #2723 · TeamNewPipe/NewPipe. 2019. Available from: <https://github.com/TeamNewPipe/NewPipe/issues/2723>
- [34] Bode, K. Google Locks Historian's Account Over Terrorism Research Videos. 2021. Available from: <https://www.vice.com/en/article/qj8yj7/google-locks-historians-account-over-terrorism-research-videos>
- [35] karlicoss. karlicoss/HPI: Human Programming Interface. 2022. Available from: <https://github.com/karlicoss/HPI>
- [36] Willison, S. Dogsheep. 2022. Available from: <https://dogsheep.github.io/>
- [37] Hernbroth, M. Slack, the newly-public chat app worth about \$20 billion, has a hidden meaning behind its name. 2019. Available from: <https://www.businessinsider.com/where-did-slack-get-its-name-2016-9>
- [38] Haughey, M. Shrinking the haystack: how to narrow search results in Slack. 2021. Available from: <https://slack.com/blog/productivity/shrinking-the-haystack-how-to-narrow-search-results-in-slack>
- [39] Graham, M.; Elias, J. How Google's \$150 billion advertising business works. 2021. Available from: <https://www.cnbc.com/2021/05/18/how-does-google-make-money-advertising-business-breakdown-.html>
- [40] Potoroaca, A. Google updated look of search results makes ads less obvious. 2020. Available from: <https://www.techspot.com/news/83683-google-updated-look-search-results-makes-ads-less.html>
- [41] Martori, A. Spamdexing: What is SEO Spam and How to Remove It. 2020. Available from: <https://blog.sucuri.net/2020/02/spamdexing-seo-spam.html>
- [42] dkb868. Google Search Is Dying. 2022. Available from: <https://dkb.io/post/google-search-is-dying>
- [43] Reddit. Reddit. 2022. Available from: <https://www.reddit.com/>

BIBLIOGRAPHY

- [44] Reddit Staff. New on Reddit: Comment Search, Improved Search Results Relevance, Updated Search Design. 2022. Available from: <https://www.redditinc.com/blog/new-on-reddit-comment-search-improved-search-results-relevance-updated-search-design>
- [45] Disboard. Public Discord Servers — DISBOARD: Discord Server List. 2022. Available from: <https://disboard.org/servers>
- [46] OWL, P. Project Owl OSINT. 2022. Available from: <https://projectowlosint.org/>
- [47] Disboard. Discord servers tagged with osint — DISBOARD. 2022. Available from: <https://disboard.org/servers/tag/osint>
- [48] Peter Aldhous, C. M. How Open-Source Intelligence Is Helping Clear The Fog Of War In Ukraine - BuzzFeed news. 2022. Available from: <https://www.buzzfeednews.com/article/peteraldhous/osint-ukraine-war-satellite-images-plane-tracking-social>
- [49] Bellingcat Investigative Tech Team. How to Archive Telegram Content to Document Russia's Invasion of Ukraine. 2022. Available from: <https://www.bellingcat.com/resources/how-tos/2022/03/08/how-to-archive-telegram-content-to-document-russias-invasion-of-ukraine/>
- [50] Brown, T. B.; Mann, B.; et al. Language Models are Few-Shot Learners. 2020. Available from: <http://arxiv.org/abs/2005.14165>
- [51] Sagar, R. OpenAI Releases GPT-3, The Largest Model So Far. 2020. Available from: <https://analyticsindiamag.com/open-ai-gpt-3-language-model/>
- [52] Apideck. Chatbots — GPT-3 Demo. 2022. Available from: <https://gpt3demo.com/category/chatbots>
- [53] Radford, A.; Wu, J.; et al. Language Models are Unsupervised Multitask Learners. 2018. Available from: <https://d4mucfpksyww.cloudfront.net/better-language-models/language-models.pdf>
- [54] Dima. What is Data Liberation. 2014. Available from: <https://medium.com/dima-korolev/what-is-data-liberation-50c0fca31eee>
- [55] EUR-Lex. REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL. 2016. Available from: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A02016R0679-20160504&qid=1532348683434>
- [56] European Parliament Press Releases. Deal on Digital Markets Act: EU rules to ensure fair competition and more choice for users. 2022. Available from: <https://web.archive.org/web/20220404000556/https://www.europarl.europa.eu/news/en/press-room/20220315IPR25504/deal-on-digital-markets-act-ensuring-fair-competition-and-more-choice-for-users>
- [57] Jonathan Shieber, T. Discord now has a \$3.5B valuation and \$100M for a sales pitch lighter on the gaming. 2020. Available from: <https://techcrunch.com/2020/06/30/discord-now-has-a-3-5b-valuation-and-100m-for-a-sales-pitch-lighter-on-the-gaming/>

-
- [58] Matthew Hodgson, M. How do you implement interoperability in a DMA world? 2022. Available from: <https://matrix.org/blog/2022/03/29/how-do-you-implement-interoperability-in-a-dma-world>
- [59] @discord Twitter account. We use electron for the desktop app, so it's all javascript and react! 2017. Available from: <https://twitter.com/discord/status/822874230631100416>
- [60] Discord. Discord's Terms of Service. 2022. Available from: <https://discord.com/terms>
- [61] cancel. Ripcord: Desktop Chat Client. 2021. Available from: <https://cancel.fm/ripcord/>
- [62] Bios-Marcel. ios-Marcel/cordless: The Discord terminal client you never knew you wanted. 2021. Available from: <https://github.com/Bios-Marcel/cordless>
- [63] Clemens, A. Discord bans me, then Discord ghosts me. 2020. Available from: <https://annaclemens.io/discord>
- [64] Schramm, M. Cordless: Add notice about closing the project. 2020. Available from: <https://github.com/Bios-Marcel/cordless>
- [65] jhgg. We are not banning people for using Ripcord. [...]. 2020. Available from: <https://news.ycombinator.com/item?id=25214777>
- [66] Holub, O. GitHub - Tyrrrz/DiscordChatExporter: Exports Discord chat logs to a file. 2022. Available from: <https://github.com/Tyrrrz/DiscordChatExporter>
- [67] nischay876. is this banable from discord ?? :(· Discussion #726 · Tyrrrz/DiscordChatExporter. 2021. Available from: <https://github.com/Tyrrrz/DiscordChatExporter/discussions/726>
- [68] Holub, O. This is on purpose actually. [...] Field name consistency with the official API · Issue #454 · Tyrrrz/DiscordChatExporter. 2020. Available from: <https://github.com/Tyrrrz/DiscordChatExporter/issues/454>
- [69] Mehrotra, P. Discord starts rolling out inline replies on desktop and mobile. 2020. Available from: <https://www.xda-developers.com/discord-rolling-out-inline-replies-desktop-mobile/>
- [70] Sanqui. Add support for replies · Pull Request #455 · Tyrrrz/DiscordChatExporter. 2020. Available from: <https://github.com/Tyrrrz/DiscordChatExporter/pull/455>
- [71] Holub, O. Add a raw json export format by rtm516 · Pull Request #553 · Tyrrrz/DiscordChatExporter. 2020. Available from: <https://github.com/Tyrrrz/DiscordChatExporter/pull/553>
- [72] Holub, O. Download list of all users for a server (online & offline) · Issue #104 · Tyrrrz/DiscordChatExporter. 2018. Available from: <https://github.com/Tyrrrz/DiscordChatExporter/issues/104>

BIBLIOGRAPHY

- [73] Holub, O. Display reaction authors · Issue #133 · Tyrrrz/DiscordChatExporter. 2019. Available from: <https://github.com/Tyrrrz/DiscordChatExporter/issues/133>
- [74] chylex. Discord History Tracker. 2022. Available from: <https://github.com/chylex/Discord-History-Tracker>
- [75] TheTechRobo, chylex. Save button data? · Issue #158 · chylex/Discord-History-Tracker. 2021. Available from: <https://github.com/chylex/Discord-History-Tracker/issues/158>
- [76] tsudoko. tsudoko/pullcord: Discord archiver. 2021. Available from: <https://github.com/tsudoko/pullcord>
- [77] tsudoko. Light mode · Issue #1 · tsudoko/pullcord. 2018. Available from: <https://github.com/tsudoko/pullcord/issues/1>
- [78] tsudoko. Too many non-fatal errors are treated as fatal · Issue #21 · tsudoko/pullcord. 2020. Available from: <https://github.com/tsudoko/pullcord/issues/21>
- [79] mraof. Thread support · Issue #28 · tsudoko/pullcord. 2021. Available from: <https://github.com/tsudoko/pullcord/issues/28>
- [80] Sanqui. Sanqui/discard: Python tool for medium-scale Discord server archival operations. 2021. Available from: <https://github.com/Sanqui/discard>
- [81] Rapptz. On Resuming discord.py Development. 2022. Available from: <https://gist.github.com/Rapptz/c4324f17a80c94776832430007ad40e6>
- [82] Rapptz. Rapptz/discord.py: An API wrapper for Discord written in Python. 2022. Available from: <https://github.com/Rapptz/discord.py>
- [83] Rapptz. discord.py documentation. 2022. Available from: <https://discordpy.readthedocs.io/en/latest/index.html>
- [84] Discord Trust & Safety Team. Automated user accounts (self-bots). 2021. Available from: <https://support.discord.com/hc/en-us/articles/115002192352-Automated-user-accounts-self-bots->
- [85] Rapptz. discord.py changelog - v1.7.0. 2021. Available from: https://discordpy.readthedocs.io/en/latest/whats_new.html#v1-7-0
- [86] dolfies. dolfies/discord.py-self: A fork of the popular discord.py for self-bots. 2022. Available from: <https://github.com/dolfies/discord.py-self>
- [87] Rapptz. The Future of discord.py. 2021. Available from: <https://gist.github.com/Rapptz/4a2f62751b9600a31a0d3c78100287f1>
- [88] Weh. Message Content Privileged Intent FAQ. 2022. Available from: <https://support-dev.discord.com/hc/en-us/articles/4404772028055>
- [89] Discord. Discord API Reference - Snowflakes. 2022. Available from: <https://discord.com/developers/docs/reference#snowflakes>

-
- [90] MolSno and commenters. Change read message history permission. 2019. Available from: <https://support.discord.com/hc/en-us/community/posts/360046946331-Change-read-message-history-permission>
- [91] u/Flippi273. What framework does Discord for Android use? 2017. Available from: https://www.reddit.com/r/discordapp/comments/76szr8/what_framework_does_discord_for_android_use/
- [92] D. Your Discord Data Package. 2022. Available from: <https://support.discord.com/hc/en-us/articles/360004957991>
- [93] Lee, T. B. Court: Violating a site’s terms of service isn’t criminal hacking. 2020. Available from: <https://arstechnica.com/tech-policy/2020/03/court-violating-a-sites-terms-of-service-isnt-criminal-hacking/>
- [94] Hofmann, M. Court: Violating Terms of Service Is Not a Crime, But Bypassing Technical Barriers Might Be. 2010. Available from: <https://www.eff.org/deeplinks/2010/07/court-violating-terms-service-not-crime-bypassing>
- [95] Vollmer, A. Standing up for developers: youtube-dl is back. 2020. Available from: <https://github.blog/2020-11-16-standing-up-for-developers-youtube-dl-is-back/>
- [96] Whittaker, Z. Web scraping is legal, US appeals court reaffirms. 2022. Available from: <https://techcrunch.com/2022/04/18/web-scraping-legal-court/>
- [97] History Associates Incorporated. Privacy: The Archivist’s Dilemma. 2020. Available from: <https://www.historyassociates.com/privacy-the-archivists-dilemma/>
- [98] Discord. DISCORD PRIVACY POLICY. 2022. Available from: <https://discord.com/privacy>
- [99] Yahweasel. Craig Records! 2022. Available from: <https://craig.chat/home/>
- [100] Discord. Discord Developer Portal - Community Resources. 2022. Available from: <https://discord.com/developers/docs/topics/community-resources>
- [101] IIPC members. The WARC Format 1.1. 2015. Available from: <https://iipc.github.io/warc-specifications/specifications/warc-format/warc-1.1/>
- [102] GitHub users. Issues · iipc/warc-specifications. 2022. Available from: <https://github.com/iipc/warc-specifications/labels/warc-format>
- [103] Melnikov, A.; Fette, I. The WebSocket Protocol. RFC 6455, Dec. 2011, doi:10.17487/RFC6455. Available from: <https://www.rfc-editor.org/info/rfc6455>
- [104] Odvarko, J.; Jain, A.; et al. HTTP Archive (HAR) format. 2012. Available from: <https://w3c.github.io/web-performance/specs/HAR/Overview.html>
- [105] Basques, K. What’s New In DevTools (Chrome 76). 2019. Available from: <https://developer.chrome.com/blog/new-in-devtools-76/>
- [106] GNU Wget. GNU Wget. 2020. Available from: <https://www.gnu.org/software/wget/>

BIBLIOGRAPHY

- [107] Archive Team wiki editors. Wget with WARC output. 2022. Available from: https://wiki.archiveteam.org/index.php/Wget_with_WARC_output
- [108] ariya. Archiving the project: suspending the development · Issue #15344 · ariya/phantomjs. 2018. Available from: <https://github.com/ariya/phantomjs/issues/15344>
- [109] Internet Archive. internetarchive/warcprox: WARC writing MITM HTTP/S proxy. 2022. Available from: <https://github.com/internetarchive/warcprox>
- [110] Mitmproxy Project. mitmproxy - an interactive HTTPS proxy. 2022. Available from: <https://mitmproxy.org/>
- [111] dufferzafar. SQLite Flow Storage Format · Issue #1029 · mitmproxy/mitmproxy. 2016. Available from: <https://github.com/mitmproxy/mitmproxy/issues/1029>
- [112] cortesi. Serialization format · Issue #3075 · mitmproxy/mitmproxy. 2022. Available from: <https://github.com/mitmproxy/mitmproxy/issues/3075>
- [113] madt1m. Shifting to Protobuf Serialization by madt1m · Pull Request #3232 · mitmproxy/mitmproxy. 2018. Available from: <https://github.com/mitmproxy/mitmproxy/pull/3232>
- [114] kryptpt. PCAP support · Issue #408 · mitmproxy/mitmproxy. 2014. Available from: <https://github.com/mitmproxy/mitmproxy/issues/408>
- [115] The Tcpdump Group. tcpdump & libpcap. 2022. Available from: <https://www.tcpdump.org/>
- [116] Chifflier, P. rusticata/pcap-parser: PCAP/PCAPNG file format parser written in pure Rust. Fast, zero-copy, safe. 2022. Available from: <https://github.com/rusticata/pcap-parser>
- [117] Collinear Group. pcap-ng-parser - npm. 2018. Available from: <https://www.npmjs.com/package/pcap-ng-parser>
- [118] Wireshark. Wireshark. 2022. Available from: <https://www.wireshark.org/>
- [119] Software Freedom Conservancy. Selenium. 2022. Available from: <https://www.selenium.dev/>
- [120] Software Freedom Conservancy. Selenium History. 2019. Available from: <https://www.selenium.dev/history/>
- [121] Google Developers. Puppeteer. 2021. Available from: <https://developers.google.com/web/tools/puppeteer>
- [122] berstend. puppeteer-extra. 2021. Available from: <https://github.com/berstend/puppeteer-extra/tree/master/packages/puppeteer-extra#puppeteer-extra--->
- [123] Microsoft. Playwright. 2022. Available from: <https://playwright.dev/>

- [124] Guo, D. Playwright vs. Puppeteer: Which should you choose? 2020. Available from: <https://blog.logrocket.com/playwright-vs-puppeteer/>
- [125] Red Hat, Inc. Understanding Linux containers. 2022. Available from: <https://www.redhat.com/en/topics/containers>
- [126] Docker Inc. Docker. 2022. Available from: <https://www.docker.com/>
- [127] Containers. Podman. 2022. Available from: <https://podman.io/>
- [128] Elasticsearch B.V. Elasticsearch: The Official Distributed Search & Analytics Engine. 2022. Available from: <https://www.elastic.co/elasticsearch/>
- [129] Elasticsearch B.V. Kibana: Explore, Visualize, Discover Data. 2022. Available from: <https://www.elastic.co/kibana/>
- [130] Wu, P.; Broman, A. tcp: add support for reassembling out-of-order segments (ca423314) · Commits · Wireshark Foundation / wireshark · GitLab. 2018. Available from: <https://gitlab.com/wireshark/wireshark/-/commit/ca423314373b0a4ce7d6bc1cf94c4995e1263ea2>
- [131] Vokál, D.; Šmahel, D.; et al. Excesivní používání internetu českými dospívajícími: Srovnání před a během pandemie Covid-19. 2021. Available from: https://irtis.muni.cz/media/3345543/excessive-internet-use-report_v13_final.pdf
- [132] Kralj Novak, P.; Smailović, J.; et al. Sentiment of emojis. *PLoS ONE*, volume 10, no. 12, 2015: p. e0144296. Available from: <http://dx.doi.org/10.1371/journal.pone.0144296>

Acronyms

AIM	AOL Instant Messenger
CI	Continuous Integration
CSS	Cascading Stylesheets
CSV	Comma separated values
DM	Direct Message
DMA	Digital Markets Act
DMCA	Digital Millennium Copyright Act
DOM	Document Object Model
EFF	Electronic Frontier Foundation
GDPR	General Data Protection Regulation
GPT	Generative Pre-trained Transformer
HAR	HTTP Archive
HTML	HyperText Markup Language
HTMLZ	HTML ZIP
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol

A. ACRONYMS

IRC Internet Relay Chat

JSON JavaScript Object Notation

JSONL JSON Lines

MITM Man in the middle

MSN Microsoft Network

OBS Open Broadcaster Software

OSINT Open-source intelligence

PCAP Packet Capture

PCAPNG PCAP Next Generation

REST Representational state transfer

SEO Search Engine Optimization

SPA Single Page Application

SSL Secure Sockets Layer

SSR Server-Side Rendering

TCP Transmission Control Protocol

TLS Transport Layer Security

TSV Tab separated values

UI User Interface

WARC Web ARChive

Supplemental Material

The complete source code of this thesis and the projects described within can be found on the attached medium.

The same content is also available online on GitHub. Versions at the submission of this thesis bear the tag `thesis`.

Thesis <https://github.com/Sanqui/discard2-thesis>

Discard2 <https://github.com/Sanqui/discard2>

```
├── README.txt ..... description of contents
├── DP_Labsky_David_2022.pdf ..... thesis text in PDF format
├── src ..... source codes
│   ├── thesis ..... source code for the thesis
│   └── discard2 ..... source code for the Discard2 project
```