

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Oskar Pikkov 212922IVSB

Developing a Telegram Scraper to Identify and Alert on Leaked Credentials Based on Pre-Defined User Input

Bachelor's thesis

Supervisors: Toomas Lepikult
PhD
Mert Meissaar
BSc

Tallinn 2024

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Oskar Pikkov 212922IVSB

**Kasutaja sisendi põhjal lekkinud
kasutajakontosid tuvastava ning neist teavitava
Telegrami ämbliku arendus**

Bakalaureusetöö

Juhendajad: Toomas Lepikult
PhD
Mert Meiessaar
BSc

Tallinn 2024

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Oskar Pikkov

13.05.2024

Abstract

The prevalence of credential leaks on the Telegram platform has been very high, fuelling the status of the app as a cybercrime hub, turning it into an ecosystem for criminals. While a big business or government entity might have the needed resources and possibilities to track credential leaks on Telegram, the less-protected entities and individuals are at much higher risk of becoming victims in the form of a credential leak being shared on Telegram.

The current solutions for tracking credential leaks on Telegram for smaller organizations or individual persons are either very costly or not working in the way needed to. Thus, the author proposes and develops a solution as developing a tool, allowing for the forementioned to start finding, processing and receiving credible intelligence about the credentials of their personal or business accounts being shared on Telegram. In the theoretical part of the thesis, a cybercrime forum is utilized to acknowledge the reader with different topics of importance to the reader. The research concludes that a solution like this can be developed, but there also exist ways of further improving it.

This thesis is written in English and is 44 pages long, including 5 chapters, 39 figures and 3 tables.

Annotatsioon

Kasutaja sisendi põhjal lekkinud kasutajakontosid tuvastava ning neist teavitava Telegrami ämbliku arendus

Lekkinud kasutajakontode levik ning sagedus suhtlusrakenduses Telegram on väga suur, mis omakorda konstanteerib asjaolu, et Telegram on muutumas küberkriminaalide keskseks kogunemispunktiks. Kuigi suurkorporatsioonidel või riiklikel asutustel võivad eksisteerida vahendid ning võimalused lekete tuvastamiseks Telegramis, ei oma sellist privileegi väiksemad ärid ja organisatsioonid või eraisikud.

Hetkel saadaolevad lahendused Telegrami kasutajakontode lekete jälgimiseks väiksematele organisatsioonidele või eraisikutele on ebanormaalselt kallid või ei lahenda probleemi. Autor pakub välja ning arendab lahenduse, võimaldades eeltoodud riskirühmadel kasutajakontode lekkimise tuvastamist, töötlemist ning teavituste saamist. Töö teoreetilises osas kasutab autor küberkurjategijate seas populaarset internetifoorumit, et tutvustada lugejale mitmeid olulisi tööga seotud teemasid. Uurimuse järelendusena tuleb välja, et sellise lahenduse valmistamine on võimalik, kuid eksisteerib mitmeid viise selle edasiarenduseks.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 44 leheküljel, 5 peatükki, 39 joonist ja 3 tabelit.

List of abbreviations and terms

APT	Advanced Persistent Threat
CIS	Commonwealth of Independent States
DB	Database
DDoS	Distributed Denial-of-Service
GB	Gigabyte
LEA	Law Enforcement Agency
MD5 hash	A cryptographic hash used to generate digital signatures or message digests.
OS	Operating System
OTR	Off-The-Record
P2P	Peer-to-Peer
PC	Personal Computer
PII	Personally Identifiable Information
RDP	Remote Desktop Protocol
Spider	A bot that harvests information from a digital system.
SQL	Structured Query Language
SSH	Secure Shell Protocol
TLD	Top Level Domain
URL	Uniform Resource Locator
WWW	World Wide Web
XMPP	Extensible Messaging and Presence Protocol

Table of contents

1 Introduction	11
2 Background.....	13
2.1 Theoretical Overview	13
2.1.1 Credential leaks and breaches	13
2.1.2 Ways of sharing credential leaks	18
2.2 Problem Statement.....	29
2.3 Available Solutions.....	30
2.3.1 Haveibeenpwned	30
2.3.2 IntelligenceX	31
3 Methodology.....	34
3.1 Data leak channels	34
3.1.1 Google for finding channels	34
3.1.2 Cybercrime forums for finding channels.....	36
3.2 Telegram scraper and alert bot	37
3.2.1 Telegram scraper	37
3.2.2 Telegram alert bot.....	38
4 Development and usage.....	39
4.1 Development of the scraper	39
4.2 Development of the alert bot	43
4.3 Usage	49
5 Summary.....	54
References	56
Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis	58

List of figures

Figure 1. Access sale subsection in the forum XSS.	14
Figure 2. An access sale post in the forum XSS.....	15
Figure 3. An example of an info stealer “log” [11].	16
Figure 4. A data-leak subsection on the forum XSS.	19
Figure 5. Various new data leak posts on the forum XSS.....	20
Figure 6. A pinned topic about locating data leaks on forum XSS.	20
Figure 7. A user on XSS asking for a download link to a data leak.....	21
Figure 8. A user on XSS responding to a data leak request with the file.	22
Figure 9. Starting an OTR-enabled chat on Adium.....	23
Figure 10. The administrator of LockBit using Tox Chat.	24
Figure 11. The admin of XSS opposing any illegal actions targeting Russia/CIS.....	26
Figure 12. A member of XSS opposing the use of Telegram on the forum.....	27
Figure 13. A member of XSS stating their opinion about Telegram.....	27
Figure 14. The simplicity of acquiring malicious information in Telegram.	28
Figure 15. Lines in the file from a public data leak.....	28
Figure 16. Haveibeenpwned notifying about breaches.	30
Figure 17. IntelligenceX displaying found results.....	32
Figure 18. An improved Google search for Telegram channels.....	35
Figure 19. A website sharing Telegram channels.....	35
Figure 20. Setting up the scraper.....	39
Figure 21. Gathering local information about already downloaded files.....	40
Figure 22. Downloading a .txt file with subsequent processing.....	41
Figure 23. Ways of processing a line.	42
Figure 24. Another way of processing a line.....	42
Figure 25. Adding the line to the database.	43
Figure 26. Setting up the bot.	43
Figure 27. The “/start” command.	44
Figure 28. Checking the correctness of the “/query” command.....	45
Figure 29. The functionality of the “/query” command.	46

Figure 30. Checking the correctness of the “/stats” command.....	47
Figure 31. Counting the lines in the DB.....	47
Figure 32. Finding top 10 most common domains.....	48
Figure 33. Finding top 10 most common passwords.....	48
Figure 34. Sending the statistics to the user.	49
Figure 35. Size of the scraped data.....	50
Figure 36. Querying by URL.....	51
Figure 37. Querying by username.	52
Figure 38. Querying by password.	53
Figure 39. Statistics for the whole DB.	53

List of tables

Table 1. Explanation of the contents of an info stealer “log” [11]......	17
Table 2. Analysis of the selected channels found from Google.	36
Table 3. Analysis of the selected channels found from XSS.....	36

1 Introduction

As digitization is rapidly improving and changing our world, cyber security has become a household name for many of us. Long gone are the days of it being a niche topic of information technology, having made its way to being one of the hottest subjects in the modern computerized world.

While the popularity of cyber security might be seen as a positive outcome for most, it could also be explained by the rapid rise of different forms of cyber offenses, increasing number of criminals operating solely behind a screen and changing possibilities to conduct harm in the cyberspace. This is also illustrated by the fact that the annual cost of cybercrime is predicted to reach up to 9.5 trillion US dollars in 2024 [1].

The fast-paced and seemingly limitless landscape of the cyberspace allows adversaries to conduct various types of criminal activities, ranging from DDoS attacks to transnational espionage operations. Although complex APT-intrusions or major service downtimes cause huge problems to the service owners, they do not possess a tangible effect on the average computer user. This is where data breaches and credential leaks come into play, having the possibility to affect almost everyone using a digital system.

An uneducated computer user entering their social media credentials to a cloned version of the site or a network administrator saving sensitive passwords on their system getting infected by info stealer malware might sound as two totally different events with unlike scopes, outcomes and approaches. But when analysing such episodes on a higher-level basis, one can easily conduct that illegally obtaining and reusing foreign credentials is the main objective of the threat actor in control of the operation. Various attacks like these are part of the fact that data breaches were at an all-time high in the United States in 2023. [2].

The publicity and popularity of data leaks wouldn't be possible without a form of sharing them by the illegal acquirers. The development of the worldwide cybercrime environment has seen a vast growth in miscellaneous forums, messengers and other ways of communication and expression between the participants.

There exist many forms for messaging between the cyber criminals. One of the most prevalent ways is using Telegram, an accessible and free messaging app, that even many of normal users are familiar with. In some ways, Telegram has transformed itself from a privacy and security focused messaging app to a hub for cybercrime, allowing to easily host, share and obtain illegal content, communicate with other criminals and acquire various illicit gains from it [3].

2 Background

This section acts as a base to the reader, giving a strong theoretical summary of the main topics used and discussed about in this thesis, the problem statement, currently available solutions on the market and the expected results the author tries to achieve with his work.

2.1 Theoretical Overview

This subsection aims to acknowledge the reader about the tools, products and topics mentioned or used in this thesis. While the main work is focused on developing a scraper and alert bot for the Telegram application, it is vital to get accustomed to different aspects, history and characteristics associated with developing a tool like this. For this subsection, a diverse collection of internet resources is used, combined with the author accessing an underground cybercrime forum called XSS (an account is required for the access), to get a more detailed overview of the role and the information present on such forums about this topic.

2.1.1 Credential leaks and breaches

Obtaining various digital information has been on the forefront of illicit cyber activities since the first hackers starting to take place on the Internet. As data leaks have been a hot topic and in some ways a buzzword for the larger public, the exact meaning needs to be defined for a better grasp on the scope of this thesis.

In this paper, the author uses various words such as (data) “leaks” and “breaches”, which should be interpreted as illegally obtained digital credentials, allowing to access any kind of a digital system, application or website without the knowledge or permission of the owner of these credentials.

While the topic of credential leaks might be easy to grasp for most of the readers as a collection of illegally obtained digital accesses, it is mandatory to demonstrate some popular ways of gathering them by threat actors.

2.1.1.1 Gaining entry to a sensitive system to steal credentials

One of the easiest ways for hackers to steal large amounts of credentials is to gain an entry into the system(s) of an organization or company possessing the needed data. The main option for an adversary is to buy the access outright, from various private sellers or cybercrime forum users.

For example, the product might be sold as a backdoor already implemented to the compromised system by the seller or legitimate RDP access allowed to an employee of the organization or company [4].

System accesses are a common article for sale and one can easily find auctions or advertisements dealing with them on the dark web. Take for example XSS (founded in 2004 under the name “DaMaGeLaB”) [5], a predominantly Russian-speaking cybercrime community, where there is a publicly accessible subsection for these kinds of purchases (displayed on Figure 1). There exist many similar topics with various accesses for sale, allowing criminals to start their operation with an already compromised system.




 Куплю ваши доступы Kindzadza · 15.01.2024	Ответы: 9 Просмотры: 1K	Сегодня в 02:33 Kindzadza
 Selling Corp VPN Access MX KUPWPER · Вчера в 00:01	Ответы: 1 Просмотры: 129	Вчера в 18:07 KUPWPER
 Selling access 13334 · 01.02.2024	Ответы: 14 Просмотры: 1K	Вчера в 17:36 13334
 Corp Sale DOC · 10.02.2024	Ответы: 20 Просмотры: 2K	Вчера в 17:01 PlayingMantis

Figure 1. Access sale subsection in the forum XSS.

The posts themselves (see Figure 2) contain diverse information about the accesses, such as the country of the organization or company attacked, their business revenue (used mainly by ransomware operators to determine the ransom payment size), business sector and type of access (such as a web shell, RDP, SSH credentials etc.). Often the privileges of the compromised user are mentioned in the post, such as Domain Admin, Domain User or Administrator on Windows OS’s or the availability of root access on Linux machines.

Figure 2. An access sale post in the forum XSS.

As the cybercrime ecosystem on these forums operates mostly as a mechanism where financial gain is the main motivation of the perpetrators, the pricing of these products is another factor to consider analysing the value of them. The cheaper forms start from a couple of hundred US dollars, usually with low privileges and a small (by revenue) organization or company. Bigger accesses have seen to be sold for tens of thousands of US dollars, usually with very strong privileges of the compromised user, used to access a large and popular company with a big revenue (hundreds of millions of US dollars) [6].

Another way of entering the network of an unsuspecting entity is to utilize a publicly available application of the organization with malicious actions. It can be done by means of a misconfiguration by the developer(s), a software bug or by using a powerful exploit accustomed for the exact technology present on the app [7]. One of the most trivial, but still popular ways of abusing the configurational or developmental errors by the developer(s) is to conduct a SQL Injection attack, allowing the intruder to enter as an administrator or any other high-privileged user, usually by obtaining their credentials with the intrusion [8].

The third method might be to socially engineer the user to willingly give their credentials to the attacker(s) by conducting an operation targeting them, known as phishing. The threat actor usually delivers a message from a compromised or specially crafted source via e-mail or social media, instructing the unsuspecting user to share their login and password details, which end up in the hands of the malicious attacker [9].

2.1.1.2 Using info stealer malware to gather credentials

The usage of malware has been continuously prevalent in the last decade, with the year of 2022 containing over 5.5 billion different malware attacks [10]. Info stealer malware is a type of malicious software, used to identify, obtain and exfiltrate data from a compromised computer system. While there are various stealers for sale in the cybercrime ecosystem, the general structure of them remains the same. As the data from the compromised system is stolen for further human use, it must be packaged and organized in a comprehensible way for the later user of it. A typical “log”, the package with the information stolen from one unique computer, uses an understandable format, categorizing different data files into different directories, as demonstrated on Figure 3 [11].

```
**|Stealer Log**/  
├─ Autofills/  
│   ├─ Google_[Chrome]_Default.txt  
│   ├─ Google_[Chrome]_Profile1.txt  
│   └─ Microsoft_[Edge]_Default.txt  
├─ Cookies/  
│   ├─ Google_[Chrome]_Default Extension.txt  
│   ├─ Google_[Chrome]_Default Network.txt  
│   ├─ Google_[Chrome]_Profile 1 Network.txt  
│   ├─ Microsoft_[Edge]_Default Network.txt  
│   ├─ Microsoft_[Edge]_Profile 1 Network.txt  
│   └─ Opera Software_Unknown Network.txt  
├─ CreditCards/  
│   └─ Microsoft_[Edge]_Default.txt  
├─ FileGrabber/  
│   └─ Users/  
│       └─ Pauli/  
│           └─ Desktop/  
│               └─ passwords.txt  
├─ DomainDetects.txt  
├─ ImportantAutofills.txt  
├─ InstalledBrowsers.txt  
├─ InstalledSoftware.txt  
├─ Passwords.txt  
├─ ProcessList.txt  
├─ Screenshot.jpg  
└─ UserInformation.txt
```

Figure 3. An example of an info stealer “log” [11].

As displayed above, an info stealer can obtain various data from a compromised system. The author feels the need to further explain the files shown in Figure 3, to give the reader a better understanding of the capabilities of the info stealer malware. These explanations are available in Table 1.

Table 1. Explanation of the contents of an info stealer “log” [11].

Folder or file name	Content
Autofills/	Autofill data from various browsers. Contains e-mail addresses, phone numbers, physical addresses, PII etc.
Cookies/	Cookie files from various browsers. Used to access accounts and services without the need for logging in.
CreditCards/	Credit card numbers from various browsers. Used for financial gain by the malware spreaders.
FileGrabber/	Files from various directories from the compromised computer. Used to find additional information (such as passwords saved to the Desktop), that cannot be accessed solely from the browser files.
DomainDetects.txt	Domains, that the system has accessed or interacted with. Allows the operator to quickly identify if the info stealer log is valuable to them or not (i.e. the domains paypal.com and swedbank.ee could indicate that the owner of the computer stores banking information on their PC).
ImportantAutofills.txt	Autofill data as in the first folder but filtered by importance. Contains the most sought-after information, such as addresses and phone numbers.
InstalledBrowsers.txt	List of browsers present on the victim machine.
InstalledSoftware.txt	List of software present on the victim machine.
Passwords.txt	List of domain-username-password combinations collected from browsers on the victim machine. Used to access various services and accounts, often combined with a cookie file in the Cookies/ folder, for maximum recognition as the owner of the

Folder or file name	Content
	account.
ProcessList.txt	List of processes running on the victim machine at the point of infection.
Screenshot.jpg	A screenshot taken by the info stealer malware at the point of infection. Used to gain a better understanding of the victim machine, showing saved bookmarks, desktop background or notes etc.
UserInformation.txt	A file containing most crucial information about the victim system. Usually contains system name, time zone, IP, infection date and time, hardware information etc.

While one info stealer malware log contains very valuable information to conduct identity theft, financial fraud or other illegal actions, it is not enough to be categorised as a significant credential leak or breach. That's where "Cloud of Logs"-type services come into play. These are huge datasets of hundreds of thousands of info stealer logs, where access is sold on a time-based (e.g. one week or one month) basis. It allows cyber criminals to gain access to an enormous amount of stolen credentials and other crucial digital information [12]. These services are then used by the criminals to gather colossal amounts of Passwords.txt files, filter them and combine into big text files, for further sharing. These final collections of credentials make up the vast amount of credential leaks sourced from many small info stealer malware infections all over the world.

2.1.2 Ways of sharing credential leaks

Without the credentials being leaked, we couldn't consider them as released for the bigger audience, resulting in different types of malicious actions, such as but not limited to [13]:

- Identity theft
- Blackmail or catfishing
- Social engineering operations
- Trade secret compromises

- Espionage/state surveillance

A simple file transfer containing some usernames and passwords between two adversaries could be considered as a shared credential breach, but in the scope of this thesis, the author defines “sharing” as publishing a large set of credentials to an external audience for free digital access. The following subsections showcase some of the ways credentials are distributed by the threat actors.

2.1.2.1 Forum posts

Since the first appearances of popular cybercrime forums in the 2000s [14], they have served as handy platforms for sharing various data and credential leaks. While there exist multiple of such communication boards, they are quite similar in their content. For the purpose of displaying some of the parts of a service of this kind, the cybercrime forum “XSS” is used, just like in section 2.1.1.1.

Forums feature special subsections for sharing illegally acquired data, allowing users to discuss, obtain and guide others on this topic. As seen on Figure 4, the forum XSS hosts a subsection solely for this, called Bases (i.e. databases, “*Базы*” in Russian). As of April 2024, the subdivision has over 4600 unique topics (“*Темы*” in Russian) and over 32 100 unique messages by users (“*Сообщения*” in Russian).

БИБЛИОТЕКА			
Новости	Темы 6.4K	Сообщения 28.8K	H0 В Самарской области осужден раз... Вчера в 14:58 · H0mT
Бэгтрек	Темы 821	Сообщения 2.5K	telegram rce poc Только что · sed
Статьи Мануалы / Книги Видеоматериалы	Темы 2K	Сообщения 20.4K	Exodus Phishing Page Фиш... Вчера в 16:58 · TheProlg34
Базы	Темы 4.6K	Сообщения 32.1K	Да США 2024 Сегодня в 06:44 · DataFor
Софт	Темы 1.7K	Сообщения 16.4K	Mail.ru spammer (bas) Вчера в 08:09 · che

Figure 4. A data-leak subsection on the forum XSS.

The subsection is quite active, with daily (see Figure 5) new posts about data or credential leaks. This is indicated by the publication date, shown on the right-hand side of the screenshot. The two latest posts are added on the date of capturing the screen for this thesis (“*Сегодня*” in Russian).










































 CUJA 2024 DataFor - Сегодня в 06:44	Ответы: 0 Просмотры: 16	Сегодня в 06:44 DataFor
 Forto.com / FreightHub.com nightly - 09.03.2024	Ответы: 3 Просмотры: 354	Сегодня в 06:30 nightly
 Privat fresh Mail Access Mix 14.04 Mailcrafter - Вчера в 07:33	Ответы: 0 Просмотры: 27	Вчера в 07:33 Mailcrafter
 So Fresh Check Now!!! 14.04 Mailcrafter - Суббота в 05:53	Ответы: 0 Просмотры: 97	Суббота в 05:53 Mailcrafter
 Hotmail Fresh 13.04                                    		



Figure 7. A user on XSS asking for a download link to a data leak.

Upon reviewing the user's request, another forum member responds with the file asked for in Figure 7. As seen on Figure 8, he has included two download links for the same file (redacted by author), in case of a file hosting service removing one of them, to keep the file available for other members to access and download. He adds that "There isn't anything more available on the web [about this data leak]" (*"Больше ничего публичного в сети нет"* in Russian). As a sign of thanking the replier, the requester acknowledges the response with a green thumbs up icon, adding to the reputation count of that profile.

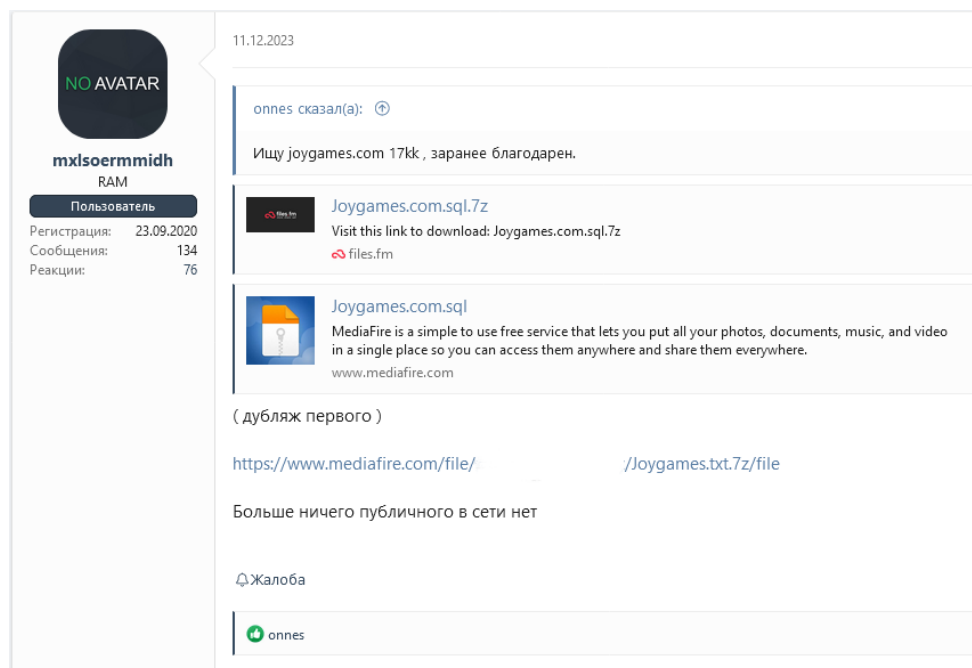


Figure 8. A user on XSS responding to a data leak request with the file.

Friendly sharing of data breaches as displayed above in such search-topics and other separate postings about data leaks make up a great quantity of the data leak ecosystem, as such forums are still dominant in 2024, with new users signing up every day.

2.1.2.2 Messaging solutions

There's no doubt that the popularity of messaging applications has also carried over to cyber criminals, who similarly to normal users, need to communicate in a fast and secure way. Although many of online communication solutions are available even for cyber criminals, they tend to shift towards the securest solutions, giving them peace of mind either about the encryption, security or logging policies (the promise of not saving user actions and/or data by the messaging application provider) of that service. In this section, some of the most prevalent communication solutions for cyber criminals are analysed.

One of the oldest ways of exchanging information for cyber criminals is Jabber (also known as XMPP). It has dominated the Russian-speaking cybercrime landscape, offering them to host their own independent servers for accounts [16]. Combined with the plugin called Off-the-Record Messaging [17], allowing the participants in a XMPP chat to be sure who they are really talking to, makes the Jabber and OTR combination

quite powerful in terms of security and privacy. There also exist some Jabber clients with built-in OTR functionality, such as Adium [18] (see Figure 9).

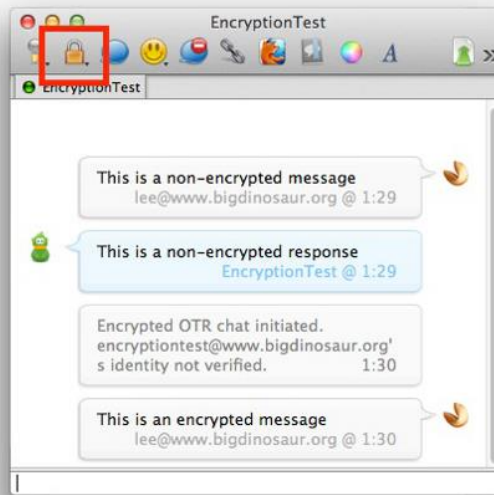


Figure 9. Starting an OTR-enabled chat on Adium.

Jabber is a great tool for secure communication but lacks the needed functions to allow large groups of members to connect to a certain channel to share various data leaks. There is no common way of hosting files on a Jabber server, thus not allowing for users to easily access them. There is also some technical knowledge needed to set up and use Jabber, which has some functions that younger cyber criminals might not be used to (i.e. creating a Jabber account on a server used by others for communicating about such topics). It's obvious that an XMPP-style way of communication is not suitable for comfortably sharing data leaks between many users.

Another tool for communicating in this sphere is called Tox Chat. It is a fully P2P way of exchanging messages, without the need of a centralized server for relaying data. All the exchanges between two users are sent directly and, in some cases, a Tox bootstrap node might be used by the messaging client [19]. The phenomenon of Tox has risen together with the popularity of ransomware operators in the last decade, thus making it often used by more sophisticated cyber criminals [20].

Each Tox user has a unique identifier (Tox ID) [19], a long string of letters and numbers, identifying their account from millions of others. This is the "username" of a

Tox account and is utilized to add other users of Tox to the friends list. As seen on Figure 10 [21], one of the strongest and most sophisticated ransomware groups, LockBit, has seen the owner of it operate on Tox Chat for years [21]. Although LockBit's operations and infrastructure has been dismantled for multiple times, the original administrator remains at large, successfully avoiding LEA [22].



Figure 10. The administrator of LockBit using Tox Chat.

As we can see, Tox is a secure and popular solution for cyber criminals. Although popular in one-to-one conversations, it lacks the traction to be used widely for group chats. Just as Jabber, the rise of use of Tox has been mostly among more sophisticated criminals, with the needed technical and operational knowledge. The bigger part of the cybercrime ecosystem is still to widely accept and promote Tox as the standard for communicating, thus still leaving it in the area of niche-messengers in the year of 2024. There currently is no broad use of Tox for sharing data leaks to a larger audience.

The third option for communicating and sharing data leaks discussed in this thesis is Telegram. The origins of Telegram date back to 2013, when the founders of VKontakte (Russia's Facebook), Pavel and Nikolai Durov, were pressured to sell their shares of it, related to ignoring the wishes of the Russian government to start censorship of protests

on the platform. They left Russia and founded Telegram, aiming to provide the world with a communication solution without any government interference [23].

After 10 years of activity, it reached over 800 million active users in 2023, solely on word-of-mouth marketing [24]. It has transformed itself from a largely unknown messenger in its early days to a behemoth of communication, accepted as the way of transmitting messages in almost any free country.

While Telegram has had its commercial success, it has also gained the attention of cyber criminals. It has always allowed to register an account without disclosing any PII, just a phone number for account confirmation. Thus, a cybercriminal using an SMS-receiving service, can easily rent a virtual phone number and receive the confirmation code on it, thus validating their Telegram account and gaining access to all of the functions. Also, there are hardly any limitations on creating multiple accounts, allowing the malicious users to discard old accounts and create new ones rapidly, somewhat disrupting LEA efforts to profile them in the long run [3].

Telegram is also very confident in their data privacy claims. According to them, they have not released any user information to LEA, claiming to notify the user base on a special channel, if it happens. Although their privacy policy claims to only disclose information about terrorists and child abusers, there has yet to exist concrete proof of them doing so. A neat feature about Telegram is the ability to conduct encrypted one-to-one chats (called Secret Chats), enabling only the sender and the receiver to read the messages. Normal chats and channels are not protected in such a way, but the long track record of safeguarding user information and not storing much data about them, has still made Telegram the most popular messaging app for cyber criminals [3].

Even though Telegram has secured its place in the cybercrime world, it has met a lot of resistance from the “old school” hardliners, who find it unimaginable to use such an application, pointing to the uncertainty about data collection and privacy claims. For example, many older and respected members of the Russian-speaking cybercrime community are very opposed to the shift of the community in switching their communications to such a platform. On Figure 11, a topic started by the long-time administrator of the cybercrime forum XSS, states that the whole forum is against any form of cybercrime targeting Russia and/or CIS-countries (*“Мы, Дамага, против*

работы по РУ/СНГ” in Russian). The admin also included a banner that the members could add to the signatures of their future posts, indicating their support of the idea. There exist many reasons for such a mindset in the Russian-speaking cybercrime community, but the main reasons for this claim are patriotism (i.e. not causing harm to their homeland) and freedom of the criminals themselves (i.e. most likely transnational requests for extradition of Russian nationals accused of cybercrime are ignored by the Russian government, but for illegal actions internally, they would surely be prosecuted).

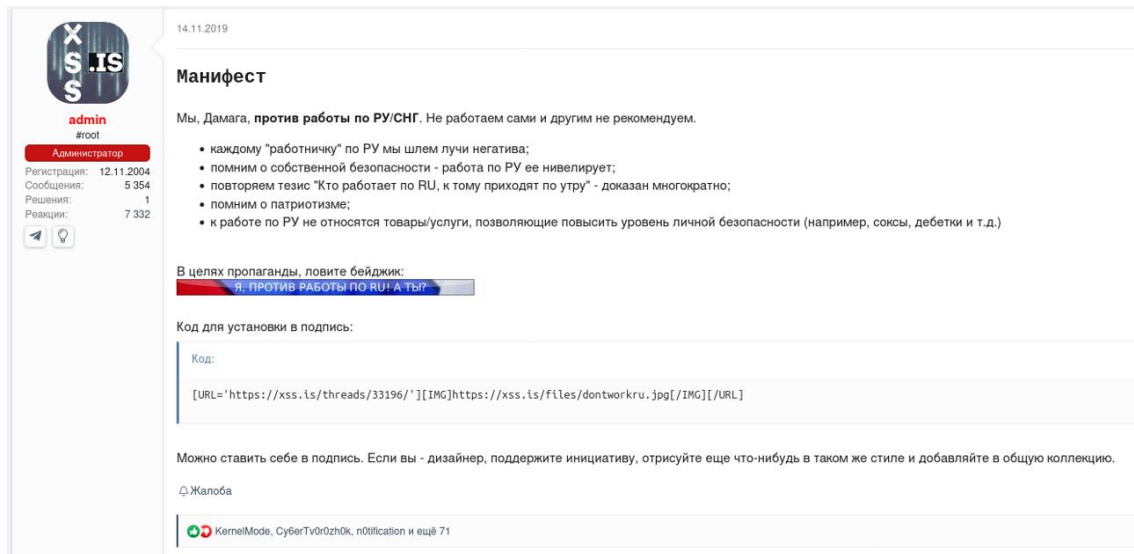


Figure 11. The admin of XSS opposing any illegal actions targeting Russia/CIS.

While this statement was accepted by the community, it somehow attracted many opinions regarding the rise of Telegram as a messaging application for “work” (full-time cyber criminals regard their actions as work and quite often compare it in some ways to a normal office job). As seen on Figure 12, the first answer in this thread asked for a banner for opposing the use of Telegram (“Отписуйте что нибудь похожее, только с текстом «Я против Телеграм»” in Russian). It received only positive reactions from other members.

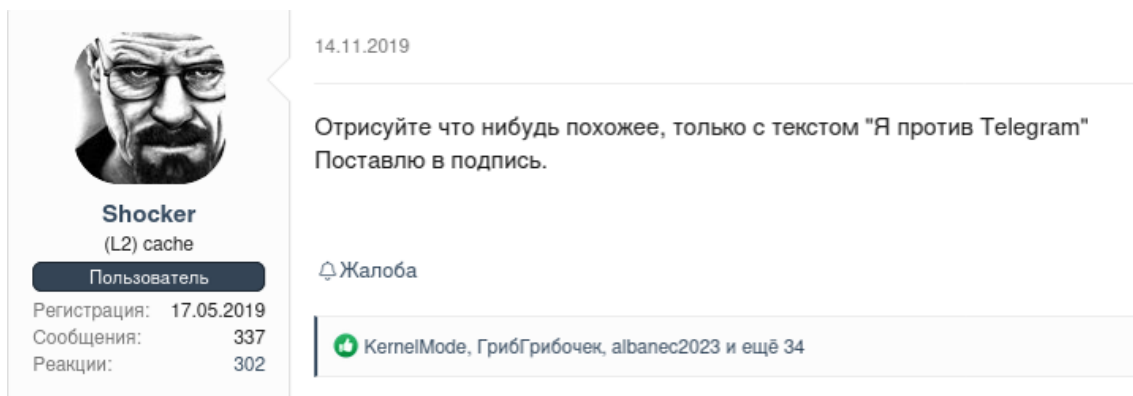


Figure 12. A member of XSS opposing the use of Telegram on the forum.

Soon, a second message (see Figure 13) appeared in the thread. Again, the use of Telegram was questioned, this time more vulgarly, stating that only an “idiot or a cop” would use Telegram for cybercrime (*“В телеграме дела вести будет либо откровенный идиот, либо коп”* in Russian), implying the lack of trust in the privacy of Telegram.

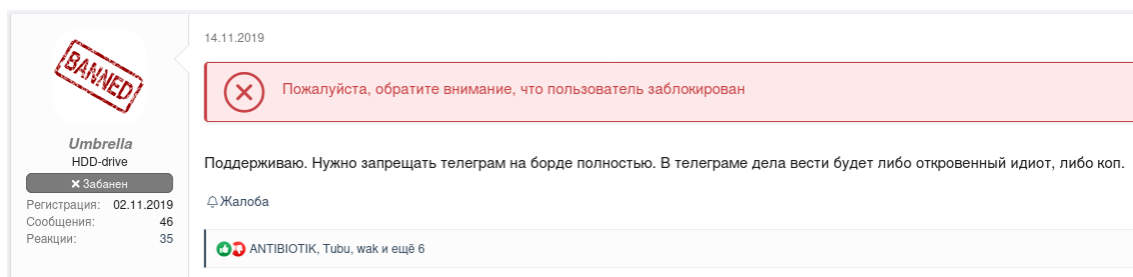


Figure 13. A member of XSS stating their opinion about Telegram.

There are many such cases of opposing Telegram in the cybercrime ecosystem, but nonetheless, it has secured its place as the top application for communication between cyber criminals. As easy as it is to install Telegram and to become a member, one can quickly access various illegal content, such as prohibited items, cybercrime chats, sale of PII and data leaks. To demonstrate this, the author opened Telegram, registered a dummy account, and searched the phrase “data leaks” in Telegram. The second channel in the search contained many files with Facebook information (see Figure 14). It took less than 30 seconds to locate a file with the name “Estonia.zip”, hinting that the content of it might be associated with data related to Estonia.

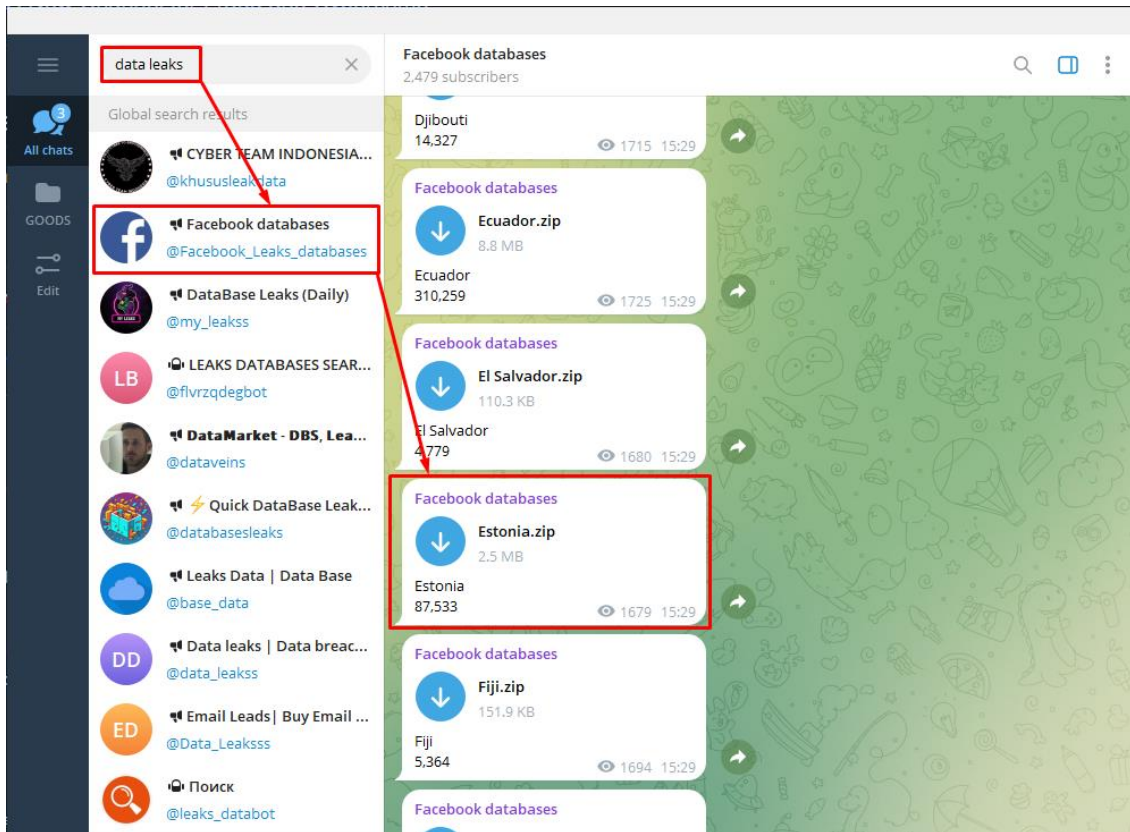


Figure 14. The simplicity of acquiring malicious information in Telegram.

Upon finding such a file, the author downloaded it and opened it in a secure environment, as all of such files should always be treated as malicious. The text file contained over 87 thousand different Facebook users from Estonia. A unique line in the text file had the user's phone number, account ID, first and last name, gender, home and work information and often an email (see Figure 15 for an example, sensitive information is blurred by the author). Subsequently, the author verified the authenticity of the leak by finding his contacts known to him in it with all of their information being valid.

Line	Phone Number	Account ID	First Name	Last Name	Gender	Home	Work	Email
139	3725	6:1000	Jan	Tartu	male	Tartu, Estonia	Jõgeva	In a relationship::11/12/2018 12:00:00 AM::
140	3725	5:1000	Andrei	Tallinn	male	Tallinn, Estonia	Pärnu	Married:::12/2/2018 12:00:00 AM::
141	3725	5:1000	Jelena	Tallinn	female	Tallinn, Estonia	Tallinn	Estonia:::3/23/2019 12:00:00 AM::
142	3725	2:1000	Kristel	Pärnu	female	Pärnu	Pärnu	:::4/28/2019 12:00:00 AM::
143	3725	1:1000	Glen Kerdo	Rapla	male	Rapla	Rapla	:::8/25/2018 12:00:00 AM::

Figure 15. Lines in the file from a public data leak.

Taking note of the most recent dates present in the lines, the author dated the leak to sometime in the summer of 2019. Using a Google search, he was able to determine that

a vulnerability was exploited to scrape the data of over 533 million Facebook users in 106 countries in August of 2019 [25]. An example like this perfectly illustrates the capability of Telegram, both in allowing such data to be present and for users to quickly interact with it. Thus, we can conclude that Telegram is a great platform for sharing data leaks and must be taken into scope to better mitigate various risks that are prevalent in the case of digital credentials being shared with other cyber criminals.

2.2 Problem Statement

So far, the author has given the reader a basic overview of the cybercrime ecosystem and the use of messaging applications and their role in it. The author has also shown different ways how data or credential leaks might occur and on what platforms they might be shared between various cyber criminals. Telegram has been identified as a popular way of distributing stolen credentials and PII, with various sensitive information being accessible from the search bar of the application. Obviously, finding and gathering special data leak channels will support, if not amplify such a statement, making Telegram an even more important target for gaining intelligence about the past, present and future credential leaks. While leaked credentials might be in the safety focus of large corporations and government entities, the average citizen or small business might not have the time, resources and funds to track, distinguish and filter their digital credentials being shared on Telegram.

The solution to this massive-scale data leak sharing on the Telegram messenger is to create an application that could automatically detect, download, filter and notify the operator of the application of pre-set credentials (i.e. username, password or URL) being leaked and/or shared on Telegram, without any significant financial cost to the operator of the app.

As a result, a digital program shall be created, that allows anyone with basic computer knowledge to start detecting and locating various credentials of their interest being shared on Telegram. The solution must be free to use and only require a dummy Telegram account for its successful utilization. This would give less tech-savvy Internet users, persons who value their time, small business or organization representatives and many others the possibility to increase their digital safety and security by being more aware of their credentials circulating on the Telegram platform.

2.3 Available Solutions

There exist many different solutions, free and paid, that in some way help the user or organization to identify leaked credentials. Although different in pricing, functionality, history and capabilities, the main function remains quite the same. This subsection acknowledges the reader with some available services, which might help to detect credential leaks.

2.3.1 Haveibeenpwned

Haveibeenpwned (accessible at haveibeenpwned.com) is a great tool for individuals to identify if their e-mail address has appeared in credential breach(es). First launched in 2013 as a small project, it currently hosts over 730 credential leaks and over 13 billion leaked accounts. It allows the user to see the largest breaches by size and search for sites and services that were compromised with the data ending up on Haveibeenpwned [26].

The search function present on the website can be used to quickly get an overview of the different leaks the user's email has been present in, without any cost (see Figure 16, with a search conducted for a very popular e-mail address).

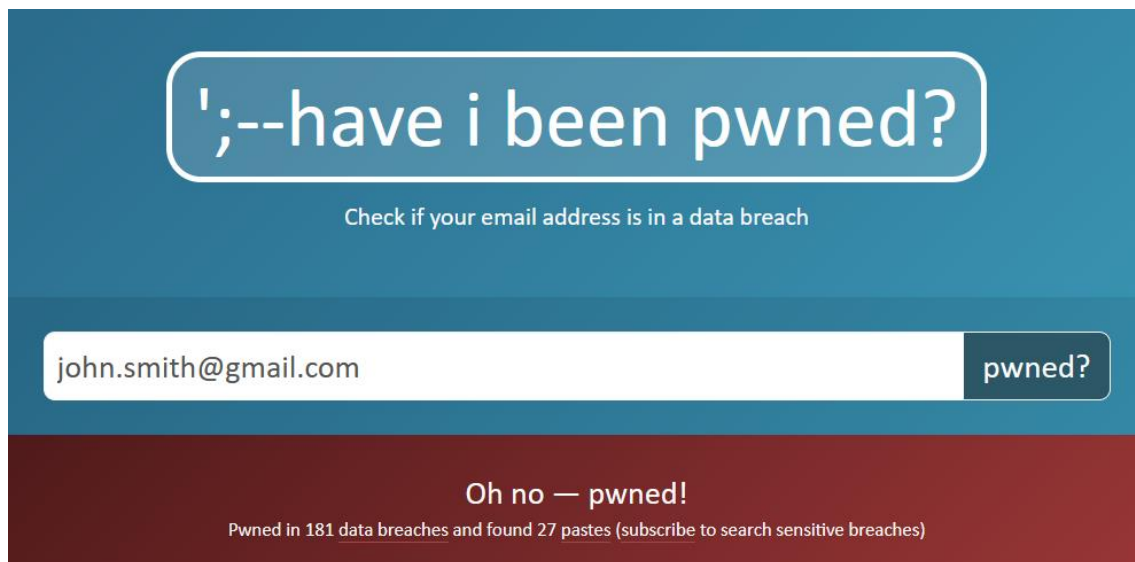


Figure 16. Haveibeenpwned notifying about breaches.

Although free and easy to use, it has doesn't solve the problem stated in subsection 2.2 of this paper. The author has found the following shortcomings in the functionality of

this credential leak service, which make it quite weak in terms of operating in the scope of the issue explained in this thesis:

- There is no possibility to search for a domain and get all of the leaked credentials associated with it.
- The search result(s) are just the leaks with some basic information. There are no passwords included.
- It does not incorporate smaller leaks, and if so, they need to be vetted, filtered and approved for use by the website administrator.
- It does not conduct active collection of data leaks in Telegram.

2.3.2 IntelligenceX

IntelligenceX (accessible at intelx.io), is another tool for finding various leaked credentials, data breaches and info stealer logs gathered from the dark web. It is quite powerful, allowing the user to search for domains, URLs, IP-addresses, cryptocurrency addresses etc. Founded in 2018 in Prague, it has defined its target customers as companies of any size and governments [27].

IntelligenceX allows free searches, but they are quite useless in terms of actually gathering credentials leaked by an adversary. Almost all of the results are fully obfuscated (see Figure 17 for a search with a very popular e-mail address), prompting the user to acquire a license (2500 euros per year for the cheapest, 20 000 euros per year for the most expensive one) [28].

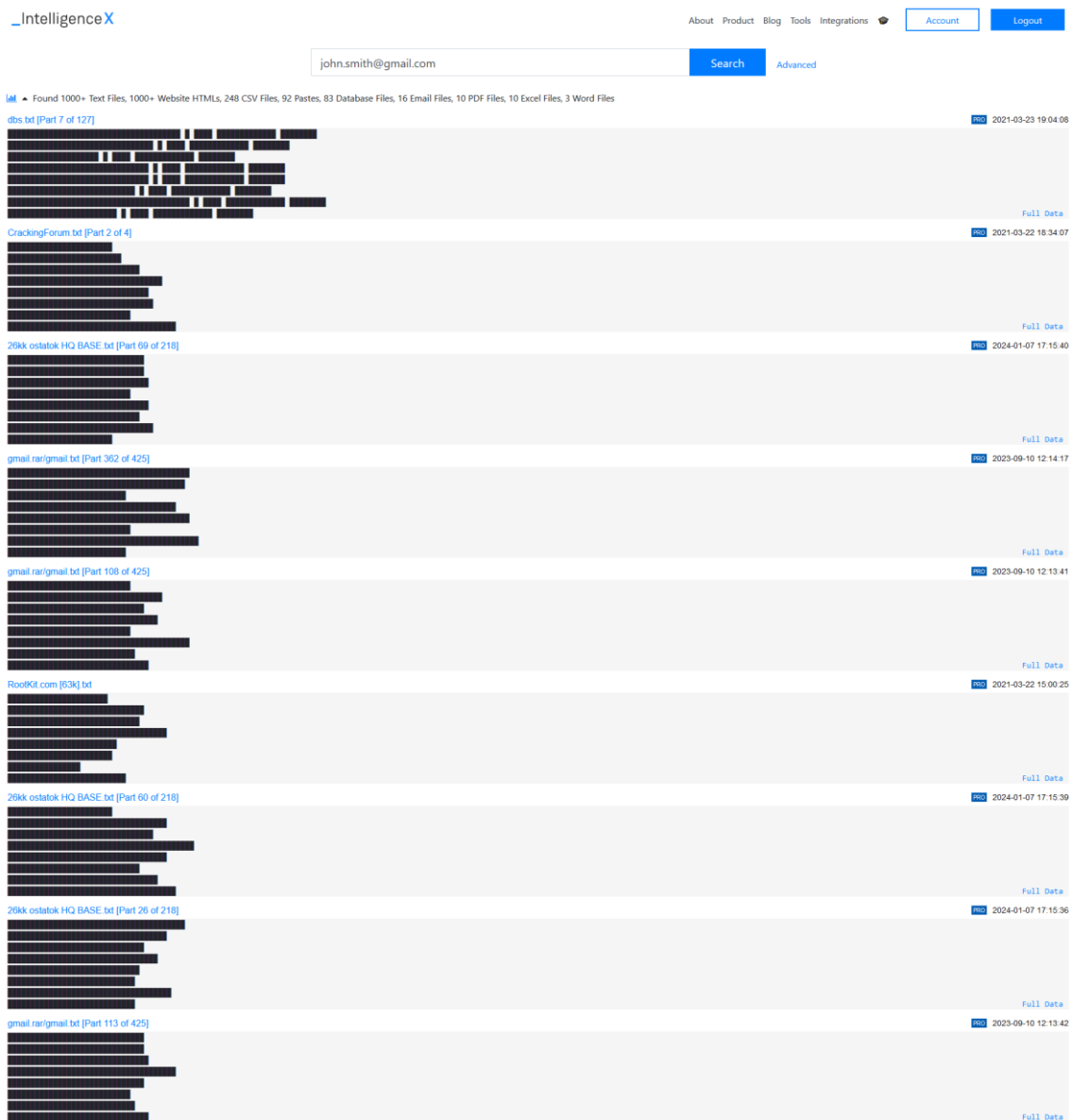


Figure 17. IntelligenceX displaying found results.

As with Haveibeenpwned, IntelligenceX has many issues to be classified as a superior alternative to the solution proposed in this paper. Some of the shortcomings are:

- The product is unusable without a pricey license.
- It does not conduct active collection of data leaks in Telegram.
- The leaks are displayed as full files, the exact account searched for is never available separated from the others (the user has to collect and export the needed lines manually to a better format).

Certainly, many other alternatives exist for the proposed solution proposed by the author. But they are not popular and easy enough to use for the average PC user, as required in the Problem Statement subsection. Also, most of them require some means of payment for usage, either as a one-time lump sum or a monthly or yearly license fee. The need to create a working solution to the problem defined beforehand is crucial, to ensure the timely detection, identification and alerting of credential leaks in Telegram, without any real financial cost to the user of it.

3 Methodology

The main objective of this research is to develop a working solution to the issue described in the Problem Statement subsection. The author aims to give insight into the ways of researching information about Telegram channels used to spread credential leaks, acquiring access to them with a dummy account and the collection of tools, packages and code used for building the working application for scraping the channels identified in the research step.

3.1 Data leak channels

The topic of data leak channels is quite interesting. While it was demonstrated in the end of section 2.1.2.2 that credential leaks are easily found on Telegram, it still takes some effort and research to locate the channels with almost every-day sharing of credentials, that provide fresher data and more a more active community. For this, the author utilizes two ways of approaching the problem. First, he uses Google and conducts different searches to find Telegram channels, which do not appear in the application's first search results. Secondly, he leverages the beforementioned cybercrime forum XSS, to gather more data leak channels. For the purpose of this paper, two channels are found through the Google search engine and two other channels are located on the forum XSS. As the number of files present on an average data leak channel is quite large, four different channels are enough for demonstrating the needed functionality of the solution for scraping and alerting of data found on these channels.

3.1.1 Google for finding channels

Google is a great resource, as it has the capability of indexing almost anything available on the WWW. For the purpose of finding Telegram credential leak channels, the keywords “telegram credential leaks channels list” are used. The author adds a special tag “intext: t.me/” to the query, indicating the need to only see the results with the prefix of a Telegram channel link in it. This is known as a Google dork, a way of telling the search engine exactly what it should return to the user. While useful for locating sensitive information, such as open web directories, password files and various

misconfigurations, one can still leverage it to improve their usual searches [29]. As seen on Figure 18, the search only displays 9 results, with the first one being accessed by the author on Figure 19.

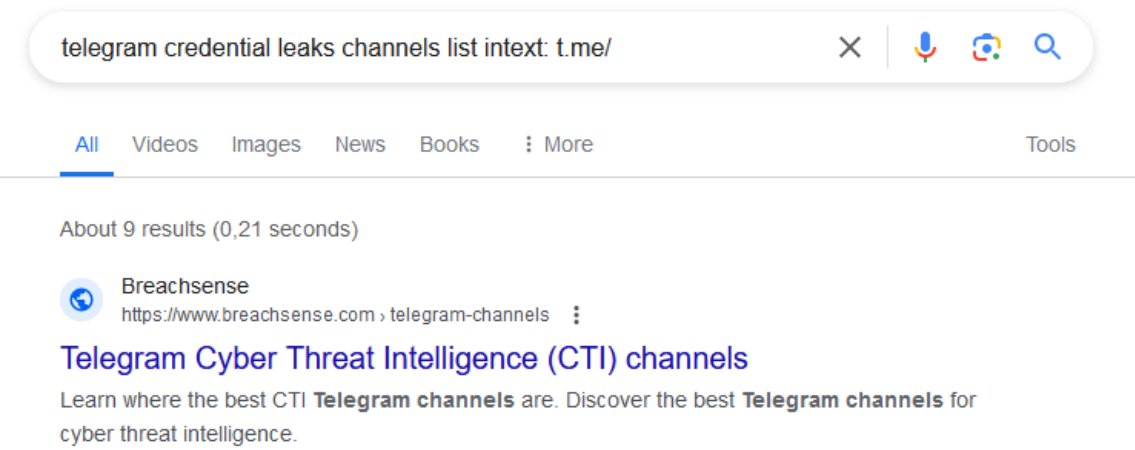


Figure 18. An improved Google search for Telegram channels.

BREACHSENSE

CTI Telegram Channels	
Last updated: April 16, 2024	
Name	Channel
Name	Telegram
GODELESS CLOUD Botnet Logs	https://t.me/+8DxOrHQdrzwIZjUy
HUBHEAD Logs	https://t.me/+fcxhFI9JSRE3YTdi
Luffich Logs - Redline Stealer	https://t.me/+NshXICbUEZkxZDMY
Goblin's Free Logs	https://t.me/+OzheKtZ368YxMDBI
Log Leaks Group	https://t.me/+V_oM-vx0YnSN7nzH
Bank Logs	https://t.me/banklogplug2
Redline Stealer	https://t.me/berserklogs
Redline Data Leaks	https://t.me/BorwitaFreeLogs
Redline and Raccoon Data Logs	https://t.me/bradmax_cloud
Log Leaks Channel	https://t.me/cbanke_logs
UnixSeller89 Redline Stealer	https://t.me/CloudLogsPrivate
Redline LogZone	https://t.me/cloudlogs

Figure 19. A website sharing Telegram channels.

This proves the ability to access data leak channels even from the most popular [30] search engine. As two channels have to be used from Google for this experiment, they are hand-picked from the website by the author and analysed in Table 2.

Table 2. Analysis of the selected channels found from Google.

Channel name	Author's description	Activity
DNFTM Cloud	A channel intended to publish maliciously obtained data. The admin often shares credential leaks, Cloud-of-Logs info stealer logs obtained from other services and data leaks that have happened recently. Advertises to the members about the possibility to gain access to a “private” collection of credential leaks in exchange for payment. Hosting over 1300 files as of April 2024.	Almost daily, with some pauses.
OCTOPUS [LOGS/URL/COMBO]	Channel for sharing info stealer logs and stolen credentials. Heavy advertising of a “private” access, available to members for an additional fee. Hosting over 500 files as of April 2024.	Almost daily, with some pauses.

3.1.2 Cybercrime forums for finding channels

The other two channels of the four will be found by the author on a popular cybercrime forum, XSS. For this, the author browses the subsection of the forum called Bases (i.e. databases, “*Базы*” in Russian) and located two channels of interest to be used with the scraper of this thesis.

Table 3. Analysis of the selected channels found from XSS.

Channel name	Author's description	Activity
--------------	----------------------	----------

AgressorDB FREE ComboLists Бесплатные базы email:pass	Channel for only sharing credential leaks. Often files are of increments of 50 thousand lines (i.e. 50, 100 or 150 thousand lines per file). Hosting over 200 files as of April 2024.	Daily, without pauses.
👁 OBSERVER CLOUD - BEST FREE LOGS CLOUD	Channel for mainly sharing info stealer logs but might contain some credential leak files. Hosting over 10 000 files as of April 2024.	Daily, many posts in a day.

3.2 Telegram scraper and alert bot

The application developed as a solution to the problem is divided into two parts:

- Telegram scraper – An automatic tool, used to iterate through the channels present on an account, download all of the found credential leak files, filter and clean the contents and save the results to a local database.
- Telegram alert bot – A Telegram bot, useable by the operator of the scraper. Allows to search the database created and populated in the first part by various parameters (such as username, password and URL).

In the following subsections, an overview of the tools, packages and technologies used to develop both parts of the solution are presented.

3.2.1 Telegram scraper

For the scraper development, the programming language Python 3.12 is used. Python package Telethon version 1.30 (used to interact with Telegram) and Python libraries Sqlite3 version 3.0 (used to interact with the local SQL database) and Collections version 3.3 (used for better datatypes and their structuring) are used. Also, Python modules os, re and hashlib are imported to the program for various tasks.

A Telegram account is controlled via the official API to scrape the channels, available at my.telegram.org for each account for free.

3.2.2 Telegram alert bot

For the alert bot development, there are not many differences. Python package Tldextract version 5.1.2 (used for separating URLs to various parts) and standard module Asyncio (used for asynchronous loops for the bot) are used.

Another Telegram account, separate from the scraper account, is deployed via the official API to act as a front-end, allowing the user to easily receive the scraped information in an easily comprehensible way.

4 Development and usage

This section focuses on the more technical side of developing the scraper and the bot, showing important functions, methods and solutions used to build both of the products. The author only demonstrates the most important functionality and code of the scraper and bot.

The usage of the whole system together is also demonstrated in a way of a “test-run”, beginning from launching the scraper and ending with formatted leaked credentials being sent back to the user querying them from the bot.

4.1 Development of the scraper

The main task of the scraper is to iterate through the channels the account is subscribed to, gather credential leak files and add them to the local database in a formatted way.

As shown in Figure 20, various API tokens need to be used of the Telegram account. Due to privacy concerns, they are redacted from this paper. A local database is established and populated with a simple table to house the formatted credentials.

```
# Establishing the client.
api_id = [REDACTED BY AUTHOR]
api_hash = [REDACTED BY AUTHOR]
client = TelegramClient('session', api_id, api_hash)

# Establishing the database.
conn = sqlite3.connect(credentialsDatabase.db')
cursor = conn.cursor()

cursor.execute('''
CREATE TABLE IF NOT EXISTS credential_lines (
    id INTEGER PRIMARY KEY,
    url TEXT NOT NULL,
    username TEXT NOT NULL,
    password TEXT NOT NULL,
    link TEXT NOT NULL
)
''')
```

Figure 20. Setting up the scraper.

After the initial setup, the iteration process of the channels may begin. As seen on Figure 21, the iteration function first loads any MD5 file hashes or file links on a channel present on the system to avoid downloading a duplicate credential leak file in the future. This is due to the fact that two separate channels might post the same credential leak, thus a basic link comparison is not sufficient. A MD5 hash check allows the program to generate a string relative to the contents of the file, no matter the name of it. The “load_downloads_info()” function simply returns hashes and message links separately from the local file “download/fileHashes.txt”.

```
async def download_files_from_subscriptions(client: TelegramClient):
    # Defining the hash-link file path.
    hashes_file_path = os.path.join('downloads', 'fileHashes.txt')
    # Reading from the hash-link file path.
    downloaded_files_hashes, downloaded_message_links =
load_downloads_info(hashes_file_path)
```

Figure 21. Gathering local information about already downloaded files.

After these actions, the iteration begins. A channel is selected and a subfolder in the “/downloads” directory is created for that unique channel and future downloaded credential leak files.

The iteration goes over every message present on a channel. It checks against the message ID and compares it to the local list (present in the file “download/fileHashes.txt”). If it finds a message ID that has already been downloaded, it skips the message. If the program has not seen that message ID, it checks if a text file is included with that message and starts a download of it to the subfolder of that channel.

After downloading it, the MD5 hash of it is generated and compared against the local list. If the hash is already present, the new file is deleted, and the iteration continues. If it is not present, the code saves the information of a new credential leak file to the local list and starts processing the file. These steps are illustrated in Figure 22.


```

# Checking if the file in the Telegram message is a text file.
if message.file and message.file.name.endswith('.txt'):
    temp_file_path = os.path.join(download_directory, message.file.name)
    # Downloading the file and calculating the MD5 hash.
    print(f"Starting download: {temp_file_path}")
    await message.download_media(file=temp_file_path)
    file_hash = compute_md5(temp_file_path)

    # Deleting the file if already logged as downloaded.
    if file_hash in downloaded_files_hashes:
        print(f"File with the same hash already exists:
        {downloaded_files_hashes[file_hash]}")
        os.remove(temp_file_path)
    # Saving the download info locally, starting processing.
    else:
        print(f"Downloaded: {temp_file_path}")
        downloaded_files_hashes[file_hash] = temp_file_path
        downloaded_message_links.add(message_link)
        save_download_info(hashes_file_path, file_hash, temp_file_path,
        message_link)
        process_file(temp_file_path, channel_username, message.id)

```

Figure 22. Downloading a .txt file with subsequent processing.

The “process_file()” function is a loop for each line inside of the downloaded file. After a line is cleaned of whitespaces, it is sent to the “insert_line_into_database()” function, which acts as the processing and cleaning function of the lines. This allows them to be inserted into correct sections of the database for later querying via the bot.

This function starts off by processing the line in the format of “URL username:password” and moves on to processing it as “email:password”, if a match for that format is not found. This implementation is shown on Figure 23.

```

# Using lines formatted as URL username:password.
if ' ' in line and ':' in line.split(' ')[-1]:
    parts = line.split(' ')
    if len(parts) == 2:
        url, credentials_part = parts
        credentials = credentials_part.split(':', 1)
        if len(credentials) == 2:
            username, password = credentials
        else:
            return
    else:
        # Using lines as email:password.
        credentials = line.split(':', 1)
        if len(credentials) == 2:
            email, potential_password = credentials
            if re.match(r"^[^@]+@[^@]+\.[^@]+", email):
                username, password = email, potential_password
            else:
                return
        else:
            return

```

Figure 23. Ways of processing a line.

If no match for the format is found with these two attempts, the program tries to categorize the line as of Android origin (credential lines saved from Android applications, starting with the prefix “android://”).

If still no match is found, the system is almost confident that the credential leak is of the format “url:username:password”. As seen on Figure 24, it starts processing it like this.

```

# Locating URL start points.
scheme_end = line.find('://') + 3 if '://' in line else 0
first_colon = line.find(':', scheme_end)

# Determining URL type.
if first_colon != -1 and (line.find('/', scheme_end) < first_colon or
line.find('/', scheme_end) == -1):
    # Separating username from URL.
    url = line[:first_colon]
    credentials = line[first_colon+1:].split(':', 1)
    if len(credentials) == 2:
        username, password = credentials
    else:
        return
else:
    url = line

```

Figure 24. Another way of processing a line.

After fully processing a single line and determining that at least the username and password have been extracted (URL is not mandatory), an SQL INSERT command is executed, successfully attaching the valid line to the local database. This part of the code is displayed on Figure 25. After all of the lines are added to the database, a SQL “commit()” function is used at the end of processing each file, thus saving the entries to the local DB.

```
# Validating the existence of the needed parts.
if username and password:
    cursor.execute("INSERT INTO file_lines (url, username, password, link)
VALUES (?, ?, ?, ?)", (url, username, password, link))
```

Figure 25. Adding the line to the database.

As mentioned at the start of this subsection, the full code file of the scraper is not present in this topic, as the author hopes to give an overview of only the most crucial parts of the inner workings of this Telegram spider.

4.2 Development of the alert bot

The main task of the alert bot is to serve as an easy-to-use front end for the credential leak database, allowing the user to interact with it without any deep technical or programming knowledge. As seen on Figure 26, the setup is quite alike to the scraper, with some adjustments made to serve as a bot, not a default Telegram account. Sensitive data and tokens are redacted by the author.

```
# Establishing the bot client.
bot_sername = [REDACTED BY AUTHOR]
api_id = [REDACTED BY AUTHOR]
api_hash = [REDACTED BY AUTHOR]
bot_token = [REDACTED BY AUTHOR]
client = TelegramClient('bot', api_id, api_hash).start(bot_token=bot_token)

# Connecting the to the database.
conn = sqlite3.connect('credentialsDatabase.db', check_same_thread=False)
cursor = conn.cursor()
```

Figure 26. Setting up the bot.

Telegram bots are interacted with by using custom commands. The author has added three commands to be used withing the bot:

- Command “/start” – Sending a welcome text to the user, with instructions on using the bot.
- Command “/query” – Querying data from the DB, returning results.
- Command “/stats” – Calculating statistics of the present results in the DB.

The “/start” command is quite trivial and only transmits two messages to the user of the bot, as seen on Figure 27.

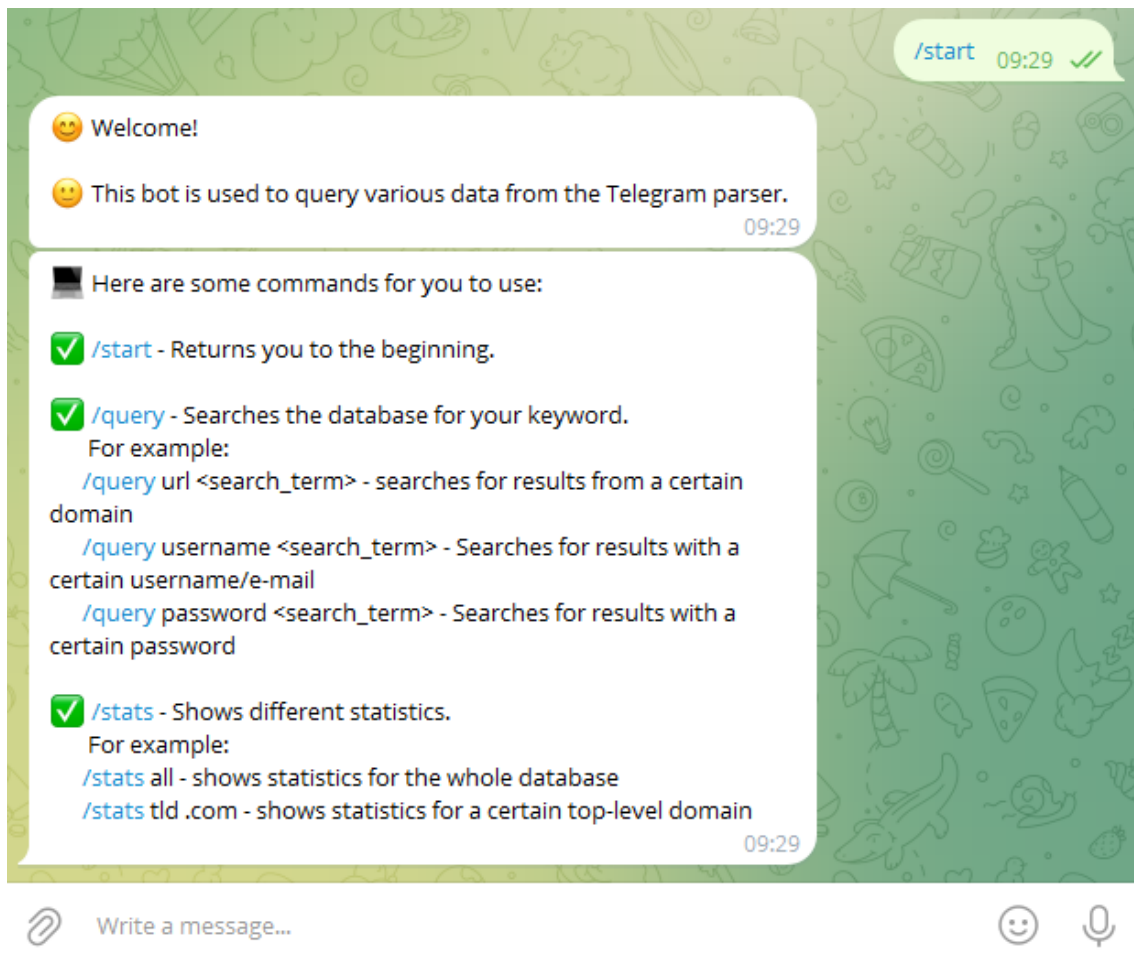


Figure 27. The “/start” command.

The “/query” command is a bit more sophisticated. It features a checker, to ensure that the user has sent the correct full command and hasn’t skipped any parts (see Figure 28). The correct field (url, username or password) must be selected before starting the query.

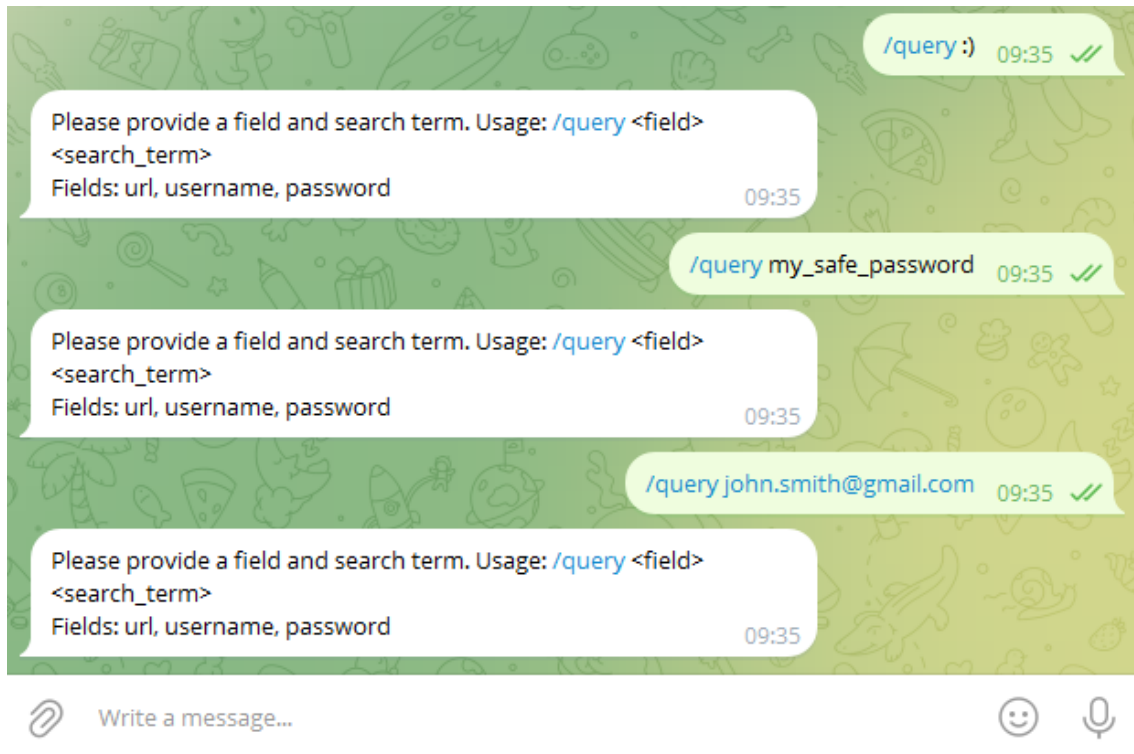


Figure 28. Checking the correctness of the “/query” command.

If the command is correct, the query from the DB begins. A SQL SELECT operator is used, as seen in Figure 29. If result(s) exist in the DB, they are displayed to the user in a structured and designed way, allowing for quick identification and tracking of the credentials obtained by the scraper. As Telegram messages have a maximum length of 4096 characters [31], it is also checked, with the message getting broken down into a new one if the limit is to be reached.

```

# The “/query” command of the bot.
@client.on(events.NewMessage(pattern='/query'))
async def query(event):
    # Splitting the message to 3 parts.
    parts = event.message.text.split(maxsplit=2)

    # Ensuring the correctness of the command.
    if len(parts) < 3 or parts[1] not in ['url', 'username', 'password']:
        await event.respond('Please provide a field and search term. Usage:
/query <field> <search_term>\nFields: url, username, password')
        return

    # Querying data from the local DB.
    field, search_term = parts[1], parts[2]
    query = f"SELECT * FROM credential_lines WHERE {field} LIKE ?"
    cursor.execute(query, ('%' + search_term + '%',))
    results = cursor.fetchall()

    # Displaying the results in an aesthetic way.
    if results:
        response = '\n'.join([f"🔗 URL: {row[1]}\n👤 Username: {row[2]}\n🔑 Password: {row[3]}\n🔗 Link to the post: {row[4]}\n" for row in results])
        if len(response) > 4096:
            for x in range(0, len(response), 4096):
                await client.send_message(event.chat_id, response[x:x+4096],
link_preview=False)
            else:
                await client.send_message(event.chat_id, response,
link_preview=False)
            else:
                await client.send_message(event.chat_id, '😞 Sorry! No results
found.', link_preview=False)

```

Figure 29. The functionality of the “/query” command.

The last command available to the user, “/stats”, also has a checker for the needed parts being present, as seen on Figure 30. The needed fields are either “all” for receiving statistics about all of the lines present in the DB or “tld <search_term>” for receiving statistics about a certain TLD selected by the user (i.e. “/stats tld .ee”).

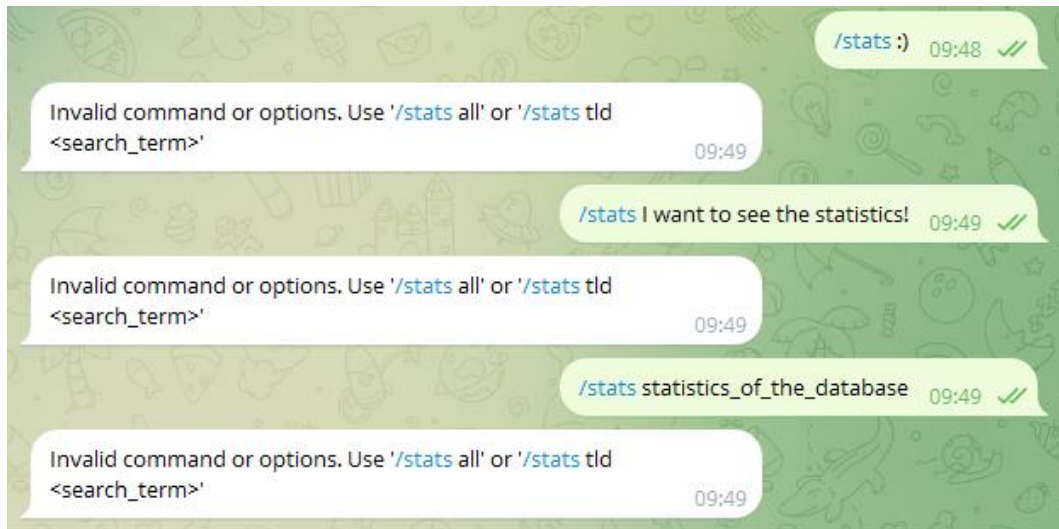


Figure 30. Checking the correctness of the “/stats” command.

There are different values displayed about the database. Firstly, the user gets information about the line count in the local DB, as implemented on Figure 31.

```
# Constructing the WHERE clause for the DB.
where_clause = ""
params = ()
# Used only if user defined a custom TLD.
if custom_tld:
    where_clause = "WHERE url LIKE ?"
    params = ('%' + custom_tld,)

# Calculating the total lines from the DB.
cursor.execute(f"SELECT COUNT(*) FROM credential_lines {where_clause}",
params)
total_lines = cursor.fetchone()[0]
```

Figure 31. Counting the lines in the DB.

After the count of the lines, statistics about the top 10 most popular domains are presented for the user (see Figure 32). If a TLD is selected, then the top 10 most popular domains of that certain TLD are displayed.

```

# Selecting URLs from the table.
cursor.execute(f"SELECT url FROM credential_lines {where_clause}", params)
    urls = cursor.fetchall()
    domains = [f"{extracted.domain}.{extracted.suffix}" for url in urls if
url[0] for extracted in [tldextract.extract(url[0])] if not custom_tld or
extracted.suffix == custom_tld.strip('.')]
    # Counting and only using top 10 most common domains.
    domain_counter = Counter(domains)
    top_domains = domain_counter.most_common(10)

```

Figure 32. Finding top 10 most common domains.

For statistics about the top 10 most popular passwords, the process is quite same. If a TLD is selected by the user, only top 10 most popular passwords on that TLD are searched for. If not, the whole DB is utilized for determining the top 10 most popular passwords. The implementation for this is displayed on Figure 33.

```

# If a TLD is selected.
if custom_tld:
    top_domain_patterns = [f"%{domain}%" for domain, _ in top_domains]
    cursor.execute(f"SELECT password FROM file_lines WHERE url LIKE ?",
(top_domain_patterns[0],))
    passwords = [pw[0] for pw in cursor.fetchall() if pw[0]]
# If a TLD is not selected.
else:
    cursor.execute("SELECT password FROM file_lines")
    passwords = [pw[0] for pw in cursor.fetchall() if pw[0]]
password_counter = Counter(passwords)
top_passwords = password_counter.most_common(10)

```

Figure 33. Finding top 10 most common passwords.

After all of the statistics have been calculated, they are sent to the user. Again, it is formatted aesthetically, including some emojis and new lines in the Telegram bot message. An implementation is shown on Figure 34. Link previews are disabled for a better visual experience to the user.


```

# Preparing domain statistics.
domain_stats = "\n".join([f"{i + 1}. {domain} (Occurrences: {count})" for i,
(domain, count) in enumerate(top_domains)])

# Preparing password statistics.
password_stats = "\n".join([f"{i + 1}. {password} (Occurrences: {count})" for
i, (password, count) in enumerate(top_passwords)])

# Formatting the message.
response_title = "📊 Database Statistics 📊" if option == "all" else f"📊
Database Statistics for TLD: {custom_tld} 📊"

response = f"""\{response_title}

📄 Total Lines: {total_lines}

🌐 Top 10 Most Common Domains:
{domain_stats}

🔑 Top 10 Most Common Passwords for the domains above:
{password_stats}"""

# Sending the message.
await client.send_message(event.chat_id, response, link_preview=False)

```

Figure 34. Sending the statistics to the user.

4.3 Usage

This section shows the reader how the Telegram scraper and bot work together to prove the possibility of solving the problem stated in the thesis.

The author started the solution and it successfully scraped all of the four channels previously mentioned for credential leak text files. There were no corrupt files, and no major errors were encountered. The Python program did not crash. The digital size of the database created and filled with credential lines reached over 43 GB after the scrape (see Figure 35). Although there may exist duplicate lines, the author is confident that most of the lines are unique, and the DB is not largely oversized because of the duplicates.

The author has not and will not access any resource, system, website or application with any of the credentials or information found with this solution.

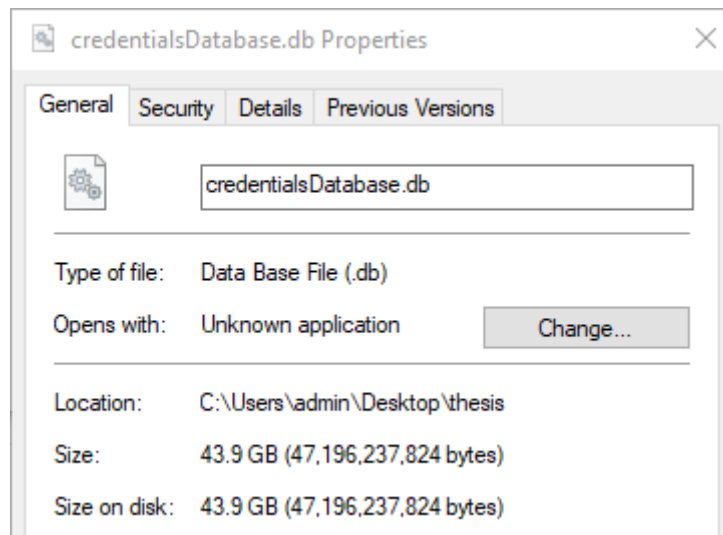


Figure 35. Size of the scraped data.

After verifying the completion of the scraping process, the author started the Telegram bot. The first query sent was for an URL, and the author searched the findings for “postimees.ee”, a popular news site in Estonia. This is a great resource to search for, as the page features a simple username-password login feature, where users might have saved the combinations into their browser and thus leaked them via infection by info stealer malware. As seen on Figure 36, the bot returned multiple combinations of credentials for “postimees.ee”, proving the ability to find, scrape, process and alert the user of leaked credentials on Telegram, searched by the URL.

The author has redacted the username and password from Figure 36, only showing the first and last symbols for visual proof.

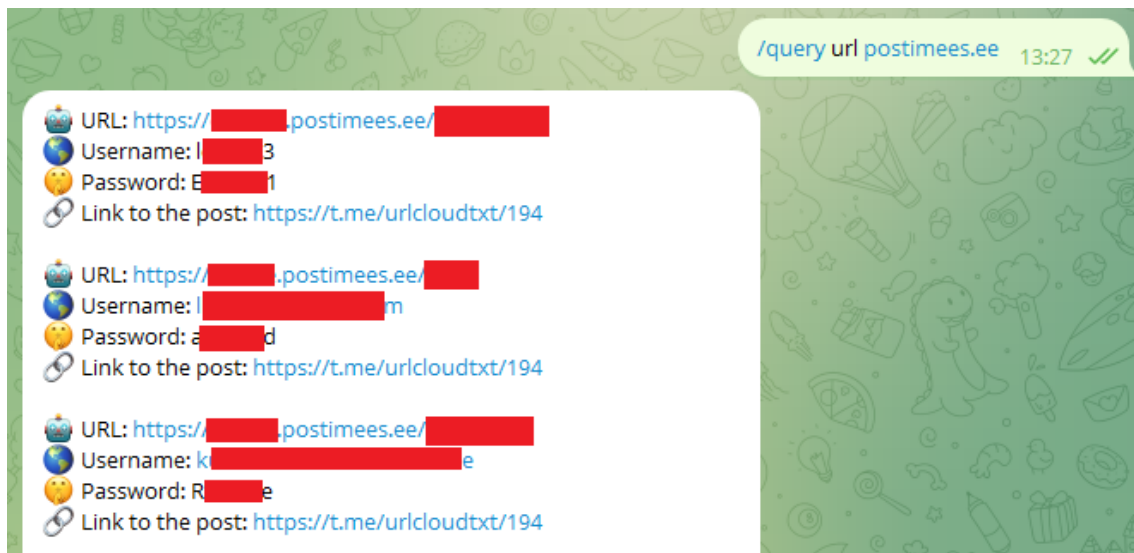


Figure 36. Querying by URL.

The next function of the “/query” command was tested by searching the local database for a username. The author picked a username widely used, thus having a bigger chance of receiving a match from the database, but still rare enough to prove that the database contains a diverse set of data. A value “kasutaja123” (“*user123*” in English) was queried by the author, with the result displayed on Figure 37. As seen, the bot returned many combinations of credentials for “kasutaja123”, proving the ability to find, scrape, process and alert the user of leaked credentials on Telegram, searched by the username.

The author has redacted the URL and password from Figure 37, only showing some non-identifiable symbols for proof.

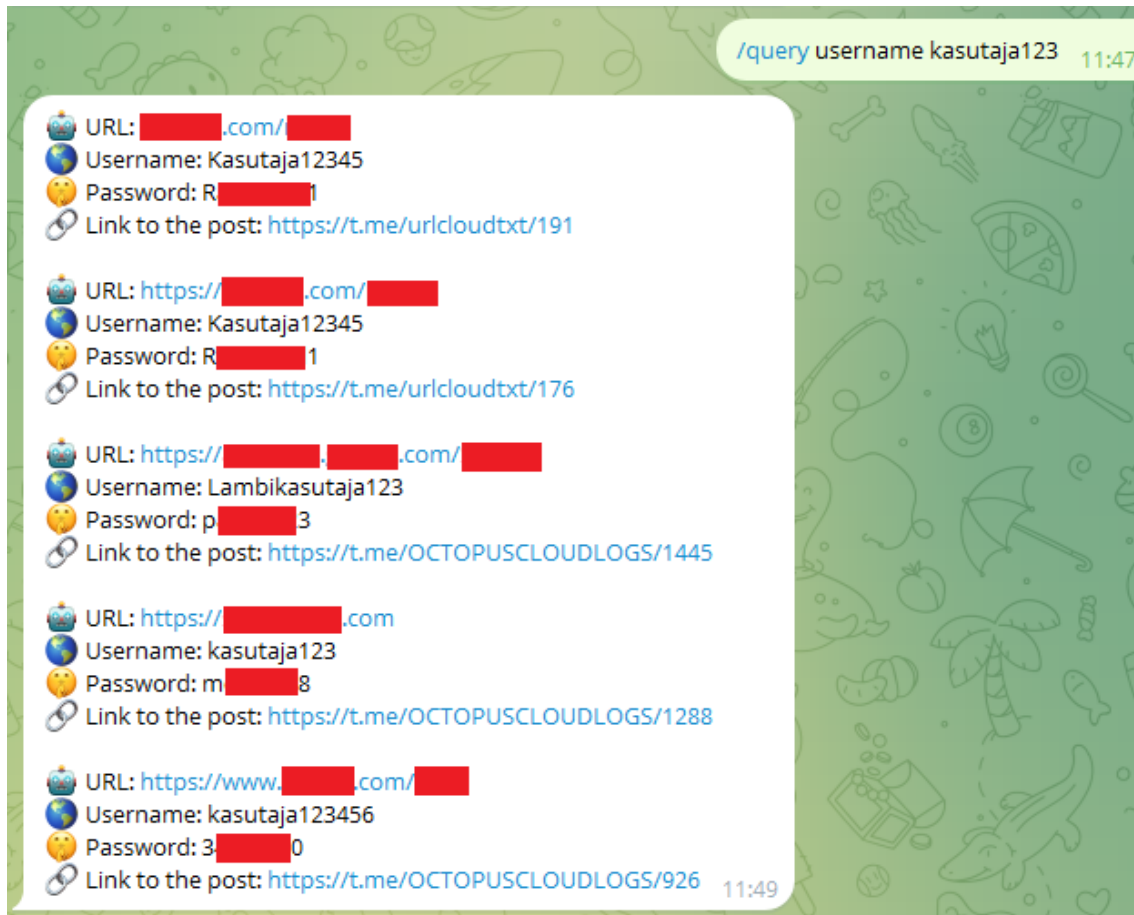


Figure 37. Querying by username.

The last function of the “/query” command was tested by searching local database for a password. The author picked a popular, but still somewhat unique password to be used for the search for the same reasons as outlined for other queries. A value “salajane123” (“*secret123*” in English) was used for the operation. Featured on Figure 38 is a successful query with it, proving the ability to find, scrape, process and alert the user of leaked credentials on Telegram, searched by the password.

The author has redacted the URL and username from Figure 38, only showing some non-identifiable symbols for proof of the working solution.



Figure 38. Querying by password.

As for the statistics function, the author queried all of the available data by using the command “/stats all”. The result successfully showed the top 10 most popular domains and top 10 most popular passwords found in the local database populated by the scraper. As seen on Figure 39, the output also included the line count of the DB, which was over 415 million lines.

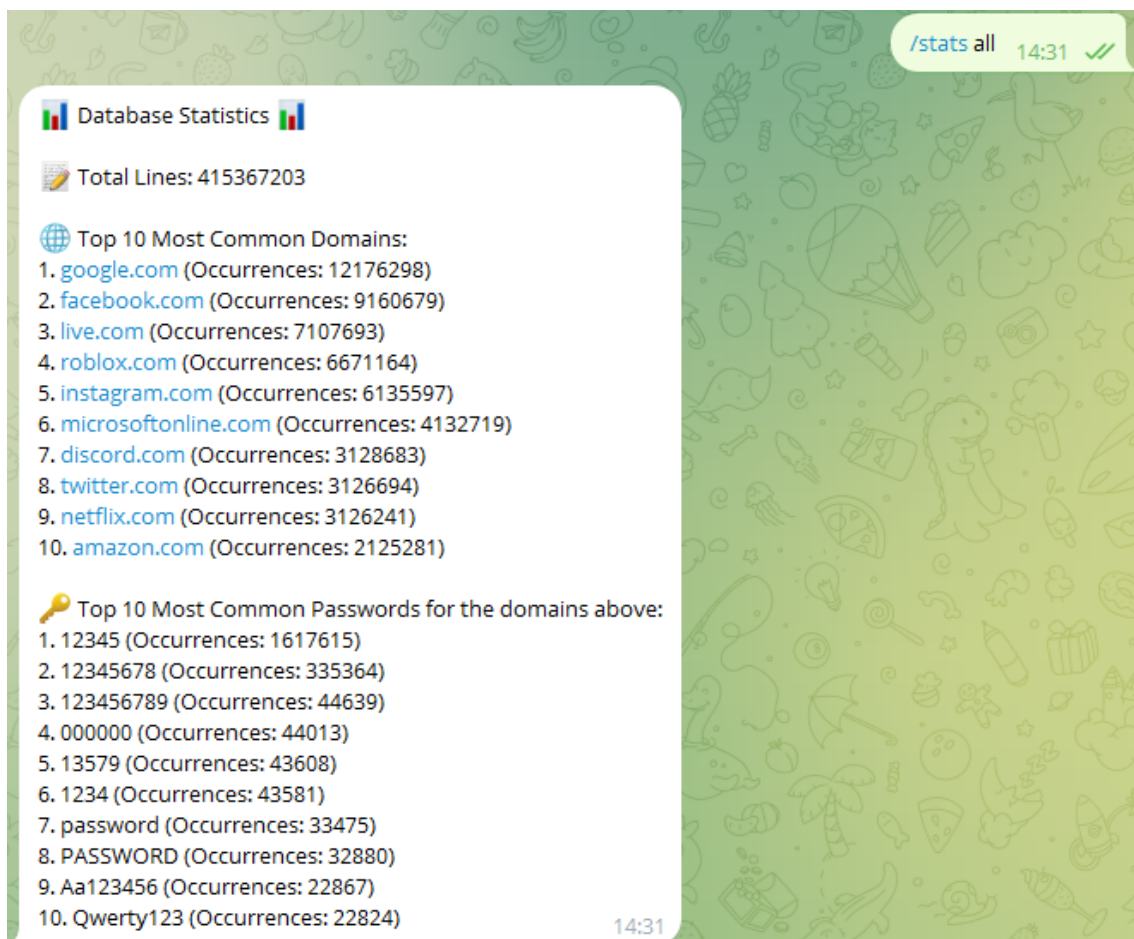


Figure 39. Statistics for the whole DB.

5 Summary

The topic of credential leaks has been an important talking point in the cyber security scene for decades. While trivial conceptually, simple data breaches are utilized by cyber criminals, threat actors and APT groups worldwide. Also, the internet is still full of services, applications and sites that allow for access to be granted solely by using a username and a password.

This thesis successfully demonstrated the prevalence of credential leaks on the web, with the possibility to resource them from cyber crime forums or the Telegram messenger, which acts as a hub for cybercrime. One does not need to have a strong skillset for acquiring various illegally obtained credentials from Telegram, thus increasing the count of potential malicious users of that data.

While big corporations and governments have strong cyber security budgets and safeguards in place, which surely cover such credential leaks, the less-protected bodies and individuals in our society are especially at risk against such attacks. There exist some solutions to track and map the enormous landscape of credential leak sources, but they are either very costly or lack the features of tracking them on Telegram app, which makes up a large part of such ecosystem.

In this paper, the author proposed, developed and demonstrated a fully working solution to automatically locate, download, process and save various forms of credential leaks shared on Telegram. Also, an alert bot was developed on the same platform, giving the user handy access to all of the data found by the scraper, without any limitation's payment plans, obfuscations or other blockages.

This project serves as a helping hand for individual users and smaller businesses or corporations, who might not be as skilled or resourced as bigger corporations or governments, but still find the need to track their credential leaks on Telegram.

As for future developments, the solution could be optimized in a couple of ways, but not limited to:

- Researching more about Telegram credential leak channels with the hypothesis that channels with more sensitive and fresh information exist on Telegram.
- Improving the scraper to parallelly download and process multiple files, instead of one at a time as with the current development.
- Improving the data storage and retrieval functions, as it currently slows down the working processes of the bot when having a very large database.

References

- [1] eSentire. “2023 official Cybercrime report,” eSentire. [Online]. Available: <https://www.esentire.com/resources/library/2023-official-cybercrime-report> (accessed Apr. 10, 2024).
- [2] ITRC. “ITRC Annual Data breach report,” ITRC. [Online]. Available: <https://www.idtheftcenter.org/publication/2023-data-breach-report/> (accessed Apr. 10, 2024).
- [3] KELA Cybercrime Intelligence. (2023, Feb.). “How a messenger turned into a cybercrime ecosystem by 2023,” KELA Cybercrime Intelligence. [Online]. Available: https://www.kelacyber.com/wp-content/uploads/2023/02/KELA_Telegram_CEBIN.pdf (accessed Apr. 10, 2024).
- [4] MITRE ATT&CK®. “Acquire Access, Technique T1650 – Enterprise”. [Online]. Available: <https://attack.mitre.org/techniques/T1650/> (accessed Apr. 10, 2024).
- [5] Searchlight Cyber. “XSS,” Searchlight Cyber. [Online]. Available: <https://www.slcyber.io/dark-web/xss/> (accessed Apr. 15, 2024).
- [6] Kaspersky. (2022, Jun. 15.). “Cybercriminals sell access to companies via the dark web from \$2000,” Kaspersky. [Online]. Available: https://www.kaspersky.com/about/press-releases/2022_cybercriminals-sell-access-to-companies-via-the-dark-web-from-2000 (accessed Apr. 15, 2024).
- [7] MITRE ATT&CK®. “Exploit Public-Facing Application, Technique T1190 – Enterprise,” [Online]. Available: <https://attack.mitre.org/techniques/T1190/> (accessed Apr. 10, 2024).
- [8] OWASP Foundation. “SQL Injection”. [Online]. Available: https://owasp.org/www-community/attacks/SQL_Injection (accessed Apr. 10, 2024).
- [9] MITRE ATT&CK®. “Phishing, Technique T1566 – Enterprise,” [Online]. Available: <https://attack.mitre.org/techniques/T1566/> (accessed Apr. 10, 2024).
- [10] A. Petrosyan. (2023, Mar.) “Number of malware attacks per year 2022,” Statista. [Online]. Available: <https://www.statista.com/statistics/873097/malware-attacks-per-year-worldwide/> (accessed Apr. 10, 2024).
- [11] D. Collyer. (2023, Aug. 2.). “Stealer logs – what you need to know,” SOS Intelligence. [Online]. Available: <https://sosintel.co.uk/stealer-logs-what-you-need-to-know/> (accessed Apr. 14, 2024).
- [12] Group-IB. “Underground cloud of logs,” [Online]. Available: <https://www.group-ib.com/resources/knowledge-hub/underground-cloud-of-logs/> (accessed Apr. 14, 2024).
- [13] D. W. Opderbeck, “Cybersecurity and Data Breach Harms: Theory and Reality”, 82 Md. L. Rev. 1001 () Available at: <https://digitalcommons.law.umaryland.edu/mlr/vol82/iss4/4> (accessed Apr. 14, 2024).
- [14] J.Buxton, T. Bingham. (2015, Jan.) “The rise and challenge of Dark Net Drug Markets,” Global Drug Policy Observatory. [Online]. Available: <https://www.swansea.ac.uk/media/The-Rise-and-Challenge-of-Dark-Net-Drug-Markets.pdf> (accessed Apr. 15, 2024).

- [15] Mozilla Monitor. (2023, Dec. 7.). “Were you affected by the Joygames Data Breach?,” Mozilla Monitor. [Online]. Available: <https://monitor.mozilla.org/breach-details/JoyGames> (accessed Apr. 15, 2024).
- [16] P. H. O’Neill. (2017, Apr. 19.). “Why jabber reigns across the Russian cybercrime underground,” CyberScoop. [Online]. Available: <https://cyberscoop.com/jabber-xmpp-cybercrime-russia-encrypted-chat/> (accessed Apr. 15, 2024).
- [17] OTR Development Team. (2024). Off-the-Record Messaging. [Online]. Available: <https://otr.cypherpunks.ca/> (Accessed: 15 April 2024).
- [18] OTR.im. “Whispering Off The Record,” OTR.im. [Online]. Available: <https://otr.im/clients.html> (accessed: 15 April 2024).
- [19] Tox Project. “A new kind of instant messaging,” Tox Project. [Online]. Available: <https://tox.chat/faq.html> (accessed Apr. 16, 2024).
- [20] ubivis_project. (2022, Apr. 18.). “Understanding Tox Chat,” The Cyber Social Hub. [Online]. Available: <https://cybersocialhub.com/csh/understanding-tox-chat/> (accessed Apr. 16, 2024).
- [21] J. DiMaggio. “Ransomware diaries v. 3: Lockbit’s Secrets,” Analyst1. [Online]. Available: <https://analyst1.com/ransomware-diaries-volume-3-lockbits-secrets/> (accessed Apr. 16, 2024).
- [22] M. Burgess. (2024, Feb. 20.). “A global police operation just took down the notorious Lockbit Ransomware Gang,” Wired. [Online]. Available: <https://www.wired.com/story/lockbit-ransomware-takedown-website-nca-fbi/> (accessed Apr. 16, 2024).
- [23] M. Matthias. (2024, Apr. 15.). “Telegram” Encyclopædia Britannica. [Online]. Available: <https://www.britannica.com/topic/Telegram-software> (accessed Apr. 16, 2024).
- [24] P. Durov. (2023, Aug. 14.). “Du Rove’s channel” [Online]. Available: <https://t.me/durov/216> (accessed Apr. 16, 2024).
- [25] E. Bowman. (2021, Apr. 9.). “After data breach exposes 530 million, Facebook says it will not notify users,” NPR [Online]. Available: <https://www.npr.org/2021/04/09/986005820/after-data-breach-exposes-530-million-facebook-says-it-will-not-notify-users> (accessed Apr. 16, 2024).
- [26] Have I Been Pwned. (2024). Check if your email has been compromised in a data breach [Online]. Available: <https://haveibeenpwned.com/> (accessed Apr. 16, 2024).
- [27] Intelligence X. (2024). About - Intelligence X [Online]. Available: <https://intelx.io/about> (accessed Apr. 16, 2024).
- [28] Intelligence X. (2024). Product - Intelligence X [Online]. Available: <https://intelx.io/product> (accessed Apr. 16, 2024).
- [29] GeeksForGeeks. (2023, Mar. 28.). What is google dorking? [Online]. Available: <https://www.geeksforgeeks.org/what-is-google-dorking/> (accessed Apr. 16, 2024).
- [30] T. Bianchi. (2024). “Global Search Engine Desktop Market Share 2024,” [Online]. Statista. Available: <https://www.statista.com/statistics/216573/worldwide-market-share-of-search-engines/> (accessed Apr. 16, 2024).
- [31] CM.com Developers Portal. (2024, Apr. 16.). Telegram API Docs [Online]. Available: <https://developers.cm.com/messaging/docs/telegram> (accessed Apr. 17, 2024).

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis⁸

I, Oskar Pikkov,

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Developing a Telegram Scraper to Identify and Alert on Leaked Credentials Based on Pre-Defined User Input”, supervised by Toomas Lepikult and Mert Meissaar,
 - 1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
 - 1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

13.05.2024

⁸ The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.