OSINT TECHNIQUES: LEAKS, BREACHES, & LOGS

MICHAEL BAZZELL

DIGITAL EDITION

OSINT TECHNIQUES:

LEAKS, BREACHES, & LOGS

MICHAEL BAZZELL

OSINT TECHNIQUES:
LEAKS, BREACHES, & LOGS

First Published: December 2023

Project Editors: Anonymous Editor #1, Anonymous Editor #2

Cover Concept: Anonymous Podcast Listener

Due to the use of quotation marks to identify specific text to be used as search queries and data entry, the author has chosen to display the British rule of punctuation outside of quotes. This ensures that the quoted content is accurate for replication. To maintain consistency, this format is continued throughout the entire book.

The technology referenced in this book was edited and verified by a professional team for accuracy. Exact tutorials in reference to websites, software, and hardware configurations change rapidly. All tutorials in this book were confirmed accurate as of November 1, 2024. Readers may find slight discrepancies within the methods as technology changes.

Revision: 2024.11.01

CONTENTS

PREFACE
INTRODUCTION
CHAPTER 01: Investigative Benefits
CHAPTER 02: Hardware Configuration
CHAPTER 03: Software Configuration
CHAPTER 04: Terminal Commands
CHAPTER 05: Data Leaks
CHAPTER 06: Data Breaches
CHAPTER 07: Stealer Logs
CHAPTER 08: Ransomware Data
CHAPTER 09: Scripts, Databases, & Backups
CONCLUSION

These contents are provided as a summary. Page numbers and hyperlinks are not included because this is a living document which receives constant updates. Please use the search feature of your PDF reader to find any exact terms or phrases, as that is much more beneficial than any index or hyperlinked table of contents.

ABOUT THE AUTHOR

MICHAEL BAZZELL

Michael Bazzell investigated computer crimes on behalf of the government for over 20 years. During the majority of that time, he was assigned to the FBI's Cyber Crimes Task Force where he focused on various online investigations and Open Source Intelligence (OSINT) collection. As an investigator and sworn federal officer through the U.S. Marshals Service, he was involved in numerous major criminal investigations including online child solicitation, child abduction, kidnapping, cold-case homicide, terrorist threats, and advanced computer intrusions. He has trained thousands of individuals in the use of his investigative techniques and privacy control strategies.

After leaving government work, he served as the technical advisor for the first season of the television hacker drama *Mr. Robot*. His books *OSINT Techniques* and *Extreme Privacy* are used by several government and private organizations as training manuals for intelligence gathering and privacy hardening. He now assists individual clients in achieving ultimate privacy, both proactively and as a response to an undesired situation. More details about his services can be found at *IntelTechniques.com*.

LEAKS, BREACHES, & LOGS PREFACE

We released the 10th edition of our 550-page ***OSINT Techniques*** print book in January of 2023. While most of that edition is still accurate, we felt the section about data leaks, breaches, and logs could use more details and tutorials. It has been the most popular topic from the book and readers seem hungry for more content. That print book has reached the maximum page limit, so a new edition would not allow us to expand the information. Therefore, we have released this digital guide as a response to the interest and a supplement to the original text.

Each topic has been expanded and we offer numerous new resources for fresh data. All expired and outdated methods were replaced with new techniques, and brand-new topics were introduced throughout. As we explain various Terminal commands, we present practical exercises with real data to make sure you have grasped the concepts. By the time you begin collecting your own data, you will be proficient in the commands required to make the content easily searchable. By the end of the guide, you will be able to fully replicate the databases behind many paid services without spending any money. Combined with the new automated scripts provided on our website, you will be ready to acquire, sort, and query all publicly-available breach data. We also explain all daily, weekly, and monthly tasks required to maintain your data collection.

With this release, we are only providing a PDF. There are no official print versions and we have eliminated Amazon from the entire publication process. This allows us to offer a lower price, and 90% of each purchase directly supports our efforts to keep the content updated as things change.

We realize that a native PDF will lead to immediate piracy of this work online. We accept that. We believe that we can offer further benefits to legitimate purchasers by offering free updates when appropriate. When we need to modify existing content or add entire new sections, we can send an email blast to all purchasers which will allow them to download a new copy with all updates for free. This model has worked well for our last four ***Extreme Privacy*** digital guides. If you find anything which needs updated or corrected, please email us at books@inteltechniques.com. My staff cannot respond to emails directly, but they will monitor them for any changes which we need to apply to the next version of this guide.

Since each copy of this work is watermarked with both a visible and hidden unique code, we can block updates from those who publish the book without consent. Overall, we want to reward those who support us with a searchable, copyable, updatable, and printable document, even at the risk of losing half of our sales to the pirates. If you bought this, thank you for your support! If you did not, consider purchasing a legitimate copy in order to receive all future updates. With ***OSINT Techniques: Leaks, Breaches, & Logs***, we present a new approach to our OSINT tutorials. It is not a replacement for ***OSINT Techniques*** (the printed book). Please consider it a much more thorough supplement about this specific topic which we can update as things change.

INTRODUCTION

The print edition of *OSINT Techniques* had a single section devoted to data leaks, breaches, and logs. It was possibly the most controversial section, and contained the content which was most discussed and debated. This new digital guide, which contains nine chapters dedicated to these topics, attempts to expand on the original teachings.

Obligatory Warning: The techniques that you will read about in this section are for educational use only. Many, if not all, of the methods described here could violate organizational policy if executed. While I will only discuss publicly available data, possession could violate your security clearance or be determined illegal by state or federal law. Distribution (not necessarily possession) of the types of content that will be discussed here may be illegal in most cases. However, I will explain ways in which you can apply these practices in a legal way, and keep yourself employed. Overall, please do not take any action from this instruction unless you have verified with your organization's legal counsel or supervisory personnel that you have the authority to do so. Please do not let this warning persuade you to abandon the guide. I promise there is something here for everyone.

In previous books, I discussed the services Have I Been Pwned (HIBP) and others as online resources for compromised email search. These services possess huge databases of publicly available content which were originally stolen from the host companies and distributed over the internet. When you search an email address on these services and are informed that it was compromised within the LinkedIn data breach, this means that a partial copy of this stolen data resides within these breach lookup services. HIBP and others are often applauded as a great resource to monitor your own accounts for any reported compromises. Well, what if we created our own unredacted collection of this data? Is there additional value? I believe so.

This is where things get tricky. While you can find copies of thousands of stolen databases all over the internet, what are the legalities of downloading and possessing the data? First, let me say that I am not an attorney and I offer no legal advice. I have spoken with many attorneys and prosecutors about this, and the feedback was similar from each. If the data is publicly available, possession alone would likely be legal. This is similar to viewing an email stolen from Hillary Clinton posted on WikiLeaks or an internal document stolen from Google posted on a blog. If you do not violate the laws and policies applicable in your city, county, state, or country when you encounter this publicly available, yet stolen, data, then you could likely get away with viewing stolen credentials existing in the various database leaks online.

What matters most is that you never take any illegal action with the data which you possess. In a moment, I will explain how to access valid email addresses and passwords of billions of accounts. Using this data as an investigation technique is one extreme, but attempting to use this data to access someone's account is another. There is no situation where casually gaining access to a target's account is acceptable, unless you have a valid search warrant or court order to do so. Since some of you are in law

enforcement, this chapter may identify new strategies for you when you have the legal authority to access an account. We will start with some very basic legal "leaked" data in just a moment.

Some will argue that this type of book is reckless and teaches criminals how to do bad things. I disagree. The criminals are way ahead of us. They have been using and sharing these techniques for many years. We can either ignore this type of data and hope for the best, or we can face it head on and be better investigators. I choose the latter.

I offer one last vital piece of information before we start. I encourage you to generate your own opinions as you read along. You may disagree with me at times, which is ideal. That means you are really thinking about how all of this applies to you. **If everyone unconditionally agrees with every word I say, then I am probably not saying anything interesting. If this book only presented content which no one could dispute, then there was no need for the text.** Please read with an open mind and willingness to try new things. Let's begin.

CHAPTER ONE
INVESTIGATIVE BENEFITS

Before proceeding, we should discuss some definitions for the types of data which will be discussed and beneficial uses for each. We can then tackle the hardware and software requirements. First, let's define the scope. There are many lengthy definitions and explanations of this type of data, but let's keep things simple. For the purposes of this guide, the following apply.

Leaks: This type of data is accidentally released, or "leaked", through public online resources. It could be public voter data which was legally acquired and then shared online; a marketing database of public information which was archived through a third-party site; a misconfigured database which is publicly leaking all details if you know where to look; or a plethora of other possibilities. The common theme is that no one intentionally compromised a secure server and stole anything. Of the types of data, this is the most likely to be legally accessed.

Breaches: This type of data takes us into the criminal world. Breaches occur when a criminal compromises a source of data illegally and copies the content. Examples include the Twitter and LinkedIn breaches where people's email addresses and passwords were stolen and published online. There are now tens of thousands of data breaches floating around, which have introduced billions of plain-text passwords for the majority of people online. While accessing or downloading this publicly-available data might not get you into legal trouble, further publishing it might. Also, your organization's policies might prevent you from this access. Possession and research of this data is extremely common today, even by us law-abiding internet citizens. More warnings are to come.

Logs: This data, sometimes referred to as "Stealer Logs" or "Password Logs" is illegally obtained from infected machines. This occurs when a criminal tricks a victim into installing a virus which extracts data and sends it to an accessible third-party server. This often includes stored passwords, session cookies, documents, photos, and other sensitive information. This may sound like a rarity, but hundreds of thousands of victim's logs are uploaded every day. This type of data can expose the passwords of people who are not within any known data breaches, and the content is much more likely to be accurate.

Ransomware: This type of data is also illegally acquired. Criminals install a virus within a company's network; infect many machines; make a copy of all data; encrypt all files on the network; and hold the decrypted copies hostage. They then offer to decrypt all data for a ransom, and threaten to publish the stolen content if payment is not received. Every day, new private company data is published online for anyone to see. This often includes passwords, sensitive documents, and copies of passports.

Offense: There are many reasons for interest in leaks, breaches, logs, and ransomware. I always split them into the sides of "offense" and "defense". Let's discuss a few of both, beginning with offense.

Identify Home Addresses: Some people do a very good job of keeping their names and home addresses off of the internet. You may strike out with people search websites. However, they often slip-up when ordering products to their home or registering to vote. Breach and leak data will often disclose true home addresses when no other online resource was successful.

Identify Alias Account Holders: In a later chapter, I will explain how I uncover the true owners of many "burner" email addresses due to common recycling of passwords. When my suspect uses a junk email account, and it appears within a data breach, I may see his password is "badguy432!@". When I discover an additional email address which had a password of "badguy432%$", I call that a lead. A surprising number of suspects will use the same password across their real account and those used for criminal purposes. Once their information appears within multiple data breaches, we can put together the pieces. I will provide many examples later.

Associate Social Network Accounts to People: Every day, my company uses breach data to connect a social network account to a real person. When a service has a data breach, the otherwise hidden email addresses and cellular telephone numbers are often disclosed along with the username for the product. This can immediately uncover the account holder's true identity. When used in conjunction with the previous two examples, we often solve entire cases from breach data alone.

Identify Telephone Number Owners: Very few of us own a landline these days which is listed in a phone book. Many of us use VOIP numbers and prepaid cellular plans in order to protect ourselves. Criminals also do this. When I want to identify the owner of a telephone number harassing a client, I assume the standard OSINT techniques will fail. However, I typically identify some valuable information about these number from within breach data.

Machine Identifiers: When my target shows up within stealer logs, I am often presented a unique hardware identifier and screen capture of their desktop. This is priceless information which would never show up within a website. Over the past year, stealer logs have been the most beneficial type of data to acquire for us.

Identify Additional Email Addresses: Once you know a target's unique username, searching that detail within breach data often reveals additional email addresses very similar to the username. This can allow us to pivot to new target data which could quickly open up an investigation.

Business Details: If you are investigating a company which has suffered a data breach, possessing the stolen data can reveal much more than any website or business listing page. Knowing the number of customers, types of encryptions, and other details can provide more value than public data.

Defense: The following are only a few defense considerations.

Identify Stolen Assets: If you are employed by a company which has suffered a data breach, you may want to track down the stolen data. This helps understand the magnitude of the breach better than any claims or demands from the criminal. I believe every victim of a data breach should investigate the incident and obtain the data being shared publicly. This is one of our most requested services by clients.

Aid Disclosure Notifications: When a company is hit with a cyber-attack, there are many laws and regulations which require notification to those impacted by the stolen data. Knowing the exact data being shared within criminal groups can help identify the proper disclosure requirements for impacted victims.

Defend Future Attacks: I rely on our daily update of leaks, breaches, logs, and ransomware in order to protect my clients. Every week, we are able to notify a potential victim of newly exposed information which could be devastating to them if abused. In many cases, we can intercept this data before the criminals begin to use it. We can then help the client block access before bad things happen.

These are only a few benefits to this type of data. I query our leaks, breaches, logs, and ransomware data every day. My staff uses it to quickly identify new leads and to protect our clients. I cannot overstate the value of this data, and the value of possessing your own offline copy.

Many online services which provide breach data queries are very limited. Most start as a free service but then move to a paid model. Your organization may have policies preventing the purchase of stolen data. Most of these services require an online account in order to conduct any queries. Your organization might prohibit this type of access, and you risk the service collecting and analyzing your target queries. Finally, many of these services eventually shut down. What would happen if your favorite breach data search engine disappeared tomorrow? By the end of this guide, I hope that you have no concern for this.

We have a lot to discuss, and I am excited to work through the process together.

CHAPTER TWO
HARDWARE CONFIGURATION

Let's have a conversation about hardware. If you begin acquiring this type of data, it can become overwhelming very quickly. Every day, our systems ingest anywhere from 50 GB to 5 TB of data. Storing and processing this amount of data can be both time consuming and resource intensive. Even if you stick to public leaks, the data can quickly fill your drives. Because of this, there are many hardware considerations. Don't worry, I offer affordable solutions.

First, your selection of computer is vital. The following may sound contradictory to previous teachings, but I no longer recommend a virtual machine for large data collections. Downloading huge files within a VM will be slightly slower than on a host machine. Decompressing large files within a VM will be much slower, and VM disk space will be quickly depleted. You could connect an external drive to the VM for storage, but the data transfer speed will be a huge bottleneck to the overall efficiency of data queries. Unless you are simply dabbling in this area, I would stay away from VMs while trying to conduct any extensive breach work.

Much like my opinions about investigative actions within ***OSINT Techniques, 10th Edition***, I also never recommend using your primary personal host computer for this type of work. I would never download breach data onto the same machine which I use for personal communications, banking, or other private tasks. This is because the type of data we will be discussing is full of bad players, malicious files, viruses, and other shady concerns. I would never jeopardize the host operating systems of my personal machines to the risks associated with stolen data.

What do I do? I possess a machine solely for breaches, leaks, logs, and ransomware. It possesses minimal applications and blazing fast hardware. When I am not actively downloading data, it is offline. However, that may be extreme for you.

In the previous print book, I stressed the importance of a dedicated OSINT computer which possessed virtual machine software for investigations. In some scenarios, I have no objection to also using this OSINT machine's host operating system for breach data collection, but there are caveats which I will explain in a moment.

In a perfect world, you have a macOS or Linux machine dedicated to breach data collection and analysis, a macOS or Linux machine dedicated to OSINT research, and another macOS or Linux machine for personal use. However, I realize that is asking a lot and I understand that we do not live in a perfect world. If your dedicated OSINT machine has Linux or macOS as the host operating system, I believe it is acceptable to use it for the techniques explained throughout this guide. I also respect that many readers will prefer Microsoft Windows due to familiarity and availability. Let's dive deeper into all of these considerations.

Apple macOS

In *OSINT Techniques, 10th Edition,* I praised macOS as a great operating system for OSINT machines. I stand by this. The blazing fast virtual machines and Android emulation cannot be beat by Linux or Windows. The hardened operating system is considered by many, including me, to be more secure than Linux or Windows (but not as private as Linux). This OS also works well for data collection, with modification.

The disk speed should be your highest priority. Once you possess hundreds of gigabytes of data, queries will take some time. As you work with larger datasets to make them more efficient to query, the disk speed will determine if a command executes within seconds, minutes, hours, or days. I never recommend a "spinning disk" hard drive, such as a standard 2.5" or 3.5" SATA drive. At a bare minimum, I recommend a newer Solid-State Drive (SSD) for the operating system. This will be used to download data and to work with active files.

I believe any MacBook Pro with a newer M series processor (M1, M2, M3) is optimal for the tasks presented here. The internal storage drives for these systems are all comparable to newer NVMe drive speeds (7,000 MB/s), and more than sufficient for our needs. Internal drives for macOS systems prior to the M series processors are fast, but typically cap at 2,500 MB/s. Much older machines which possess spinning disks or early SSDs will not be suitable. I present more on disk speed in a moment. I recommend at least 16 GB of RAM, with 32 GB preferred.

Overall, I believe a newer M Series MacBook Pro is the optimal machine for breach data collection. It will be fast and fluid, and many people are already familiar with navigating the system. However, any Linux computer comes in at a very close second place, as explained next.

Linux

If you have a Linux-capable machine available to dedicate to data collection, you will also be safe from most threats. The processor speed is not the biggest concern, but it should be considered. If you can, upgrade to a computer which is capable of housing an NVMe drive directly onto the motherboard. This small device which resembles a stick of RAM can have read and write speeds of up to 7,000 MB/s. Let's compare that to other options.

Average USB 3.0 Flash Drive:	125 MB/s
Traditional Spinning Disk Hard Drive:	135 MB/s
Solid-State Drive (SSD):	600 MB/s
PCIe Gen 3.0 NVMe Drive:	3500 MB/s
PCIe Gen 4.0 NVMe Drive:	7000 MB/s

In other words, a modern NVMe drive will provide 60 to 70 times faster data access. When you start dealing with large data sets, you will want to squeeze every drop of data bandwidth possible. Computer RAM is also important, and quite inexpensive.

The more RAM you have, the more data which can be manipulated at a faster rate. At a minimum, I believe you should possess 16 GB of RAM, but I prefer 32 GB.

I do not have any brand loyalty for Linux machines for this purpose. I care more about the hardware specifications. Any processor on a motherboard which supports PCIe Gen 4.0 NVMe Drives will probably be sufficient for the tasks. Machines without this will also perform well, you will only see a speed decrease when processing large amounts of data. For many years, I conducted this type of work without a 7,000 MB/s drive. When I began, I was using slow spinning drives. If you can, start with existing repurposed hardware and identify your own needs.

For most Linux breach data machines, I recommend the latest Long-Term Support (LTS) version of Pop!_OS or Ubuntu (I prefer Pop!_OS). They will have most of the utilities we need by default, and adding the others will be easy. They can also be installed on practically any hardware with default configuration. If you prefer another flavor of Linux, it should also work the same.

Microsoft Windows

I would never consider a Windows host computer for data collection. However, I will provide tutorials to make them work. The problems are two-fold. First, data manipulation and queries are typically slower within Windows than Linux or macOS, and configuring your system to replicate the commands within other operating systems will take some effort. However, we can get past that. My big complaint is risk. Most of the shady data which we will acquire has been stolen from Windows machines.

Often, we encounter virus files which were involved in the theft. We do not want to infect our own Windows machines while attempting to build our data collections. We encounter stealer log criminals who infect their own Windows systems and upload stored credentials every day. I do not want you to join their ranks. If you are very careful, you can make Windows work for you throughout this guide, but you have been warned. I highly recommend dedicating a Linux or macOS system for this purpose.

I know that the data I download is toxic, but I don't care. If I encounter a virus included within some stealer log data, I know it is likely targeted toward the Windows operating system and would have no impact on my macOS or Linux machines. If I encountered a rare macOS or Linux threat, I know that there is no personal information on this machine which could expose me. If the machine becomes infected, it is no big deal to wipe it out and reinstall the OS.

External Drives

I suspect many readers are not ready to commit to new hardware with multiple large internal NVMe drives for these tasks. Therefore, I want to offer some considerations which cost less. If you are just beginning your journey into data collection, consider a proper external drive for storage of your content. You can use your fast internal drive for downloading and processing of your data, and then move your final product to an external drive. This USB drive will be slower than anything internal, but we will only use it when we need to query the data. Once we build our databases which index all data, the query speed will be very similar to that of an internal drive. I would never consider a mechanical or flash drive for this process. Instead, I would rely on fast portable SSD. For most readers, I recommend a 4 TB external USB-C SSD.

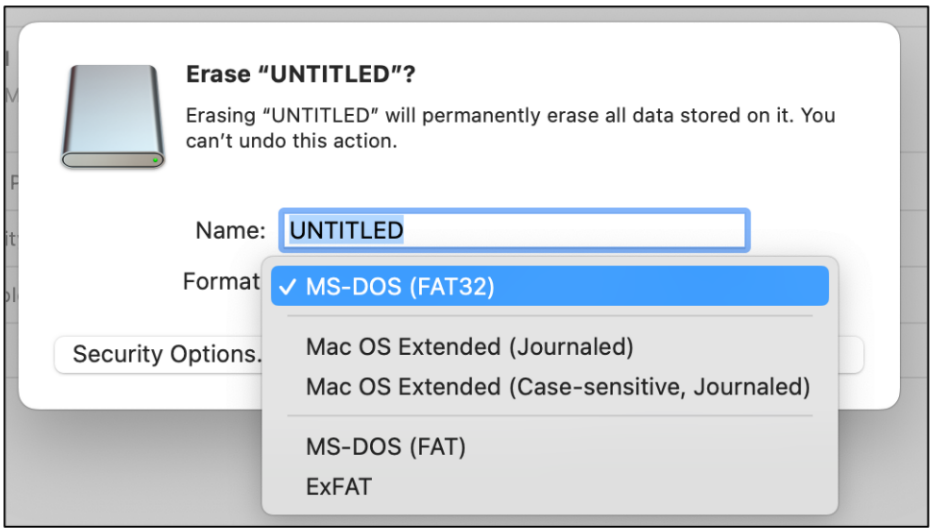
I have had great success with the SanDisk Extreme Portable SSD (amzn.to/3yPcMJ2) and the SanDisk Extreme PRO Portable SSD (amzn.to/3MKbvZl). The "Pro" version boasts twice the speed of the regular version (2,000 MB/s vs. 1,000 MB/s), but this can be misleading. Many computers do not support "USB 3.2 Gen 2x2", and your computer may only take advantage of half the advertised speed anyway. Do your research first. Even the standard version should provide 1,000 MB/s of read access, which is eight-times faster than most mechanical drives. You may encounter many reports of SanDisk SSD failures. This was a valid concern for a large batch of devices, but was then corrected with a firmware update. Some reports claim there is still a problem with these drives. Always check for firmware updates when you buy any device. I have never had any issues with mine, but I have also had success with the SAMSUNG T7 Shield 4TB SSD ($240 - amzn.to/3SBo7qM). Again, do your research.

Also consider disk space. While the $90 1 TB drive might be enticing, how quickly will you fill it up? I believe the SanDisk 2 TB Extreme Portable SSD (amzn.to/3yPcMJ2) is a great starter drive. It can be purchased for $125 and should be compatible with any modern computer. The 4 TB model (amzn.to/3sADepQ) typically costs $250, but gives you double the storage space. I will be using the SanDisk 4 TB Extreme Portable SSD (amzn.to/3sADepQ) for all of my demonstrations, but any SSD should work. Always check the USB connections. If your computer only possesses a USB-A port, then you want to make sure you buy a drive which includes this connection. I always prefer a USB-C port on the computer and direct USB-C to USB-C cable to make that connection to the external drive. Do your homework before you commit.

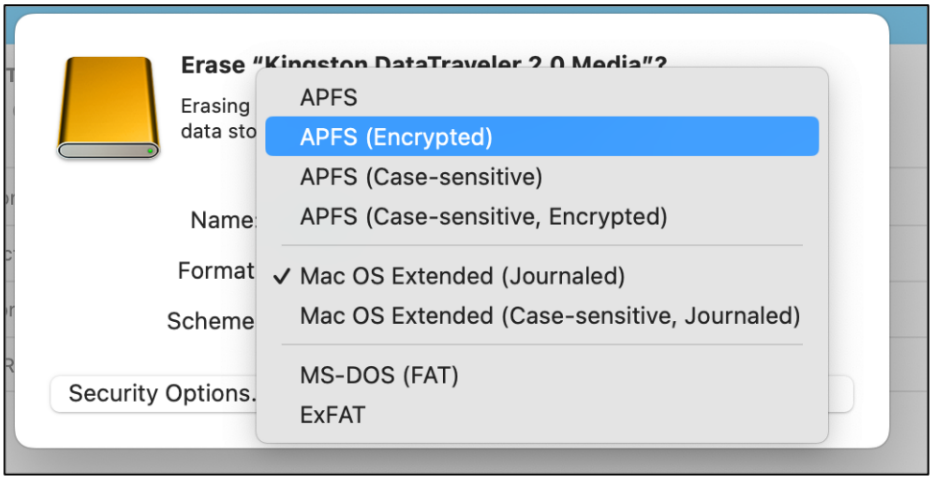
Once you have an SSD full of exciting data, you will want to make a backup. Any drive could fail, and we do not want to start over. I possess a 4 TB traditional spinning disk as my backup drive. These can be found online and in retail stores for less than $100. I will be using a Toshiba Canvio Basics 4 TB Portable External Hard Drive (amzn.to/3sBAJUk) for my demonstrations. Since it is only used for backup data, and I will never query directly from it, speed is no concern. Once you have your external drives available, they should be reformatted and encrypted. In later chapters, I explain how to use an external USB SSD for fast breach queries and a cheaper drive for backups. It is vital to encrypt the data within these drives. What if they are lost, stolen, or seized? We should protect this type of sensitive data. I always recommend encrypting drives before they store any data.

External Apple macOS Drives

Encrypting external drives within macOS is not always easy. Sometimes, macOS hides the settings we need to protect an external device. As an example, I inserted a USB SSD which was formatted as "FAT32", which is common for universal drive access. I wanted to erase the drive and encrypt it. However, the Disk Utility application (Applications > Utilities) only displayed the following options. Right-clicking the drive in Finder also did not present an option to encrypt the drive.



The first step to take within the Disk Utility application is to select "Show All Devices" under the "View" menu. Next, select the device (not the formatted volume) within the left menu and click the "Erase" button. This may still only present volume formats which cannot be encrypted. Be sure to change the "Scheme" to "GUID Partition Map". You should now see an option of "APFS (Encrypted)" under "Format". This option will encrypt the entire external drive with macOS encryption. I believe this is the best option for users who will only need to access this drive from a macOS system. If you want to follow along with my tutorials and use my script, make sure to name the drive "DATA" and any backup drive "BACKUP".



External Linux Drives

Linux systems are more user-friendly for encrypting external drives. The following applies to Pop!_OS and Ubuntu, but the steps should be similar for most Linux systems.

- Launch "Files" and right-click your external drive.
- Select "Format" and provide a name of "DATA" for your SSD.
- Choose "Internal disk for use with Linux systems only (Ext4)".
- Select the "Password protect volume" option.
- Click "Next" and supply a secure password when prompted.
- Confirm the password when prompted.
- Click "next" until you reach the "Format" option and click it.
- Repeat the process for your backup drive, but name it "BACKUP".

When you encrypt the drives, the process may take some time to complete. You will need to unlock it with the password every time it is inserted into your machine.

Missing Linux Disk Space

Many Linux users may notice that their external hard drives are not as big as advertised. By default, EXT4-formatted disks reserve 5% of the space for internal system usage. This is an archaic setting which is no longer relevant to today's modern drives. If you would like to regain this disk space, consider the following.

First, identify your drive details with the following Terminal command.

```
lsblk
```

You should see something similar to "`sda1`" or "`sda2`" which possesses the partition size of your external drive. Mine was "`sda1`". Apply your identifier to the following command.

```
sudo tune2fs -r 0 /dev/sda1
```

This should recover 5% of the drive to your available usage.

External Windows Drives

This is where things get tricky. If you have a professional version of Windows, then you should be able to right-click the drives within Explorer and "Turn On BitLocker". This is always the preferred option. If BitLocker is not available on your system, you could encrypt the entire disks with VeraCrypt, but this can present more problems. I have seen many drives with full-disk encryption via VeraCrypt within Windows refuse to unlock, and then have severe speed issues whenever things properly functioned. This is yet another reason I prefer to stay away from Windows for this type of work.

Hardware Summary

Is this all overkill? Possibly. It depends on how much you rely on this type of data. Could you get by with just using more traditional hardware for this task? Absolutely. It will only take more time to download, decompress, process, clean, store, and query the data every day.

If you begin to rely on this data often, you might find yourself looking at faster machines with embedded ultra-fast storage in order to make the process more tolerable. A MacBook Pro with 4 TB of internal 7,000 MB/s storage is a luxury, but you will pay for it. A Linux machine with a 4 TB internal NVMe drive will be more affordable, and just as fast. However, I want to be realistic. The current most affordable MacBook Pro with 4 TB of internal storage is $3700. A 4 TB External SSD is less than $250. The latter is more approachable.

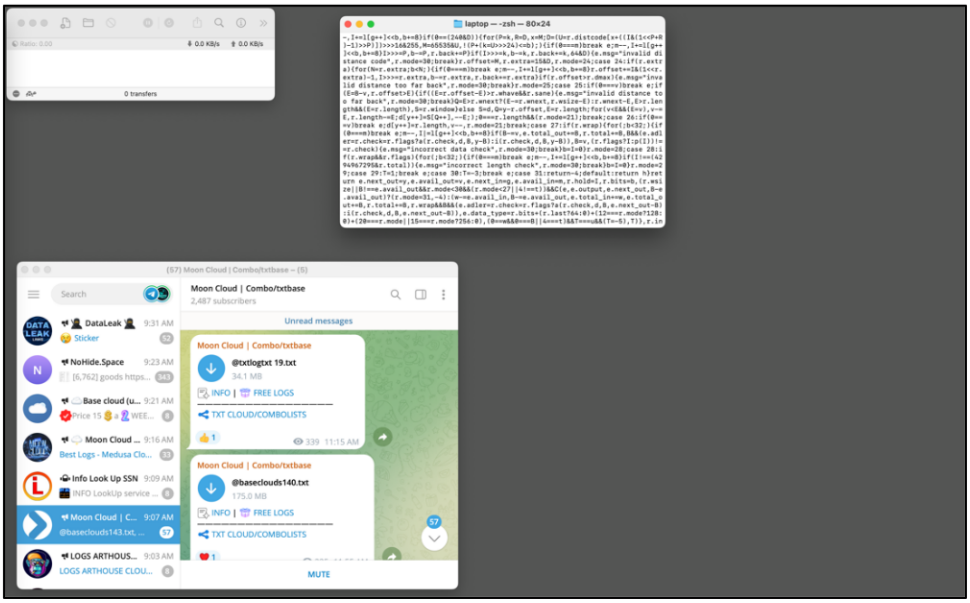
There is no harm using older equipment in order to understand the strategies presented within this section. The limit to your resources will become apparent as soon as you start dealing with large data sets.
As an example, I downloaded 250 GB of compressed stealer logs to an older Intel-based MacBook Pro laptop with a 2,100 MB/s internal SSD. The decompression of the data took over an hour. Parsing the passwords from the text files took another four hours. I replicated the exact same process on both a new MacBook Pro and a new Linux machine with a PCIe Gen 4.0 NVMe Drive. The decompression on each took a few minutes, and the extraction of passwords took less than ten minutes.

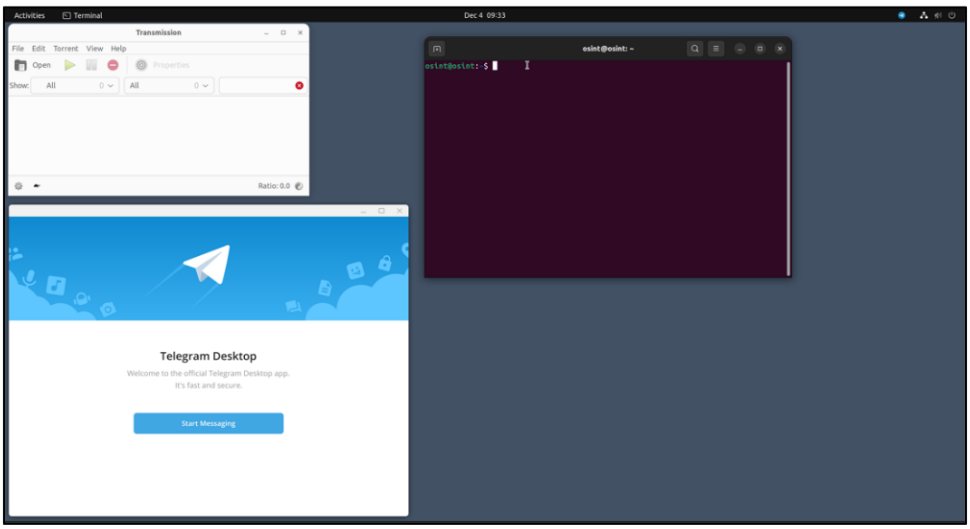
I deal with enough data to justify the expense of a dedicated machine for this purpose. When we start talking about stealer logs, the vast number of files alone can be a bottleneck on most machines. It is common to extract over two million very small files within a single compressed archive.

I will now assume that you have chosen a computer, operating system, external drive for storage, and external drive for backups. We can now begin configuring our software.

CHAPTER THREE
SOFTWARE CONFIGURATION

Once you have your desired computer, you should configure the software required for the lessons within this guide. If you are using a Linux machine for these tasks, you already have most of the utilities needed for our commands. Programs such as cut, sed, and awk are included within most Linux systems. However, others are not. This chapter will separate all required configurations by operating system. I will tackle all necessary modifications to macOS, Linux, and Windows in their own sections. Once you have completed this chapter, all remaining tutorials should work the same on each operating system. However, your desktop may appear different than mine. Below is an example of graphical differences between macOS and Ubuntu.





Apple macOS

The first application I install on any new macOS operating system is a package manager called Homebrew, often shortened to Brew. This application is very beneficial when there is a need to install utilities which would usually already be present on a Linux computer. It also simplifies installation of applications which would otherwise require manual download or access to Apple's App Store. Brew is my favorite software for macOS computers. The easiest way to install Brew is to visit the website brew.sh then copy and paste the following command into the Terminal application (Applications > Utilities > Terminal). After completion, you are ready to use Brew to install and update applications.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.
com/Homebrew/install/HEAD/install.sh)"
```

You will likely receive a notice from macOS that you need to install "Developer Tools". Click "Install" and "Agree", then allow the process to complete. After Brew installation is complete, you will likely be presented with one or two commands which need to be manually executed within Terminal. My installation presented the following notice.

```
==> Next steps:
- Run these two commands in your terminal to add Homebrew to your PATH:
    (echo; echo 'eval "$(/opt/homebrew/bin/brew shellenv)"') >> /Users/ventura/.zprofile
    eval "$(/opt/homebrew/bin/brew shellenv)"
```

This is unique to my installation, as my chosen username was "ventura" at the time. Copy any commands presented here and paste them within the same Terminal window, executing each by striking return. Let's test everything with a few commands.

- `brew doctor` - This command confirms that Brew is configured properly and that all paths are set. You should receive a notice that "Your system is ready to brew".
- `brew update` - This command checks for any pending updates to Brew itself. You should receive a response of "Already up to date".
- `brew upgrade` - This command updates any installed programs. You should receive no response since we have not installed anything.
- `brew analytics off` - This command disables Brew's embedded analytics which monitor the number of times an application is installed using Brew. These metrics are only used to understand how users interact with the product, but I prefer to limit my exposure.

If everything is working, you are now ready to use Brew as a software installation repository. Treat this as a replacement for the App Store, but it does not require an Apple ID. If you are using a macOS host which already possesses Brew, and several other applications as discussed in *OSINT Techniques, 10th Edition*, your results will appear slightly different than mine. However, there is no conflict with these tutorials and those within the book. Let's use Brew to install our first application.

```
brew install wget
```

This command ensures the Wget is installed, which we will use to download data from the internet within terminal. Next, we should install Ripgrep. This is a faster version of Grep, which queries any file or folder for specific text. We will use this to conduct general queries within specified large data sets when we need quicker results than a standard text search.

```
brew install ripgrep
```

While Ripgrep is much faster than a standard Grep query, it has limitations. We will install Qgrep to take our queries to the next level. Qgrep allows us to create compressed databases of our data and receive search results in a fraction of the time Ripgrep takes to go through all of the data. First, navigate to the following URL within your browser.

https://github.com/zeux/qgrep/releases/tag/v1.3

Download the most recent "macos" zip file to your default "Downloads" folder. Then, execute the following within Terminal.

```
mkdir ~/Documents/Qgrep
cd Downloads
unzip qgrep* -d ~/Documents/Qgrep
chmod +x ~/Documents/Qgrep/qgrep
```

The first command creates a new directory titled "Qgrep" within the Documents folder of the current user. The second command navigates to the Downloads folder. The third command unzips the file which you downloaded, and places the data into the new folder you created in your Documents. The final command makes the Qgrep file (program) executable.

Apple systems already possess sed, awk, cut, and many other commands which we will use throughout this guide. However, we should not use these default applications. The macOS versions of these programs are slower, less robust, and possess fewer features than the traditional Linux versions. Therefore, we need to download better options and make them the default for our usage. Let's start with "Sed". This program will be essential for replacing, removing, and modifying text within large data files. The following command installs the application.

```
brew install gnu-sed
```

You can now use Sed, but you must use the command "gsed" if you want to use the "GNU" version which is identical to the default Linux options. Executing "sed" within Terminal will still rely on the macOS version, which is undesired. The following command will tell us how to change this.

```
brew info gnu-sed
```

You should receive something similar to the following.

```
GNU "sed" has been installed as "gsed".
If you need to use it as "sed", you can add a "gnubin"
directory to your PATH from your bashrc like:

PATH="/opt/homebrew/opt/gnu-sed/libexec/gnubin:$PATH"
```

Copy and paste the "PATH" line into Terminal and press Return. You should now have "sed" as the new version, but let's test with the following command.

```
sed --version
```

The result should include something similar to "sed (GNU sed) 4.9". If you see an error, you are still using the macOS version. However, we have a new problem. If you were to completely close Terminal and open a new instance, Apple's version of Sed will become the default again. This is quite an annoyance, but easily repaired. Instead of simply executing the previous "PATH" command, we must create a hidden file in your home directory called .zshrc (assuming you are using the latest version of macOS). We can do this with the following command.

```
touch ~/.zshrc
```

Next, we must open this file inside of a text editor to modify it with the following command.

```
open -a TextEdit ~/.zshrc
```

Finally, paste the previous PATH command into this file and save it, but do not close it. We should now be able to close and open Terminal and see that the new version of Sed is the default. Let's replicate this process and install a better version of Awk, which is a scripting language used for manipulating data. We will use it to sort our large stealer logs later. First, take a look at the default version with the following command.

```
awk --version
```

The result should appear similar to the following.

```
awk version 20230909
```

This confirms we are using the default macOS version of Awk. The following command installs the GNU version of Awk.

```
brew install gawk
```

We can now execute this new version of Awk with "gawk" inside of Terminal. However, "awk" will still rely on the macOS version. The following command will tell us how to change this.

```
brew info gawk
```

You should receive something similar to the following.

```
GNU "awk" has been installed as "gawk".
If you need to use it as "awk", you can add a "gnubin"
directory to your PATH from your ~/.bashrc and/or ~/.zshrc
like:

PATH="/opt/homebrew/opt/gawk/libexec/gnubin:$PATH"
```

Paste this PATH command on a new line into the .zshrc file which we previously created and save the file. Close and reopen Terminal and you should now have "awk" as the new version, but let's test with the following command.

```
awk --version
```

The result should include something similar to "GNU Awk 5.3.0". If you see anything similar to "awk version 20230909", you are still using the macOS version. Let's do this again with the Cut command, which will be vital for extracting columns of data later.

The following command installs the GNU version of Cut, along with many other valuable programs.

```
brew install coreutils
```

We can now execute this new version of Cut with "gcut" inside of Terminal. However, "cut" will still rely on the macOS version. The additional programs within this repository will also require you to append "g" before each execution. The following command will tell us how to change all of these utilities to the default GNU version.

```
brew info coreutils
```

You should receive something similar to the following.

```
Commands also provided by macOS and the commands dir,
dircolors, vdir have been installed with the prefix "g".
If you need to use these commands with their normal names,
you can add a "gnubin" directory to your PATH with:

PATH="/opt/homebrew/opt/coreutils/libexec/gnubin:$PATH"
```

Paste this PATH command on a new line into the .zshrc file which we previously created and save the file. Close and reopen Terminal and you should now have "coreutils" as the new version, but let's test with the following command.

```
cut --version
```

The result should include something similar to "cut (GNU coreutils) 9.4". If you see an error, you are still using the macOS version. This should also make the GNU version of Sort and other utilities the default. Perform one last test with the following command.

```
sort --version
```

The result should include something similar to "sort (GNU coreutils) 9.4".

At this point, my .zshrc file appeared as follows. If you are using a macOS host which was used to create an OSINT build following the previous book, you may see extra lines in the file, which are fine.

```
PATH="/opt/homebrew/opt/gnu-sed/libexec/gnubin:$PATH"
PATH="/opt/homebrew/opt/gawk/libexec/gnubin:$PATH"
PATH="/opt/homebrew/opt/coreutils/libexec/gnubin:$PATH"
```

When we start applying our techniques to automated scripts, we will need a program called Zenity to present a selection menu. It can be installed in macOS with the following command (Zenity is already present within Pop!_OS and Ubuntu).

```
brew install zenity
```

FreeFileSync

Later in this guide, I will explain the importance of data backups. For now, install our synchronization program, FreeFileSync, with the following steps.

- Navigate to https://freefilesync.org/download.php.
- Download the macOS version.
- Double-click the downloaded file to extract the installer.
- Double click the "FreeFileSync" package.
- Complete the installation with the default options.

Telegram

This is the only program which I have reservations about. While it will be vital for our acquisition of data, it also presents mild risk. Telegram is the wild west. Anything goes. There is no content moderation or policing of any kind. You can find anything there, even things which should not be found. I do not believe the Telegram application itself is harmful, but there is no lack of viruses available to download within it. This is why I prefer to conduct data collection on an isolated machine away from my personal documents and financial logins. Telegram can be installed with the following Homebrew command.

```
brew install --cask telegram
```

Transmission

Many large data sets will only be available as a torrent file through the BitTorrent file distribution system. Transmission is our best software for this purpose. It can be installed within macOS with the following command.

```
brew install --cask transmission
```

jq

The application jq is a lightweight and flexible command-line JSON processor. We will later use it to parse long JSON entries into a single line of data. It can be installed within macOS with the following command.

```
brew install jq
```

Tor Browser

Your stock web browser can only access public websites. This web browser can do the same, but can also access sites on the Tor network. When we get into ransomware conversations and data collection, this will be required. It can be installed with the following command.

```
brew install --cask tor-browser
```

Python

Your macOS system should now have Python ready within Homebrew, but we can make sure with the following commands

```
brew install python
brew unlink python && brew link python
python3 -m pip install pip
```

BBEdit

I highly recommend the free version of BBEdit (barebones.com/products/bbedit) for all macOS users. It opens incredibly large files and allows you to easily view and modify the content. I know of no other text viewer which will open a 5 GB file in seconds and allow immediate find and replace functionality.

Linux

Linux users are all set on most of the software requirements, but there are a few programs which we will need to install. We should first make sure our system is updated and then confirm that we have two essential programs installed with the following commands within Terminal. I highly recommend that you add a shortcut to Terminal within your Linux Dock. For Pop!_OS and Ubuntu, you can do this by clicking the nine dots in the Dock; navigating to Terminal; right-clicking the Terminal icon; and selecting "Add to Favorites".

```
sudo apt update
sudo apt upgrade -y
sudo apt install -y wget
sudo apt install -y curl
sudo apt install -y python3-pip
```

The first two commands update our Linux software. The third command ensures the Wget is installed, which we will use to download data from the internet within terminal. The final command installs Curl, which will also help us retrieve data within Terminal.

Next, we should install Ripgrep. This is a faster version of Grep, which queries any file or folder for specific text. We will use this to conduct general queries within specified large data sets when we need quicker results than a standard text search.

```
sudo apt install -y ripgrep
```

While Ripgrep is much faster than a standard Grep query, it has limitations. We will install Qgrep to take our queries to the next level. Qgrep allows us to create compressed databases of our data and receive search results in a fraction of the time Ripgrep takes to go through all of the data. First, navigate to the following URL within your browser.

https://github.com/zeux/qgrep/releases/tag/v1.3

Download the most recent "ubuntu" zip file to your default "Downloads" folder. Then, execute the following within Terminal.

```
mkdir ~/Documents/Qgrep
cd Downloads
unzip qgrep* -d ~/Documents/Qgrep
chmod +x ~/Documents/Qgrep/qgrep
```

The first command creates a new directory titled "Qgrep" within the Documents folder of the current user. The second command navigates to the Downloads folder. The third command unzips the file which you downloaded, and places the data into the new folder you created in your Documents. The final command makes the Qgrep file (program) executable.

FreeFileSync

Later in this guide, I will explain the importance of data backups. For now, install our synchronization program, FreeFileSync, with the following steps.

- Navigate to https://freefilesync.org/download.php.
- Download the Linux version.
- Right-click the downloaded file within Files and select "Extract Here".
- Double-click the new FreeFileSync folder.
- Right-click the FreeFileSync file and choose "Run".
- Complete the installation with the default options.

Note that FreeFileSync does not play well with ARM-based processors, but a typical Linux Intel or AMD laptop should not have any issues. Some software app stores have FreeFileSync ready to install, which is much easier.

Telegram

This is the only program which I have reservations about. While it will be vital for our acquisition of data, it also presents mild risk. Telegram is the wild west. Anything goes. There is no content moderation or policing of any kind. You can find anything there, even things which should not be found. I do not believe the Telegram application itself is harmful, but there is no lack of viruses available to download within it. This is why I prefer to conduct data collection on an isolated machine away from my personal documents and financial logins. If you are using Ubuntu, you can install Telegram with the following command.

```
sudo snap install telegram-desktop
```

If you are using a Linux system without Snap, then the following applies.

- Navigate to https://desktop.telegram.org.
- Expand the "Show All Platforms" option.
- Click "Get Telegram for Linux" to download the file.
- Right-click the downloaded file within Files and select "Extract Here".
- Double-click the "tsetup" and "Telegram" subfolder.
- Right-click the Telegram file and choose "Run".

Transmission

Many large data sets will only be available as a torrent file through the BitTorrent file distribution system. Transmission is our best software for this purpose. It can be installed within Pop!_OS and Ubuntu with the following command.

```
sudo apt install transmission
```

jq

The application jq is a lightweight and flexible command-line JSON processor. We will later use it to parse long JSON entries into a single line of data. It can be installed within Linux with the following command.

```
sudo apt install -y jq
```

Tor Browser

Your stock web browser can only access public websites. This web browser can do the same, but can also access sites on the Tor network. When we get into ransomware conversations and data collection, this will be required. The Linux installation of Tor Browser changes often. Always follow the latest instructions on their official website at https://tb-manual.torproject.org/installation/.

Microsoft Windows

I want to stress again that conducting breach data work within Windows is risky. It is also more complicated. While the following steps will configure your system to replicate most of the techniques within the remaining chapters, expect occasional hurdles. The experience with Windows will never be as fluid as Linux or macOS. During the Mac setup, we used Homebrew as a package manager. For Windows, we will use Chocolatey. It can be installed with the following steps.

- Click the Windows menu button (lower-left) and type "powershell".
- Right-click on Windows PowerShell and select "Run as Administrator".
- Enter the following command and press Enter (Confirm if required).

```
Set-ExecutionPolicy AllSigned
```

- Execute the following single command within PowerShell.

```
Set-ExecutionPolicy Bypass -Scope Process -Force;
[System.Net.ServicePointManager]::SecurityProtocol =
[System.Net.ServicePointManager]::SecurityProtocol -
bor 3072; iex ((New-Object
System.Net.WebClient).DownloadString('https://communit
y.chocolatey.org/install.ps1'))
```

If everything is working, you are now ready to use Chocolatey as a software installation repository. You do not need to open PowerShell any more. We will use Command Prompt to install our desired applications.

- Click the Windows menu button (lower-left) and type "cmd".
- Right-click on Command Prompt and select "Run as Administrator".
- Enter the following commands and press Enter after each.

```
choco install -y wget
```

This command ensures the Wget is installed, which we will use to download data from the internet within terminal. Next, we should install Ripgrep. This is a faster version of Grep, which queries any file or folder for specific text. We will use this to conduct general queries within specified large data sets when we need quicker results than a standard text search.

```
choco install -y ripgrep
```

While Ripgrep is much faster than a standard Grep query, it has limitations. We will install Qgrep to take our queries to the next level. Qgrep allows us to create compressed databases of our data and receive search results in a fraction of the time Ripgrep takes to go through all of the data. First, navigate to the following URL within your browser.

https://github.com/zeux/qgrep/releases/tag/v1.3

Download the most recent "windows-x64" zip file to your default "Downloads" folder. Then, execute the following within Command Prompt.

```
mkdir C:\users\%username%\Documents\Qgrep
cd C:\users\%username%\Downloads
choco install -y unzip
unzip qgrep* -d C:\users\%username%\Documents\Qgrep
```

The first command creates a new directory titled "Qgrep" within the Documents folder of the current user. The second command navigates to the Downloads folder. The third command installs the unzip program. The fourth command unzips the file which you downloaded, and places the data into the new folder you created in your Documents.

Windows systems do not possess sed, awk, cut, and many other commands which we will use throughout this guide. The following commands install the applications which will replicate the actions within the previous Linux and macOS tutorials.

```
choco install -y sed
choco install -y awk
choco install -y curl
choco install unxutils
```

FreeFileSync

Later in this guide, I will explain the importance of data backups. For now, install our synchronization program, FreeFileSync, with the following command. You may be required to manually confirm the installation and then exit any popup window.

```
choco install -y freefilesync
```

Telegram

This is the only program which I have reservations about. While it will be vital for our acquisition of data, it also presents mild risk. Telegram is the wild west. Anything goes. There is no content moderation or policing of any kind. You can find anything there, even things which should not be found. I do not believe the Telegram application itself is harmful, but there is no lack of viruses available to download within it. This is why I prefer to conduct data collection on an isolated machine away from my personal documents and financial logins. Telegram can be installed with the following Chocolatey command.

```
choco install -y telegram
```

Transmission

Many large data sets will only be available as a torrent file through the BitTorrent file distribution system. Transmission is our best software for this purpose. It can be installed within Windows with the following command.

```
choco install transmission
```

jq

The application jq is a lightweight and flexible command-line JSON processor. We will later use it to parse long JSON entries into a single line of data. It can be installed within Windows with the following command.

```
chocolatey install jq
```

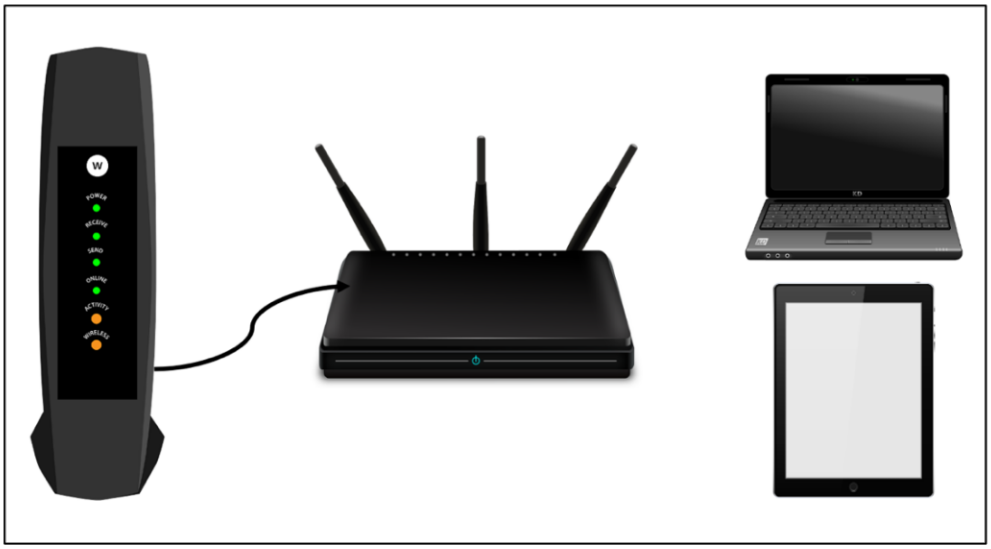
Tor Browser

Your stock web browser can only access public websites. This web browser can do the same, but can also access sites on the Tor network. When we get into ransomware conversations and data collection, this will be required. The Windows installation executable can be found at https://tb-manual.torproject.org/installation/.

Virtual Private Network (VPN)

This is a vital layer of protection when working with leaks, breaches, and logs. We should start with an extended understanding of the Virtual Private Network, which I will only present as VPN for the rest of this guide. Let's work through the "what" before we tackle the "why" and "how".

First, let's take a visual look at a traditional home network configuration without any protection. In the image below, your home internet connection begins at the modem, which could be a fiber, cable, satellite, or DSL connection. It is the first device within the home which accepts data from your provider and makes it available to your devices. From there, most people possess a Wi-Fi router which wirelessly broadcasts the availability of internet access to any other device in the home. Any device connected to your internet service is using the same public Internet Protocol (IP) address. When your laptop, tablet, or any other internet-capable device connects to any website or service, it shares the same public IP address assigned to your home internet connection. In many cases you are the only person in the world using this IP address at any given time.



I estimate that 99% of households possess a similar scenario to this example. Some may argue that there is no threat in sharing your true IP address with every site you visit and service you use. I disagree. While a true IP address does not disclose the home address of the user directly, it does present numerous threats, as outlined next.

- **Internet Activity:** Assume that I am suing you through civil court, and I have convinced a judge to grant me a court order to collect your internet activity. Since I know where you live, I can assume the provider of your internet service. A court order could be issued to your ISP for your internet activity. If your ISP logs your traffic, which most do, the response would tell me every domain which you visited and the dates and times of occurrence. I could use this to prove you were visiting specific websites or transmitting large amounts of data

to designated services. I have witnessed child custody disputes enter online history as evidence, which was then presented without any context to discredit a parent's abilities to care for a child.

- **Search Queries**: When you connect to Google and conduct a search for "inteltechniques", the response URL presented to you, including the search results from the query, is https://www.google.com/search?q=inteltechniques. Does your Internet Service Provider (ISP) know you conducted a search on Google? Yes. Do they know you searched for "inteltechniques"? No. This is because Google encrypts the actual search URL. The provider of your internet connectivity can only see the domain name being accessed. It cannot see any details about specific pages or any credentials entered. This is why https versions of websites are so important. Your browser can see this entire URL, but it does not directly share any details with your provider. However, if a site does not include proper SSL protocols, then any search query on that site could be captured by your ISP.
- **Location**: Next, assume I want to know where you live. I know your email provider is Gmail, and a subpoena to them would reveal your IP address at a specific date and time. If this IP address belongs to your internet service provider, a second subpoena will disclose the address of service (your home). This could be applied to any website you have ever visited.
- **Fingerprinting**: Every website you visit collects and stores your IP address. If you are the only person in the world with that address, they know when you return to the site and any activity conducted. They know every click you make, and attribute that to you. This is one way so many websites seem to know what you are searching, buying, and discussing before you provide the full details within your devices.
- **Download History:** Shady law firms monitor questionable files such as pirated movies, music, and other media. Once an IP address is seen downloading content without authorization, they issue subpoenas to identify the home with the offending connection. They then issue threats of lawsuits unless an extortion is paid. Does your nephew use your home Wi-Fi without supervision at any time? You could be liable for any of his activity.
- **Breach Data**: Every day, services are breached and the databases they possess are published online. Almost all of these include data containing the home IP address of the user. If you have followed my other guides to possess a private home which is not publicly associated with your name, this could unravel all of your hard work. I can search your name within breach data and see your true home IP address. I can then use the previous methods to discover your home address. If you have multiple accounts in alias names, I can tie them all together thanks to your unique public IP address.

If you believe any of this could be a threat to you, then you need a properly configured VPN. VPNs provide a good mix of both security and privacy by routing your internet traffic through a secure tunnel. The secure tunnel goes to the VPN's server and encrypts all the data between your device and that server. This ensures that anyone monitoring your traffic before it reaches the distant server will not find usable, unencrypted data. Privacy is also afforded through the use of a distant server. Because your traffic appears to be originating from the VPN's server, websites will have a more difficult time tracking you, aggregating data on you, and pinpointing your location. Let's now revisit the previous threats with the assumption a VPN was used.

- **Internet Activity:** Your ISP cannot see your internet activity when a VPN is used. They only see that a connection was made to the VPN, and then all traffic is encrypted. They have no log of your online activity. Reputable VPN companies have a "No Logging" policy which prevents them from storing the IP address assigned to you at any given time. They would be unable to identify your traffic from everyone else.
- **Search Queries:** After connecting to your VPN, you conduct the same search as before. Does your ISP know you conducted a search on Google? No. Does your VPN provider know you conducted a search on Google? Yes. Does your VPN provider know you searched for "inteltechniques"? No. If you encounter a website without proper SSL, your queries will be visible to the VPN provider, but not attributed directly to you.
- **Location**: VPNs offer numerous server locations which you can select and change at any time. You can make your website traffic appear to be occurring from London, New York, Los Angeles, Australia, or any location in between. No online service will ever know your true location.
- **Fingerprinting**: The IP address provided by the VPN will be shared with hundreds or thousands of other users at any given time. However, websites will then rely on other ways to try to track you. We will tackle this later.
- **Download History:** When a law firm subpoenas your VPN IP address to begin their extortion campaign, they will discover the owner of the address is a VPN company which cannot provide the information they need. They will move on to the next victim.
- **Breach Data**: When you appear in the next data breach, the IP address associated with your account will be a VPN provider, and that address will be useless to anyone wanting to use this information in a malicious manner.

VPNs are not a perfect anonymity solution. It is important to note that VPNs offer you privacy, not anonymity. The best VPNs for privacy purposes are paid subscriptions with reputable providers. There are several excellent paid VPN providers out there and I strongly recommend them over free providers. Free providers often monetize through very questionable means, such as data aggregation. Paid VPN providers monetize directly by selling you a service, and reputable providers do not collect or monetize your data. Paid providers also offer a number of options which will increase your overall privacy and security.

I think I have worked through the "what" and "why", it is now time to tackle the "how". This is where you must select a VPN provider. If you already possess a VPN service which you like, then you should proceed with that option. Please do not change providers solely because of my preference. However, please be informed of my considerations when choosing a VPN provider and ensure that your selection passes all of the tests.

Recommending a VPN provider today is similar to claiming a preference for the best version of Linux. No matter what I say, I will offend someone. Please note that any reputable VPN provider is better than none at all. However, I do believe there are some much better than others. Okay, enough beating around the bush. I currently use and recommend Proton VPN as my exclusive VPN provider, and almost all of my clients possess a Proton VPN account. Please allow me to explain my opinions, and my reasons for not recommending your favorite service. Overall, I mostly care about the following categories when choosing a VPN provider.

- **No Logging:** As stated previously, most reputable VPNs offer a "No Logging" policy which prevents them from saving logs about customer usage. However, some VPN companies claim this logging policy without following it. To be fair, the idea of absolute zero logs is a myth. There must be some sort of logging of connections for the service to function. I care mostly about whether the service stores these logs and has access to them when demanded. Services such as PureVPN have been caught giving away logs of user activity when demanded by court order, and breaches have disclosed that other services such as Fast VPN store user data indefinitely. There is no way to truly know the logging of your VPN data, so we should all monitor any news about this data being released. Proton VPN (and many others) have never had a known exposure of user logs. Proton VPN's logging policy can be found online at https://protonvpn.com/support/no-logs-vpn.
- **Audits:** This is where we can have some comfort. Since we are not able to monitor VPN servers directly, we must rely on third-party audits of services. Any reputable VPN provider will not only hire companies to audit their service, but will also publicly share those audits with the world. In April of 2022, Proton VPN announced that they hired Securitum to conduct a full audit of their logging practices. Proton shares the audits for all of their products (Mail, VPN, Calendar, and Drive) on their official website located at https://proton.me/blog/security-audit-all-proton-apps. I never trust any company to abide by their rules. I place more trust in the third parties allowed to access the code.
- **Open Source:** When VPN companies provide their application's source code publicly, it allows anyone to examine the code for any malicious intent. I do not have the abilities to do this myself, but I appreciate that many other people much smarter than I am are scrutinizing the code of these services. However, we never truly know if the open-source code is the same as what is being used in the live environment. This is why those audits by third parties are so important. Publicly disclosing the code of a VPN application is a nice layer, but I do not care about that as much as the other categories presented here.

- **Jurisdiction:** This will vary for every reader. You might want to consider the legal jurisdiction of your provider. Many privacy purists are very picky about the location of a VPN company's headquarters. Do you live in the United States and worry that your government will execute federal court orders to obtain your activities? Then you may not want to choose a U.S. service (or any service within cooperating jurisdictions). Proton VPN is hosted in Switzerland, and they only respond to Swiss court orders. They cannot disclose any user activity, but they could disclose payment details or account identifiers if forced. However, I believe that we place too much emphasis on jurisdiction. If you are using a U.S. server, there could always be infiltration regardless of the jurisdiction. Any country could decide to cooperate with your country at any time. I would rather rely on a Swiss company than a Russian or Chinese provider which may not be following any rules. As a U.S. citizen, I do prefer my provider to be outside of U.S. court order authorization, especially for civil cases. However, I am not naive. If my government placed all of their power into investigating me, I am sure that Swiss (or any other) courts would not protect me. My chief threat is not the government. It is data breaches, ISPs, and online services. If you are truly worried your government is monitoring you at all times, a VPN will not save you.
- **Ownership:** Who owns your chosen VPN service? Is it a small independent company with a handful of employees or a large conglomerate which owns 20 VPN brands? The first option may seem better since only a small group of people can access your data, but some may find the second option better to disappear within the thousands of other users. I prefer something in between. I prefer the VPN company to be independently-owned and not a brand under a larger VPN umbrella company. However, I also want a large user base to exist so that my traffic can disappear within all of the other activity. Proton VPN works for me.
- **Advertising:** If you search for "VPN reviews" online, you will immediately find numerous "unbiased" review sites. However, if you look closely, you will see something peculiar. The same handful of VPN providers seem to make the list every time. Also, these providers are never the services commonly used within the privacy communities. This is because these are mostly paid placements. In some cases, large VPN companies own the entire website and simply recommend their own products. I ignore all VPN review sites. I also ignore any providers which participate in this activity. This is another reason I prefer Proton VPN. They do not create fake review sites to push their product.
- **Connection Options:** Most VPN providers offer several servers within numerous countries. This is not unique to Proton VPN, but I am happy with their selection.
- **Firewall Capabilities:** Most reputable VPN providers allow their product to be used within a home firewall. If you are unsure, look for tutorials from your chosen provider by searching the VPN name along with "pfSense" or "OpenVPN". Within my testing, Proton VPN works very well with firewall software.

I should now explain my reasons for choosing Proton VPN over other respected providers. The first to discuss is Mullvad. Proton VPN and Mullvad are commonly the most recommended VPN providers within various privacy communities. I believe the privacy policies of Mullvad are great, and I have no concern over their presence in Sweden. My issue is with the reliability of the service. I tested Mullvad in late 2021 and late 2022. In 2021, I experienced slow speeds and dropped connections while using their official application. In 2022, this seemed to have been fixed, but I could not maintain a reliable connection within my firewall via OpenVPN. Many others have complained of the same failures online. A VPN is no good if it fails. If you use Mullvad and have no issues, I see no reason for you to change. Since I have experienced bad results with their service and support, I choose Proton VPN. I also do not like that anyone can brute force Mullvad user numbers to access an account without password.

I should address a very important disclosure. Most VPN companies offer an affiliate program which rewards people for introducing their product to new users. I have had an affiliate partnership with both Proton VPN and PIA for several years. If you use my referral link at **https://go.getproton.me/aff_c?offer_id=26&aff_id=1519**, Proton knows that you were referred by me, and I receive a small one-time financial reward. I receive absolutely no details about you or your order.

Some will say that this affiliate payment is the only reason I recommend their product. My response to that is three-fold. First, I would not risk my reputation recommending a product which I do not use and trust. Second, PIA was paying me more for a referral than Proton VPN, and I have stopped recommending PIA for many users. Finally, another VPN company offered me the highest reward ($60) for every referral, but I would never use their product. Therefore, I declined the offer.

I would recommend Proton VPN without the affiliate partnership, but I want to be transparent about our relationship. I also allow these affiliate payments to directly support our efforts to keep this guide updated. If you sign up for Proton VPN and want to support my show, please use the previous referral link. If you cringe at the idea of using any referral link, then you can find the absolute same pricing by simply purchasing from protonvpn.com, and I receive nothing. I was not paid anything by Proton VPN for inclusion in this guide. I maintain a page on my website which always displays my VPN preferences at **https://inteltechniques.com/vpn.html**. It also contains any affiliate links.

When purchasing a VPN service, you will need to make payment online. This is tricky because we want a VPN to hide our traffic, but then we have to tell the company something about us when we make the payment. This leaves a digital trail which could be tracked back to us. Therefore, you need to consider your own threat model when paying for a VPN service.

I prefer to pay via Bitcoin from my offline software wallet stored on my computer. There is no Bitcoin exchange involved and I can provide any name desired for the VPN. Proton maintains a website for instructions to pay for service via Bitcoin on their site at https://protonvpn.com/support/vpn-bitcoin-payments.

If you have the ability to pay via Bitcoin from a local wallet, I believe you should. However, that does not make you magically invisible. You will still be connecting from your home internet connection, so the VPN provider will always see that unique identifier. You will also be paying bills in your name, sending email from your account, and conducting other sensitive activity while connected to this VPN. My point is that VPNs do not make us bullet-proof. This is why we choose providers with proper privacy policies, but we never expect to be completely untraceable.

Because of this, I do not have a strong objection to purchasing your VPN service with a standard credit card. Since Proton VPN has a respected no IP logging policy, they can never translate a public-facing VPN IP address back to your home address. They can also never translate your true home IP address to a VPN address once used. In other words, Proton VPN could never provide the internet activity associated with a name, or the user of specific access to a website.

Some will scoff at these remarks. Some will say that you should only pay for a VPN with cash in the mail (some services do offer this). However, I believe it is overkill. The digital trail will still be present. If you are a fugitive looking for a way to check your email without getting caught, a VPN is not for you. If you are looking to prevent abusive technologies from monitoring your online activity, a VPN can be quite helpful.

Once you have purchased a VPN service, you need a way to connect to it for daily use. For most readers, and almost every client I have consulted, I recommend possessing the standard application provided by the VPN company on any device where it might be needed. These branded apps should suffice for most needs. Proton VPN can be downloaded from https://protonvpn.com/ for both mobile and desktop devices. Once installed, simply provide your account credentials and you can launch your VPN connection.

While I am at home, I rely on a network-wide firewall with VPN, as explained within my digital guide ***Extreme Privacy: VPNs & Firewalls***. When I am on this home network, I do not need any VPN application running on my devices. I only need to launch them when I am away from home and need VPN protection (or prefer a different IP address than that which is in use at my home).

I strongly advise that you always protect your internet usage through a VPN while conducting any of the tasks described throughout this guide.

It is now time to practice our Terminal commands and work with some data. Please do not skip the following chapter! The included tutorials explain usage of several important commands which will be abbreviated in later chapters about leaks, breaches, logs, and ransomware. At the minimum, please familiarize yourself with the overall techniques.

CHAPTER FOUR
TERMINAL COMMANDS

In the previous print edition of this book, I offered several Terminal commands which helped clean specific data sets. Those commands are still here within this new digital edition, but I want to offer more explanation and tutorials. Mastering Terminal commands will make your journey into data collection much easier. Therefore, this chapter has two goals. First, I want to dive into each command which we will use throughout this section to offer better understanding of the actions instead of the results. Second, I want to offer numerous practice examples which allow you to test what you have learned before moving on.

Everything in this chapter should be conducted within Terminal. For Linux and macOS users, that means opening the Terminal application and inputting text directly into the window. For Windows users, that means opening Command Prompt and replicating each command. Note that Windows file paths require a backslash (\) instead of the traditional forward slash (/). As long as you are working within the target directory, this should not be much of an issue. Please note my previous warnings about data work within Windows.

Let's start with something basic. The cd command allows us to change the working directory within Terminal while the ls command displays the contents of the current directory. We have already relied on this heavily in the first section of this book, but let's revisit to make sure we are all capable of the task. Open Terminal within macOS or Linux and execute the following commands, striking Enter/Return after each.

```
ls
cd /
ls
```

The first command should have displayed the files within your home directory, which may have identified folders such as Desktop, Documents, Downloads, and other default directories. The second command changed the directory to the root of your operating system, and the third command should have revealed a different set of folders. Next, let's replicate this within Command Prompt in Windows.

```
dir
cd \
dir
```

The first command should have displayed the files within your home directory, which may have identified folders such as Desktop, Documents, Downloads, and other default directories. The second command changed the directory to the root of your operating system, and the third command should have revealed a different set of folders. Notice the backslash for Windows instead of the forward slash.

Since we will mostly be working within the home directory, Linux and macOS users can use the tilde (~) in the upper left area of your keyboard to navigate to the default home directory, regardless of the computer username. An example follows.

```
cd ~/
```

If you wanted to navigate directly to the Downloads folder within your home folder, you would conduct the following.

```
cd ~/Downloads
```

Navigating to your Documents or Desktop would be as follows.

```
cd ~/Documents
cd ~/Desktop
```

Windows users have another hurdle here. It does not accept a tilde, and insists on "%username" to change into the default user's home directory. The same commands as before would appear as follows.

```
cd C:\users\%username%\
cd C:\users\%username%\Downloads\
cd C:\users\%username%\Documents\
cd C:\users\%username%\Desktop\
```

We can now use the mkdir (make directory) command to create a new folder titled "Datatest" within our Documents folder with the following macOS/Linux command.

```
mkdir ~/Documents/Datatest
```

We can switch to this folder with the following.

```
cd ~/Documents/Datatest
```

We can use the rm (remove) command to delete this folder. Execute the following.

```
rm ~/Documents/Datatest
```

You should have received an error about Datatest being a directory. This is because rm by itself can only delete a file. The following command will delete the folder and all contents. Be careful with this.

```
rm -r ~/Documents/Datatest
```

If you wanted to move a folder instead of delete it, the following will make a directory called Datatest within your Documents folder and move it to your Downloads folder.

```
mkdir ~/Documents/Datatest
mv ~/Documents/Datatest ~/Downloads/Datatest
```

Windows users have more hurdles. The same commands as before would appear as follows. Note that the final two lines are one single command.

```
mkdir C:\users\%username%\Documents\Datatest
cd C:\users\%username%\Documents\Datatest
rm -r C:\users\%username%\Documents\Datatest
mkdir C:\users\%username%\Documents\Datatest
mv C:\users\%username%\Documents\Datatest
C:\users\%username%\Downloads\Datatest
```

I will assume that you can now navigate through directories within Windows systems. While all of the following commands will function within any operating system, any path changes will only be presented for macOS and Linux machines. Use these lessons to replicate on Windows when needed.

The Cat command (short for concatenate) reads data from a file and provides its data as output. It can be used to view the contents of an existing file or combine several files into one. We will mostly focus on the latter, but let's try both. Within Terminal (or Command Prompt), navigate to your Downloads folder to conduct the following.

```
cd ~/Downloads
curl -O https://inteltechniques.com/data/osintdata/cat-1.txt
curl -O https://inteltechniques.com/data/osintdata/cat-2.txt
cat cat-1.txt
```

The first command switched our working directory to the Downloads folder. The next two commands downloaded two files which I have on my website. The final command displays the content of the first file, which should appear as follows.

```
1
1
1
```

Your first simple assignment (Assignment #1) is to display the content within the second file downloaded. All commands for these practice exercises are within the final page of this chapter. Your results should appear as follows.

```
2
2
2
```

We now want to combine these two files into one file called cat-final.txt with the following command.

```
cat cat-1.txt cat-2.txt > cat-final.txt
```

Your result should appear as follows.

```
1
1
1
2
2
2
```

You could add as many files as desired before the ">" which would be combined into the designated file. We will use ">" often, which is basically telling Terminal to output the data into a specific file. Let's remove the three files with the following commands.

```
rm cat-1.txt cat-2.txt cat-final.txt
```

It is now time to test your abilities. I have four files at the following locations:

https://inteltechniques.com/data/osintdata/cat-1.txt
https://inteltechniques.com/data/osintdata/cat-2.txt
https://inteltechniques.com/data/osintdata/cat-3.txt
https://inteltechniques.com/data/osintdata/cat-4.txt

Attempt to download these four files within your Downloads directory, combine them all into a new file called cat-test.txt, delete the four files downloaded, and display the content within the newly-created file. Again, the commands for this are at the end of the chapter, this is specifically Assignment #2.

Did you see 1's, 2's, 3's, and 4's within the file? If so, you passed. The Cat command may seem simple and unnecessary, but I promise it will pay off. When we start downloading hundreds of small log files which need combined into one, this utility will be vital.

The next command will probably be the most used in your arsenal. Sed, short for stream editor, is used to perform basic text transformations on an input file. It allows us to replace specific text throughout an entire file with different text or nothing at all. It is similar to "find and replace" within a text editor, but it is much faster without file size restrictions. When we need to modify a 100 GB file, we will use Sed since that file could not be opened through a traditional text editor. Some examples should help explain.

Assume you have a large file titled customers.csv with thousands of lines similar to the following.

username:test1,email:test@gmail.com,password:pass123,ip_address:10.2.2.3,phone:5551212,id:32
username:test2,email:test2@gmail.com,password:pass456,ip_address:10.2.2.4,phone:5551213,id:33

This might actually be acceptable for most data collection enthusiasts, but I am bothered by the wasted data. The presence of "username", "email", "password", etc. on every line is unnecessary. If this file only contained 200 entries, it would be no big deal. If the file contained 500,000,000 lines and had a size of 200 GB, we would really be wasting drive space. Our queries would also take much longer due to the searching of unneeded data. Let's fix this with the following command.

```
sed "s/username//gI" customers.csv > customers-cleaned.csv
```

We should now dissect this command.

sed	The command executable.
"s/	The command to substitute.
username/	The text we want to find.
/	The text we want to replace (nothing in this example).
g	The command to replace every occurrence in the document.
I"	The command to ignore case (UPPERCASE = lowercase).
customers.csv	The input file.
>	The command to output to another file.
customers-cleaned.csv	The output file name.

The result of this command would be the following.

:test1,email:test@gmail.com,password:pass123,ip_address:10.2.2.3,phone:5551212,id:32
:test2,email:test2@gmail.com,password:pass456,ip_address:10.2.2.4,phone:5551213,id:32

This removed "username" but left the colon (:) in the line. We could have modified our query better with the following.

```
sed "s/username\://gI" customers.csv > customers-cleaned.csv
```

The results would now appear as follows:

test1,email:test@gmail.com,password:pass123,ip_address:10.2.2.3,phone:5551212,id:32
test2,email:test2@gmail.com,password:pass456,ip_address:10.2.2.4,phone:5551213,id:33

Notice the presence of the backslash (\) before the colon. This is because a colon is not a standard letter or number. We must always "escape" any special characters with a backslash. I rarely conduct a Sed operation with both an input and output file. I prefer to edit the files "in-place" with the "-i switch. Note that the "I" option at the end only works for the GNU version of Sed, which we previously installed. The following command would replicate our modification to the customers.csv file without the need to create a new file, and would ignore all case (UPPERCASE or lowercase).

```
sed -i "s/username\://gI" customers.csv
```

Linux and macOS users should have no problem with this command. Windows users have yet another hurdle. The "-i" is not allowed within Windows, as that version of Sed does not include the in-place option. Windows users will need to modify their commands to match the previous example which creates a new file with the omitted details. Did I mention that I prefer macOS and Linux machines? The following command would remove the information and create a new file within Windows.

```
sed "s/username\://gI" customers.csv > customers-cleaned.csv
```

It is now time to practice. Your assignment (#3) is to download the file from the following URL and remove "username:", "email:", "password:", "ip_address:", "phone:", and "id:" from it using Sed. Don't forget to escape the underscore (_). This will require eight commands (found on the last page of this chapter) including the instructions to switch to your Downloads directory and download the file. For extra credit, Cat the file to view the contents.

https://inteltechniques.com/data/osintdata/sed-customers.csv

Your result should appear as follows.

test1,test@gmail.com,pass123,10.2.2.3,5551212,32
test2,test2@gmail.com,pass456,10.2.2.4,5551213,33

If you had a 200 GB original file, your new file would be almost half of the size. This is a substantial savings of disk space, and will be vital once you start collecting large data sets. We are barely scratching the surface with the power of Sed. Consider the following examples, assuming you want to modify a file titled 1.txt. If this all seems overwhelming, please do not worry. We will conduct many more examples using real data throughout the rest of this section.

Replace "OLD" with "NEW":	sed -i "s/OLD/NEW/gI" 1.txt
Replace all commas with hyphens:	sed -i "s/\,/\-/gI" 1.txt
Remove all data until the first comma:	sed -i "s/^\([^,]*,\)//gI" 1.txt
Remove all data until the first colon:	sed -i "s/^[^:]*://gI" 1.txt
Remove all single quotes:	sed -i " s/\'//gI" 1.txt
Remove all double quotes:	sed -i "s/\"//gI" 1.txt
Remove all data between "A" & "B:	sed -i "s/\(A\).*\(B\)/\1\2/" 1.txt
Remove all data between "{" and "}":	sed -i "s/{[^}]*}//gI" 1.txt
Remove all digits between commas:	sed -i "s/\,[0-9]*\,//gI" 1.txt
Remove any line beginning with "A":	sed -i "/^A/d" 1.txt
Remove empty lines:	sed -i "' '/^$/d" 1.txt
Remove first 10 lines:	sed -i "1,10d" 1.txt
Remove first ten characters:	sed -i "s/^.\{10\}//" 1.txt
Remove everything after the last "_":	sed -i "s/_[^_]*$//" 1.txt
Remove hyphens from phone numbers:	sed -i "s/\([0-9]\{3,\}\)-/\1/g" 1.txt

I realize we are getting very technical early in this guide. If you feel overwhelmed, don't worry. I promise this will make more sense when we deal with real data.

Before we move on, we should discuss the likelihood of errors on macOS and Linux machines due to unreadable characters. If your downloaded data includes non-ASCII characters, such as ¡, µ, ¶, and many others, the Sed command may not function. This is especially true for the Cut and Sort commands. Therefore, we should start preceding every command with "LC_ALL=C" within macOS and Linux. This command sets the system's language to ASCII. This should avoid character errors and, in some cases, make our queries faster. Let's apply it to our next tutorial about the Cut command.

Cut is extremely powerful. It allows us to easily remove entire columns of a comma separated value (CSV) file without the need to identify specific text with Sed. Assume you have downloaded a file titled cut-1.txt with text similar to the following.

internal_id,customer_id,email,password,value
1,kjh234g5,test1@gmail.com,pass123,ydt-655-hhjk4-yui7-6ddgghejkr776djhdi8f
2,asd9f87,test2@gmail.com,pass456,ydt-655-hhjk4-yui8-6ddgghejkr776djhdi8f
3,;l2k3j45,test3@gmail.com,pass789,ydt-655-hhjk4-yui9-6ddgghejkr776djhdi8f

You might have no need to store the "internal_id", "customer_id", and "value". We often see large data sets with too much content which will never be beneficial to us. This is where Cut comes in. It allows us to select specific columns to extract while ignoring the others. My command to extract only the desired columns would be the following.

```
LC_ALL=C cut -d, -f3,4 cut-1.txt > cut-2.txt
```

We should understand each part of the command, as follows.

LC_ALL=C	Sets character type to avoid errors (macOS and Linux).
cut	The "cut" command.
-d,	Identifies the delimiter, which is a comma here (,).
-f3,4	Identifies the desired columns to keep, by number.
cut-1.txt	Identifies the input file.
>	Directs the result to output somewhere.
cut-2.txt	Determines the new file name.

The results of this command are as follows.

email,password
test1@gmail.com,pass123
test2@gmail.com,pass456
test3@gmail.com,pass789

Assignment #4 will test your understanding of the Cut command. Your assignment is to extract only the "email", "password", and "phone" columns using the Cut command from the file located at the following URL. Your result file should be titled cut-cleaned.txt.

https://inteltechniques.com/data/osintdata/cut-1.txt

The original data appears as follows.

internal_id,email,customer_id,password,phone,value
1,test1@gmail.com,kjh234g5,pass123,202-555-1212,ydt
2,test2@gmail.com,asd9f87,pass456,202-555-1213,ydt
3,test3@gmail.com,l2k3j45,pass789,202-555-1214,ydt

Your final result should appear as follows.

email,password,phone
test1@gmail.com,pass123,202-555-1212
test2@gmail.com,pass456,202-555-1213
test3@gmail.com,pass789,202-555-1214

Later in this section, we will use the Cut command to extract desired columns from numerous files in one command. Next is an equally vital command called Sort. This allows us to not only sort the contents of a file alphabetically, but it also removes duplicate lines when we provide specific parameters. This is important because a lot of the data we will acquire is full of redundant entries eating up your valuable disk space. Assume you have a file called sort.txt which contains the following content.

test1@gmail.com:pass123
test2@gmail.com:pass456
test1@gmail.com:pass123
test2@gmail.com:pass456
test1@gmail.com:pass123

The following command would remove the duplicates; sort the remaining content alphabetically; and create a new file called sorted.txt.

```
sort -u sort.txt > sorted.txt
```

This new file would have only the following content.

test1@gmail.com:pass123
test2@gmail.com:pass456

Windows users should have noticed an error with this command. That is because Windows wants "/unique" instead of "-u" to complete the command. Windows hurdles seem to be the common theme here, and Windows users would execute the following command.

```
sort /unique sort.txt > sorted.txt
```

This was a very basic example, but you will encounter extremely large data which will need more tweaking. The following is my preferred command when I need to sort large data. This command will only work on macOS and Linux systems.

```
LC_ALL=C sort -u -b -i -f -S 80% --parallel=8 sort.txt
> sorted.txt
```

The following is a breakdown.

LC_ALL=C	Sets the character type to avoid errors.
sort	The "sort" command.
-u	Instructs to only save unique entries (no duplicates).
-b	Ignores leading blanks.
-i	Ignores non-printable characters.
-f	Ignores case when UPPERCASE and lowercase are the same.
-S 80%	Sets buffer size to use 80% of the system's RAM.
--parallel=8	Uses 8 processor threads to run processes concurrently.
sort.txt	Identifies the input file.
>	Directs the result to output somewhere.
sorted.txt	Determines the new file name.

I chose 8 threads because my machine can support it. You may need to lower or raise this number based on your own hardware. I believe 80% of RAM buffer will work fine on most machines. It is now time for you to test your skills. Assignment #5 challenges you to sort a file titled sort-1.txt located at the following URL.

https://inteltechniques.com/data/osintdata/sort-1.txt

Download this file to your Downloads folder; sort for unique entries and save the file as sort-2.txt; remove the original file; and display the content of the new file within Terminal. The results should appear as follows.

test1@gmail.com:pass123
test2@gmail.com:pass456
test3@aol.com:pass789
test3@gmail.com:pass789

Sometimes, you will encounter data which is simply too large to process. I once retrieved a 260 GB text file full of names, DOBs, and SSNs which possessed several duplicates. Sorting a 260 GB file on a machine with 8 GB of RAM will likely crash the computer and required a reboot. I never like to sort files which are more than twice as large as my amount of RAM. Since I possess 32 GB of RAM in my machine, I try to keep my sorting limited to files which are under 64 GB each. However, I was able to successfully sort a 100 GB Stealer Logs file with 32 GB of RAM, as explained in a later chapter, but the process was quite taxing on my drive. Sometimes, I need to split a large file into several pieces, and then sort those pieces back into one file which excludes all duplicates. To do this, we must first understand the Split command.

Split allows us to turn one large file into several smaller files. There are many options with this command, but we will focus only on those most relevant to our use. Assume I have a 260 GB file titled SSN.txt in my Downloads directory which contains thousands of duplicate entries. I will first split this one file into multiple files, each containing 25 GB of data (25,000,000,000 bytes), with the following commands.

```
cd ~/Downloads
split -C 25000000000 --additional-suffix=.txt SSN.txt
```

The result should be multiple files titled xaa.txt, xab.txt, xac.txt, etc., which each have a 25 GB file size. We can now sort each of these independently with the following macOS/Linux command.

```
for f in x*.txt; do LC_ALL=C sort -u -b -i -f -S 80% -
-parallel=8  $f > sorted_$f; done
```

This command finds all files which begin with "x" and end with ".txt" inside the working directory; executes our sort command on each file; saves each result with "sorted" added to the beginning of the file; and quits when finished. The result is several files titled sorted_xaa.txt, sorted_xab.txt, sorted_xac.txt, etc. Each file has been sorted to only contain unique entries, with all duplicates removed. We can now use our sort command with the merge flag (-m) to quickly merge all sorted files into one large file which only contains unique entries (macOS/Linux).

```
LC_ALL=C sort -S 80% --parallel=8 -m -u sorted* > SSN-
Sorted.txt
```

LC_ALL=C	Sets the character type to avoid errors.
sort	The "sort" command.
-S 80%	Sets buffer size to use 80% of the system's RAM.
--parallel=8	Uses 8 processor threads to run processes concurrently.
-m	Merges pre-sorted files into one
-u	Instructs to only save unique entries (no duplicates).
sorted*	Identifies all input files which start with "sorted".
>	Directs the result to output somewhere.
SSN-Sorted.txt	Determines the new file name.

Notice I omitted the flags for ignoring some characters since we have already sorted those out in the previous command. With my data example, the file of SSN entries went from 260 GB to 210 GB by removing duplicates. We will revisit all of this when we start sorting large files of stealer log data. I understand the frustration of Windows users. While we can make most of these commands work, there will always be hurdles. The remaining chapters of this book will allow Windows machines to acquire all of the data presented. However, minimizing and cleaning that data will be troublesome. I encourage Windows users to focus more on data acquisition than processing the content. Queries will work fine in future chapters, but the time required to search the data will be more. All of this will always be easier for macOS and Linux users.

It is now time to test your skills. Assignment #6 brings together a lot of what you have learned in this chapter, and we will be using real breach data. This will require a lot of work, but it will be great exercise for future demonstrations. I have a 19 MB file at the following URL which contains true Whois data. We will get much more of this data later, but this is a good way to test our skills. Inside this file is many names and addresses of domain owners who did not protect the information. However, it also contains many duplicates and columns of data which are unneeded.

https://inteltechniques.com/data/osintdata/Whois.txt

Your challenge is to conduct the following.

- Download the file into your Downloads directory.
- Remove all double quotes (") from the file.
- Extract the "domain_name", "registrant_name", and "registrant_address" columns into a file titled "who.txt".
- Split the who.txt file into 1 MB chunks.
- Delete the Whois.txt and who.txt files.
- Sort each split file for unique entries, each saved as a file beginning with "sorted".
- Merge the "sorted" files into one file called Whois-Sorted.txt.
- Remove the sorted* and xa* files.

Your Whois-Sorted.txt file should be 1.1 MB and include entries similar to the following.

00000777.com,Peng Goh,G.P.O. Box 732
0012222.com,Peng Goh,G.P.O. Box 732
0013333.com,Peng Goh,G.P.O. Box 732

This file is not very useful, but it served as a good demonstration. We will identify much better data sets later. The splitting of such a small file and then merging the contents was unnecessary, but I wanted you to practice these steps for preparedness when large data is acquired.

We will use the Awk command several times throughout this section. There are unlimited uses for this utility, but let's take a look at one way it may be beneficial. Assume you have a large file with entries similar to the following.

test1@gmail.com:pass123
test2@gmail.com:pass456
null
test3@gmail.com:pass789
null
test4@gmail.com:pass321

Many times, we will encounter data which has empty, or "null" lines which eat up space for no reason. Using Awk, we can remove any lines which do not contain a specific character. The following would remove any lines which do not contain a valid email address, with "@" inside the line, within a file titled awk-1.txt, which is saved to awk-2.txt.

```
awk '/@/' awk-1.txt > awk-2.txt
```

The result follows.

test1@gmail.com
test2@gmail.com
test3@gmail.com
test4@gmail.com

Another utility I often use is Word Count (wc). The following commands display the number of lines within our files.

```
wc -l awk-1.txt
wc -l awk-2.txt
```

The results from these two files were "6" and "4" respectively. I often use this to determine how many entries are within a data breach. As I am writing this, I found a data leak from a person search website. The file was over 100 GB, and the wc command revealed it possessed 21,304,654 entries.

You should now practice querying data. We previously installed Ripgrep, which allows us to quickly search within any data type for results matching our query. A typical query, which would search through all data within a file titled passwords.txt for the word "michael", would appear as follows. Excluding the file name would search all files within the working directory.

rg -a -F -i -N michael passwords.txt

The following is the breakdown.

rg	The command to launch Ripgrep.
-a	The switch to search all data as text.
-F	The switch to treat the pattern as a literal string.
-i	The switch to ignore case.
-N	The switch to exclude the line number.
michael	The search term.
passwords.txt	The file to search.

In this demonstration, I added a switch (-) for each option, which is overkill. We can add all of the options within one command and only one switch. In other words, our original Ripgrep query of rg -a -F -i -N michael passwords.txt can be replicated much more simply as rg -aFiN michael passwords.txt. Ripgrep is a very powerful query tool with many options. Let's work through the most common query parameters which we will use over the next several chapters.

rg -aFiN T@Gmail.com	Search EXACTLY t@gmail.com or T@GMAIL.com
rg -aiN Test@Gmail.com	Search ALL test and gmail and com
rg -aFN T@Gmail.com	Search ONLY T@Gmail.com and not t@gmail.com
rg -aFi Test@Gmail.com	Search EXACTLY test@gmail.com and show line #
rg --help	Show Ripgrep help menu

Each of these commands will search within all of the files contained in the current folder. If you are in Terminal within your Downloads folder and conduct these queries, they will search every file present within that folder, which may be undesired. As you begin to collect multiple large files, you will want to specify the file desired. The following would only search through a specific voter file.

```
rg -aFiN Test@Gmail.com Voter-FL-2018.txt
```

If you want to search two specific pieces of data within a single file, the following would apply. It would display only lines within our file which contain BOTH "Michael" and Bazzell".

```
rg -aFiN "Michael" | rg -aFiN "Bazzell" Voter-FL-2018.txt
```

If you want to search two potential pieces of data within a single file, the following would apply. It would display only lines within our file which contain EITHER "Bazel" or Bazzell".

```
rg -aFiN "Bazel|Bazzell" Voter-FL-2018.txt
```

Overall, we will almost always use the combination of rg -aFiN "target". You can play with this now against any test files you have downloaded. I think we are finally ready to start acquiring some real data.

I think I have exhausted the patience of many readers who want to dive into data acquisition. There are many more commands which will need explained and tested, but we can tackle those within the remaining chapters. The following page provides the solutions to the assignments presented within this chapter. The following chapter finally begins our journey into data collection.

Assignment #1

```
cat cat-2.txt
```

Assignment #2

```
cd ~/Downloads
curl -O https://inteltechniques.com/data/osintdata/cat-1.txt
curl -O https://inteltechniques.com/data/osintdata/cat-2.txt
curl -O https://inteltechniques.com/data/osintdata/cat-3.txt
curl -O https://inteltechniques.com/data/osintdata/cat-4.txt
cat cat-1.txt cat-2.txt cat-3.txt cat-4.txt > cat-test.txt
rm cat-1.txt cat-2.txt cat-3.txt cat-4.txt
cat cat-test.txt
```

Assignment #3

```
cd ~/Downloads
curl -O https://inteltechniques.com/data/osintdata/sed-
customers.csv
sed -i "s/username\://gI" sed-customers.csv
sed -i "s/email\://gI" sed-customers.csv
sed -i "s/password\://gI" sed-customers.csv
sed -i "s/ip\_address\://gI" sed-customers.csv
sed -i "s/phone\://gI" sed-customers.csv
sed -i "s/id\://gI" sed-customers.csv
cat sed-customers.csv
```

Assignment #4

```
cd ~/Downloads
curl -O https://inteltechniques.com/data/osintdata/cut-1.txt
LC_ALL=C cut -d, -f2,4,5 cut-1.txt > cut-cleaned.txt
cat cut-cleaned.txt
```

Assignment #5

```
cd ~/Downloads
curl -O https://inteltechniques.com/data/osintdata/sort-1.txt
LC_ALL=C sort -u -b -i -f -S 80% --parallel=8 sort-1.txt >
sort-2.txt
rm sort-1.txt
cat sort-2.txt
```

Assignment #6

```
cd ~/Downloads
curl -O https://inteltechniques.com/data/osintdata/Whois.txt
sed -i "s/\"//g" Whois.txt
cut -d, -f2,11,13 Whois.txt > who.txt
split -C 1000000 --additional-suffix=.txt who.txt
rm Whois.txt who.txt
for f in x*.txt; do LC_ALL=C sort -u -b -i -f -S 80% --
parallel=8  $f > sorted_$f; done
LC_ALL=C sort -S 80% --parallel=8 -m -u sorted* > Whois-
Sorted.txt
rm sorted* xa*
```

CHAPTER FIVE
DATA LEAKS

The chapter on data breaches and leaks within the previous edition of this book began with a demonstration involving voter data. I included links to websites which hosted the entire voter databases of nine states. The person who hosted those sites had filed FOIA requests for the public voter data, and published the results when received. In early 2022, all nine sites began forwarding to a genealogy site which did not possess any voter data. However, data never dies.

In early 2023, a new site called **BreachForums** (breachforums.is) surfaced to replace the previously-seized **BreachForums** (breached.to) site which was created to replace the previously-seized **RaidForums** (raidforums.com) site. The latest site is basically a clone and serves as a marketplace for breaches and leaks. **I do not recommend visiting this site until you have a full understanding of the consequences.** I also do not ever recommend purchasing credits which allow download of stolen data. There is no need, as we will access the data in other ways without paying criminals through a shady website.

In June of 2022, BreachForums released an official database of complete voter registration details from twenty-eight states. This included the original leaked data I discussed in the previous edition, plus numerous additional states acquired from other leaks. All of this data is technically public, but it was never meant to be published in this way. The original source of this voter data was official government agencies, and the data was obtained legally. The **publication** in this format is likely against some, if not all, state laws.

If you believe **possession** of this type of invasive data is illegal, consider Ohio's stance. If you navigate to https://www6.ohiosos.gov/ords/f?p=VOTERFTP:HOME:::::: and click on the "Statewide Voter File" tab, you can download the entire database of registered voters directly from the official Ohio government domain. This is truly public data and avoids the use of third-party providers.

The data set for Washington state is also available from the government if you promise not to further distribute or sell the content. You can download the entire database directly at https://www.sos.wa.gov/elections/vrdb/extract-requests.aspx.

Other states' voter databases are also out there, and we will find them together. The data sets include millions of people's names, dates of birth, and home addresses (some otherwise unpublished), along with numerous email addresses and cellular telephone numbers. I hope by now you could see how this might be beneficial to online investigations. Let's begin with some official data from Ohio.

I navigated to https://www6.ohiosos.gov/ords/f?p=VOTERFTP:HOME::::: and clicked the "Statewide Voter Files" tab. I then downloaded all four options which consisted of approximately 425 MB of compressed data. After decompressing the files, I was presented 4.6 GB of text files, each beginning with "SWVF". Now, let's apply the lessons learned within the previous chapter to slim down this data. First, I combined all four files into one file titled Ohio.txt with the following Terminal command while in the Downloads directory.

```
cat SWVF* > Ohio.txt
```

I then removed the unnecessary individual files with the following.

```
rm SWVF*
```

I now want to view the contents of the new text file, but it is quite large. While my computer could open this entire file within a text editor, it is never a good idea. When we start downloading 50 GB files, we will crash our computers if we try to open them. Instead, we will use Ripgrep. The following command displays only the first two lines within Terminal.

```
rg -m2 "" Ohio.txt
```

The result is overwhelming, as each line of this text file is enormous. However, this is a digital guide, so I will include the entire output on the following page. The first 80% of the file is line one while the last 20% is line two. The first line identifies the column headers (descriptions), while the remaining lines are the actual content. This first line can be quite helpful. It allows us to understand what the data is and why we may want it. The second line was modified for publication here.

We can see the valuable data within the second line. Most of the beneficial content is at the beginning. For this demonstration, I only want the names, addresses, and dates of birth of the voters, so I will eliminate all undesired data with the following Cut command.

```
cut -d, -f4,5,6,8,12,13,14,15,16,17 Ohio.txt > Ohio-clean.txt
```

This extracts only columns 4, 5, 6, 8, 12, 13, 14, 15, 16, and 17, saving that data to a new file called Ohio-clean.txt. The output is similar to the following.

"SMITH","DAVID","E","1975-05-20","33
MAIN","","WINCHESTER","OH","45697",""

Compare that to the second entry on the following page.

"SOS_VOTERID","COUNTY_NUMBER","COUNTY_ID","LAST_NAME","FIRST_NAME","MIDDLE_NAME","SUFFIX","DATE_OF_BIRTH","REGISTRATION_DATE","VOTER_STATUS","PARTY_AFFILIATION","RESIDENTIAL_ADDRESS1","RESIDENTIAL_SECONDARY_ADDR","RESIDENTIAL_CITY","RESIDENTIAL_STATE","RESIDENTIAL_ZIP","RESIDENTIAL_ZIP_PLUS4","RESIDENTIAL_COUNTRY","RESIDENTIAL_POSTALCODE","MAILING_ADDRESS1","MAILING_SECONDARY_ADDRESS","MAILING_CITY","MAILING_STATE","MAILING_ZIP","MAILING_ZIP_PLUS4","MAILING_COUNTRY","MAILING_POSTAL_CODE","CAREER_CENTER","CITY","CITY_SCHOOL_DISTRICT","COUNTY_COURT_DISTRICT","CONGRESSIONAL_DISTRICT","COURT_OF_APPEALS","EDU_SERVICE_CENTER_DISTRICT","EXEMPTED_VILL_SCHOOL_DISTRICT","LIBRARY","LOCAL_SCHOOL_DISTRICT","MUNICIPAL_COURT_DISTRICT","PRECINCT_NAME","PRECINCT_CODE","STATE_BOARD_OF_EDUCATION","STATE_REPRESENTATIVE_DISTRICT","STATE_SENATE_DISTRICT","TOWNSHIP","VILLAGE","WARD","PRIMARY-03/07/2000","GENERAL-11/07/2000","SPECIAL-05/08/2001","GENERAL-11/06/2001","PRIMARY-05/07/2002","GENERAL-11/05/2002","SPECIAL-05/06/2003","GENERAL-11/04/2003","PRIMARY-03/02/2004","GENERAL-11/02/2004","SPECIAL-02/08/2005","PRIMARY-05/03/2005","PRIMARY-09/13/2005","GENERAL-11/08/2005","SPECIAL-02/07/2006","PRIMARY-05/02/2006","GENERAL-11/07/2006","PRIMARY-05/08/2007","PRIMARY-09/11/2007","GENERAL-11/06/2007","PRIMARY-11/06/2007","GENERAL-12/11/2007","PRIMARY-03/04/2008","PRIMARY-10/14/2008","GENERAL-11/04/2008","GENERAL-11/18/2008","PRIMARY-05/05/2009","PRIMARY-09/08/2009","PRIMARY-09/15/2009","PRIMARY-09/29/2009","GENERAL-11/03/2009","PRIMARY-05/04/2010","PRIMARY-07/13/2010","PRIMARY-09/07/2010","GENERAL-11/02/2010","PRIMARY-05/03/2011","PRIMARY-09/13/2011","GENERAL-11/08/2011","PRIMARY-03/06/2012","GENERAL-11/06/2012","PRIMARY-05/07/2013","PRIMARY-09/10/2013","PRIMARY-10/01/2013","GENERAL-11/05/2013","PRIMARY-05/06/2014","GENERAL-11/04/2014","PRIMARY-05/05/2015","PRIMARY-09/15/2015","GENERAL-11/03/2015","PRIMARY-03/15/2016","GENERAL-06/07/2016","PRIMARY-09/13/2016","GENERAL-11/08/2016","PRIMARY-05/02/2017","PRIMARY-09/12/2017","GENERAL-11/07/2017","PRIMARY-05/08/2018","GENERAL-08/07/2018","GENERAL-11/06/2018","PRIMARY-05/07/2019","PRIMARY-09/10/2019","GENERAL-11/05/2019","PRIMARY-03/17/2020","GENERAL-11/03/2020","PRIMARY-05/04/2021","PRIMARY-08/03/2021","PRIMARY-09/14/2021","GENERAL-11/02/2021","PRIMARY-05/03/2022","PRIMARY-08/02/2022","GENERAL-11/08/2022","SPECIAL-02/28/2023","PRIMARY-05/02/2023","SPECIAL-08/08/2023","SPECIAL-09/12/2023","PRIMARY-10/03/2023","GENERAL-11/07/2023"

"OH0016289533","01","3086","SMITH","DAVID","E","","1975-05-20","1991-10-08","ACTIVE","R","33 MAIN","","WINCHESTER","OH","45697","","","","","","","","","","","","","","","","02","04","SOUTHERN OHIO ESC","","","ADAMS COUNTY/OH VALLEY LSD (ADAMS)","","LIBERTY TOWNSHIP","01AAI","05","90","14","LIBERTY TOWNSHIP","","","R","X","","","R","X","","X","R","X","","","","X","","R","X","","","X","","","R","","X","","","","","","X","R","","","X","R","","X","R","X","","","","X","R","X","","","","X","R","","","X","","","X","R","","X","","","X","R","X","","","","X","R","R","X","","","X","","","X"

I do not like all of those unnecessary quotes, so I will remove them with the following command.

```
sed -i "s/\"//gI" Ohio-clean.txt
```

The result is as follows.

SMITH,DAVID,E,1975-05-20,33 MAIN,,WINCHESTER,OH,45697,

Not only is that cleaner, but the 4.6 GB file is now only 549 MB. I believe this file is finished, and it can be added to our external SSD. The easiest way to do this is via the default file explorer application for your operating system. I prefer to create a new folder at the root of my external SSD titled "Voter" for this type of data. If you named this external drive "DATA", as previously explained, we can search the file with the following commands within macOS.

```
cd /Volumes/DATA/Voter
rg -aFiN Bazzell Ohio-clean.txt
```

As a reminder, the first command navigated to our external drive. The easiest way to do this is to type "cd", followed by a space, and then drag the desired folder from the Finder into the Terminal box. This also works in Pop!_OS and Ubuntu, however, the Linux command to change into the external drive is different.

```
cd /media/$USER/DATA/Voter
rg -aFiN Bazzell Ohio-clean.txt
```

This "rg" command queries our file for the presence of "Bazzell". The following is the redacted result.

```
BAZZELL,[redacted]ON,1935-0[redacted] SALE[redacted]G PL,,ENGLEWOOD,OH,45322,
BAZZELL,[redacted]N,1964-06[redacted]SHELTO[redacted]TERING,OH,45429,
BAZZELL,[redacted]ER,JAMES,[redacted]9,936[redacted]DR,,KETTERING,OH,45429,
```

Many states either offer their voter registration as a download option within their website or via a request directly to their offices. If I can extract raw data from the official source, that is always preferred, but I never make any requests to the government. The data is almost always floating around other places. Currently, you can download 110 GB of voter registration data from the BreachForums site previously mentioned. I cannot provide a direct link to the content, but I can help you find it. The following Google query should provide an abundance of links to investigate.

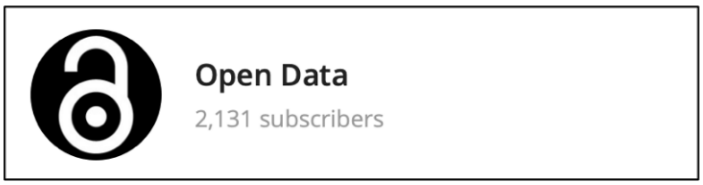
site:breachforums.is voter

One of the first results during my test was titled "USA Voter Databases Collection" which contained full voter registration files for 28 states. However, one must possess eight "credits" in order to download the data. I never recommend purchasing credits,

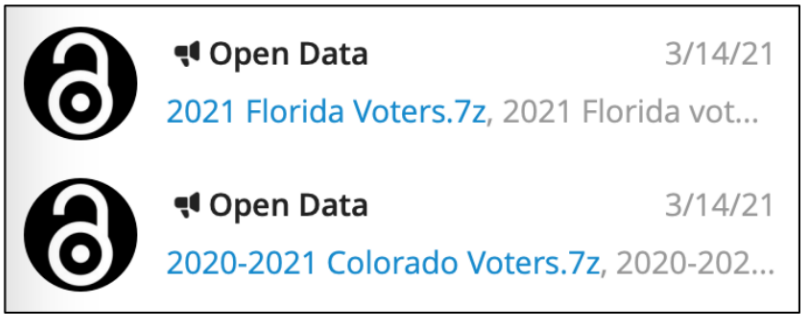
as this gives money to criminals and is often unnecessary. Most of these breach forums allow users to earn credits for free by posting and commenting within various areas of the site. One should be able to generate eight credits by leaving eight comments within an area about recent events. This is quite annoying, but it should get the job done. I typically try to find alternative sources, such as Telegram.

We previously installed Telegram onto our devices. However, you must possess a Telegram account in order to access any data. This requires registration through the Telegram mobile app. This can be completed from a mobile device or an Android emulator, as explained in ***OSINT Techniques, 10th Edition***. If Telegram requires a phone number for your account, they should accept Google Voice and other VoIP numbers. Creating the account can be a pain, but the effort is justified.

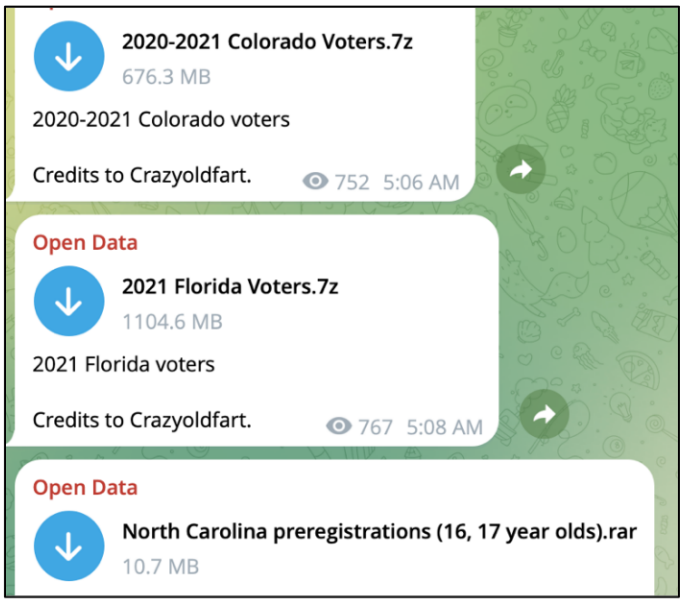
Once you have Telegram Desktop open and logged in to an active account, we can begin searching. First, query "Open Data" within the search field and look for a room similar to the following.



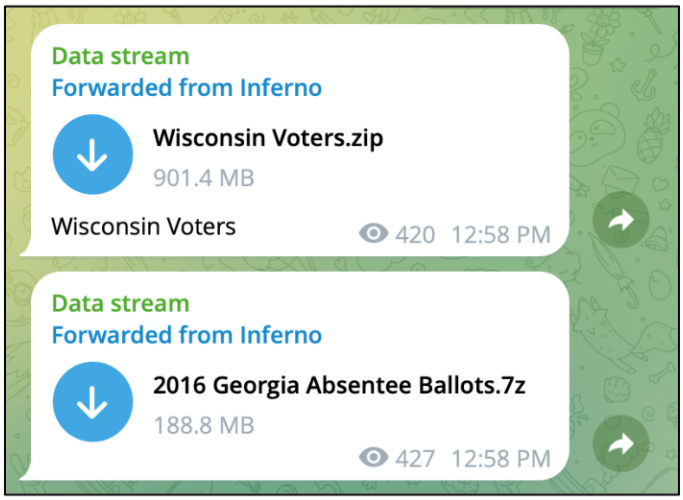
Once you see the room, click the "Join Channel" option to save it to your favorites. Next, search for "voter" within the search field and scroll down to the "Messages" section of the results. This immediately displayed the following.



Clicking these links opened that portion of the channel. This presented three interesting compressed voter registration databases. Mine appeared as follows.



Clicking the down arrows began the download process. The first file contained Colorado voter data. After decompressing the 676 MB 7z file, I possessed 3.1 GB of data. If desired, you could replicate the previous demo to combine and clean all of the data. Repeating the process and searching for "Data Stream" then "Voter" presented the following. Within a few minutes, I had downloaded six state's voter rolls.



You will likely find that many of the voter data sets on Telegram are outdated. Some are from 2015 or earlier. Since voter registration data is always changing, you may want to research newer sources. In 2020, RaidForums offered a huge download consisting of current voter registration from 19 states. That website is now gone, but the data lives on. Shortly after the domain was seized, someone created a torrent file which allows download of every official data set from the defunct website.

I am not allowed to provide a direct link to this data, but I can tell you how I found it. Searching "Raidforums databases" produced over 100,000 results, most of which

provided no valuable content. However, the following search within Google produced ten interesting results.

raidforums "magnet:?" database 2020

The "magnet:?" term forces Google to only display results which include that specific text with those characters. This will identify sites which contain an actual torrent link within them, instead of sites which simply discuss the data. The first two links both offered a torrent magnet link, similar to the following fictitious example.

magnet:?xt=urn:btih:651570d629b83e95353c47f9e1

I copied and pasted this data into my browser, which then prompted me to choose an application to open the data stream. I chose Transmission (some systems will choose this by default). This launched Transmission and prompted me to choose the download location. If you have modified your default Transmission settings, your experience might be different. I chose my Downloads folder and clicked "Open". The torrent file was added and launched. I double-clicked the torrent and clicked the "Files" tab. It took a few minutes for the metadata to be retrieved, but I eventually saw a list of data within this torrent.

I immediately deselected all of the files. In some systems (macOS), you may see a button labeled "None" which will do this for you. Within others (Linux), you will need to deselect the highest "Database Leaks" checkbox. This prevents immediate downloading of all 486 GB of this data. After I deselected all boxes, I scrolled down to the "Voter Databases" folder and selected it. My system began downloading the data from all states, including millions of people's names, home addresses, and dates of birth.

I hope you can now see the value of voter data, but we are just getting started. It is important to note that each state stores their data in a unique format. The Ohio example was just to begin the conversation about merging and cleaning of data. There is much more to discuss. Many of the voter files are already stored within comma-separated value files (CSV). Ripgrep will easily search through those for you and you may not need to take any further action. Some are text files with a slightly different format. Please remember that this tutorial is not intended to provide a single solution to all data you might encounter. We are only dipping our toes into the world of leaked data.

I have collected CSV files of leaked data which possessed over 200 columns of data. Many times, the majority of this content is not valuable and taking up precious space. The smaller your files are, the faster you can search them with Ripgrep. I could replicate this for the other state sets if desired. I could also clean up all of the other states and make one huge file called Voter.txt if I really want to minimize things. As you see, there are always many ways to do this.

Consumer Data

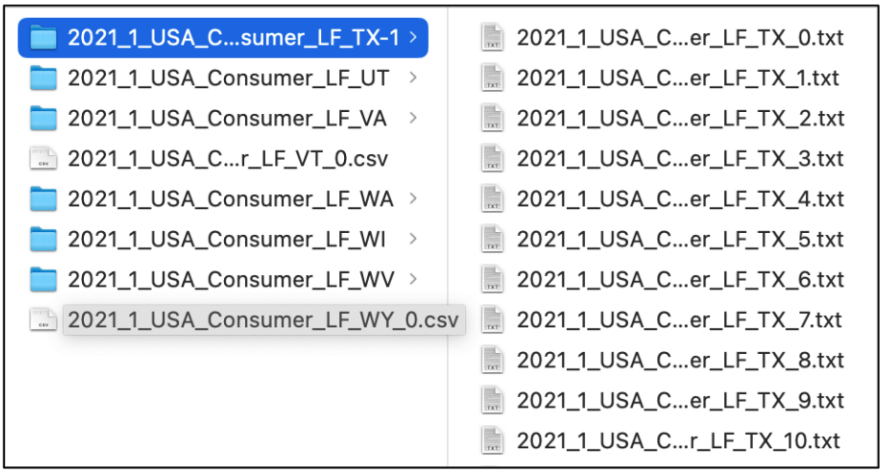
You may have noticed the wealth of information within the previously explained torrent file. Most of the databases within this download contain user credentials and fit better in the next chapter. However, let's focus on some more data within this torrent before we move on. I began the download for the USA Consumer data. We never truly know the source of any data leak, but these were supposedly extracted from an unknown consumer reporting agency through a poorly configured server.

The USA Consumer file contains almost 12 million entries, each of which displays the email address, full name, home address, and telephone number of all victims. The website which they used to enter information was also collected. A typical entry appears as follows.

wjan@yahoo.com,Jane,Burns,26 Main St,CA,Fullerton,94011,,,(650) 722-2000,http://www.coolsavings.com,Coupon Offers

Next, I opened Telegram and searched for a popular channel called "LEAKS AGGREGATOR" and joined. I then searched for "consumer" and perused the options. I found a series of files titled "2021_1_USA_Consumer", each of which appeared to be for a specific state. The Texas collection was so big it needed split into two compressed files, totaling 2.5 GB. The decompressed files were almost 20 GB, and were split into 21 CSV files. Each contained 414 columns of text, including full names, home addresses, email addresses, telephone numbers, and interests.

I went ahead and downloaded all nine files, which contained consumer data for eight states. This presented 5.56 GB of compressed data which decompressed to 59 GB of CSV files. I then conducted the following to clean the data, based on the previous tutorials. First, I needed to Cat all the files into one large file. Since the files are spread over multiple subdirectories, I needed to modify my commands. Below is a partial screen capture of the files located within the Telegram Desktop folder located in my Home Downloads folder.



I executed the following command, which combined all of those files, regardless of being in subfolders, into one 59 GB file called 1.txt.

```
cd ~/Downloads/Telegram\ Desktop
cat * */* > 1.txt
```

The "* */*" command basically tells Terminal to look into the current folder and one level deep into any subfolders. I then cleaned the data with the following commands.

```
cut -d, -f3,4,5,8,10,17,18,43,410,411,412 1.txt > 2.txt
sed -i "s/\"//gI" 2.txt
mv 2.txt ~/Downloads/Consumer-Cleaned.txt
rm -r *
```

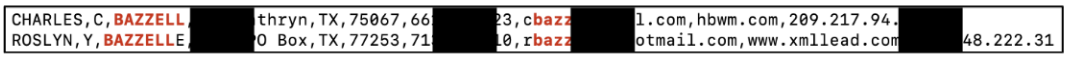
Based on the previous tutorials, can you identify each command's usage? The first command cut the columns in which I was most interested from the original file into a file called 2.txt. The second command removed all of the quotes. The third command moved my cleaned file into my Downloads directory with a more descriptive name. The final command deleted all of the files which I no longer need in the original folder, making it ready for the next attack. The final file was 1.6 GB, containing 31,631,871 entries, which each appeared as follows.

personfirstname,personmiddleinitial,personlastname,housenumber,streetname,state,
ZipCode,FullPhoneNumber,email,datasource,ip
SMITH,J,MIKE,64,Main,VT,05488,8028682000,me@hotmail.com,diaperbeagle.com
,24.127.95.2

This type of consumer data is all over the internet. Most of it contains enormous amounts of unhelpful data, but you may want to archive the original versions of everything you find. In this case, we went from 59 GB to just over 1 GB, and maintained the most commonly searched data. Let's go ahead and query for anyone with my last name with the following commands.

```
cd ~/Downloads
rg -aFiN bazzell Consumer-Cleaned.txt
```

The result should include the following redacted sample.



Marketing Lists

Let's move on to something in a grey area. There are many companies that collect millions of email addresses and create spam lists. These then get sold to companies selling fake pharmaceuticals or designer handbags, and those unwanted messages end up in your inbox. We are all on a spam list somewhere. These are very expensive to buy, especially the good ones. One such database is the Kyle Data, or "Special K", spam list. This database contains 178,288,657 email addresses. There are no passwords, but it does contain other sensitive data. In November of 2015, a web page at updates4news.com temporarily possessed active links to this entire data set. This page is long gone by now. Fortunately for us, the Wayback Machine archived all of it. The following direct link displays hundreds of CSV spreadsheets, each containing millions of email addresses.

https://web.archive.org/web/20151110195654/http://www.updates4news.com:80/kyledata/

If desired, you could use the Firefox extension called DownThemAll to automatically download all of this data overnight into a folder on your desktop titled "SpecialK". You could also grab it from the previously explained torrent file. You could then execute a command through Terminal within that folder of `cat * > SpecialK.txt`. The result would be a single file, 19.73 gigabytes in size. That is very big, but also very powerful. Let's take a look at the content. Assume I was looking for a target who possessed an email address of robynsnest2006@yahoo.com. Assuming that I had made a single file titled SpecialK.txt within my Downloads folder on my machine, my commands in Terminal would be as follows. The result of the search is directly after the search command.

```
cd ~/Downloads
rg -aFiN robynsnest2006@yahoo.com SpecialK.txt
```

MANHATTAN BEACH, robynsnest2006@yahoo.com,yahoo.com, Robyn Bazzell, 2015-04-07, 72.129.87.179, paydayloaneveryday.com, CA, 90266

This tells me that my target lives in Manhattan Beach, CA 90266. Her name is Robyn Bazzell, and on 04/07/2015 her marketing data was collected from the website paydayloaneveryday.com. At the time, her computer IP address was 72.129.87.179. We basically just converted an email address into a full dossier on our target. This type of data is simply not available on public websites. If you doubt the validity of this data, please consider searching your own content. This is only one of dozens of interesting databases stored within the Wayback Machine. It is up to you to exercise your skills from this guide to locate them all.

Social Security Death Index (SSDI)

Hopefully, you agree that possessing your own collection of this type of data could be useful. Let's add another collection of publicly available data. An individual previously purchased the entire Social Security Death Index, which is public data, and uploaded it for public consumption. This entire database can be downloaded from the public Archive.org at **https://archive.org/download/SocialSecurityDeathIndex/2013-05-31**/. After downloading the multiple files, you can combine them as stated previously into a single file and name it SSDI.txt. Assuming you placed the text file in the Downloads folder, the following commands would navigate to the folder and search for anyone in the file with my last name.

```
cd ~/Downloads
rg -aFiN bazzell SSDI.txt
```

The results will include multiple entries with the following format.

433220353 BAZZELL DOROTHY S 0118201303191921

The first set of numbers is the Social Security Number of the deceased individual, followed by the last name, first name, and middle initial. The last set of numbers represents the date of death (01/18/2013) and the date of birth (03/19/1921). I am bthered by these unnecessary tabbed spaces. The following command will replace each tab with a comma, which compresses the data and makes it easier to read. This is much cleaner and will make future search results easier to digest. Note that the spaces in the command are created by pressing the control, v, and tab keys at the same time. This represents a "tab" to the command.

```
sed -i "s/    /\,/gI" SSDI.txt
```

The result would appear as follows.

433220353,BAZZELL,DOROTHY,S,0118201303191921

Let's recap where we are at right now. You might have single, very large, text files that each include all of the registered voter data for various states. These often include dates of birth, names, telephone numbers, addresses, and email accounts. You also have the entire SSDI. You can now search through all of that data with a simple search. If you wanted to identify any entries within all of the files inside a specific folder, with a last name of Bazzell, the following would be the most appropriate search. You could also now search by social security number, date of birth, or date of death.

```
rg -aFiN Bazzell
```

SnapChat Numbers

Let's conduct another Archive.org demonstration. This one is quite dated but allows us to add some historical phone data to our collection. In January 2014, Snapchat had 4.6 million usernames and phone numbers exposed. The breach enabled individual usernames, which are often used across other services, to be resolved to phone numbers. This file is titled SnapChat.7z and can be found online at archive.org/download/SnapChat.7z. This link contains the entire breach inside one text file. Conducting the following search would query a specific Snapchat username for any leaked data. Directly after this command is the result.

```
rg -aFiN mikenap115 SnapChat.txt
```

('21220392XX', mikenap115, '', '')

This identifies the cellular telephone number of our target to include 212-203-92xx. While the last two numbers are redacted, you could try all 100 combinations within the previously discussed telephone search options. It is a great start. You could have also conducted this query on websites which possess this data. However, those sites require an absolute username. It cannot search partial names. If we only knew that our target had a SnapChat name that included the term "topher4", this database would provide the following users of interest.

('30351923XX', 'topher451', '', ''),
('41572499XX', 'topher413', '', ''),
('71974140XX', 'topher456', '', ''),
('75426428XX', 'topher481', '', ''),

We can also now search by telephone numbers. If you know your target has a cellular number of 303-519-2388, you could conduct the following search, with the partial result appearing after.

```
rg -aFiN 30351923XX SnapChat.txt
```

('30351923XX', 'topher451', '', ''),
('30351923XX', 'ben_davis', '', ''),
('30351923XX', 'rosemcdonald', '', ''),
('30351923XX', 'cuzzycook', '', ''),
('30351923XX', 'angelgallozzi', '', ''),
('30351923XX', 'kmo85', '', ''),
('30351923XX', 'rinisob', '', ''),

Your target may not appear in the results, but minimal investigation could result in a powerful new lead. You should note that the full numbers were never publicly leaked, and the group that conducted this attack redacted the results to exclude the last digits.

Public Data Sets (Usenet)

Many large data sets which are beneficial to investigations are not technically "leaks". This chapter has focused mostly on data which was never meant to become publicly available. However, there are numerous archives full of public information which should be considered for your data collection. Most archives cannot be searched through traditional resources such as Google. Instead, we must acquire the data, condense it, and conduct our own queries. As a demonstration, I will explain how I utilize Usenet archives as a vital part of my investigations.

Usenet was my first introduction into newsgroups in the early nineties. My internet service provider allowed full access to thousands of topics through Outlook Express. I could subscribe to those of interest and communicate via email with people from all over the world. This sounds ridiculously common today, but it was fascinating at the time. I located a newsgroup about my favorite band, and I was quickly trading bootleg cassettes and exchanging gossip about the music industry. I was freely sending messages without any consideration of any abilities to permanently archive everything.

Today, the Internet Archive presents huge repositories of data containing over thirty years' worth of Usenet messages. It is presented in over 26,000 archives, each containing between 1 to 20,000 files. It is massive, and would take years to download manually. Fortunately, we can use a download script created by the Internet Archive to automatically obtain all desired data.

The following tutorial will install the necessary software into our macOS or Linux machine; download the entire Usenet archive; extract email addresses and names of each member; acquire newsgroup data to associate with each person; and condense the data into a usable format. This will take some time and may be overwhelming for some readers at this stage. The final product is worth the effort, and I offer a download link at the end if you want to avoid the hassle. If you are up for the challenge, please follow along to learn how to deal with large Archive.org data sets. Let's begin.

First, we need to install the Internet Archive script within our macOS or Linux machines. Unfortunately, this tool is unreliable within Windows. Enter the following Terminal commands for Linux systems.

```
sudo apt update
sudo pip install -U internetarchive
```

Conduct the following within macOS.

```
brew install internetarchive
```

We now have the application installed and ready to use from any folder. Next, we need to identify the Internet Archive collections which we want to acquire. For this demonstration, I will focus on the following two data sets.

https://archive.org/details/giganews
https://archive.org/details/usenethistorical

These are two independent Usenet archives. Each contains unique records and some redundant data. Let's take a look at the second collection. The first folder is titled Usenet-Alt and contains over 15,000 files extracted from decades of conversations within the "Alt" communities. Opening the file titled alt.2600.fake-id.mbox reveals names, email addresses, dates, and entire messages dating back to 1997. The following is an excerpt.

From: "David N." <david.nine@juno.com>
Subject: Need Fake Texas DL
Date: 1998/07/11
Newsgroups: alt.2600.fake-id
I need a quality bogus TX DL for my 17 year old sister. Can you help me out?

Possessing a database of every email address and name from thirty years of Usenet posts can be very beneficial. Downloading every message can be overkill. This entire data set is over a terabyte in size. Instead of trying to download everything, I only want specific portions of the data. The Giganews collection includes two files for each archive. The first is the "mbox" file which includes the full messages along with user information. These are very large and could take months to download. The second is a "csv" file which only includes the date, message ID, name, email address, newsgroup, and subject of each post. This is a much more manageable amount of data which includes the main information desired (name and email address). We will only download the minimal information needed for our purposes.

First, we must create a text file which includes every archive within each collection. The following commands navigate to your Desktop and create text files for the Giganews and Usenet Historical archive.org data sets.

```
cd ~/Desktop
ia search 'collection:giganews' -i > giganews1.txt
ia search 'collection:usenethistorical -i >
usenethistorical1.txt
```

Let's dissect the Internet Archive commands. "ia" is the application, "search" identifies the type of query, "collection:giganews" identifies the target data on archive.org, "-i" instructs the application to create an item list, and " > giganews1.txt" provides the desired output. These text files on your Desktop contain the names of all the archives within the collections. The following is an excerpt.

usenet-alt.2600
usenet-alt.2600a
usenet-alt.2600crackz

We can now instruct Internet Archive to begin downloading the necessary files. The following command downloads only the "csv" files from the Giganews collection. It can take several hours if you have a slow internet connection. If you do not have sufficient space within your VM, consider saving these to an external drive as previously instructed.

```
ia download --itemlist giganews1.txt --glob="*.csv.gz"
```

You should now have thousands of folders, each containing multiple compressed CSV files. This is not very useful or clean, so let's extract and combine all the valuable data. The following command will decompress all the files and leave only the actual CSV documents. It should be executed from whichever directory contains all of the downloaded folders. In my demonstration, it is in the Desktop.

```
gunzip -r .
```

"gunzip" is the command to extract the data from the compressed files (use "unzip" if your macOS machine does not recognize this command), "-r" conducts the command recursively through any sub-folders, and " ." continues the action through every file. Below is a modified excerpt from one of the files. It identifies the date of the post (12/04/2003), name of the author (John Smith), email address associated with the account (John-smith@smh.com), specific newsgroup used (microsoft.windowsxp), and the subject of the post (Help me!).

#date from newsgroups Subject 20031204 John Smith <John-smith@smh.com> microsoft.windowsxp Help Me!

We still have thousands of files, which is not ideal. The following command will combine every CSV file into a single text file titled Giganews2.txt. Furthermore, it will only extract the columns of data most valuable to us, as explained afterward.

```
find . -type f -name \*.csv -print0 | xargs -0 cut -
f1,3,4,5 > Giganews2.txt
```

Let's break down each portion of this command, as it can be quite useful toward other data sets.

find	This is the command to "find" data to manipulate.
.	This instructs the command to find all files.
-type f	This searches for a regular file type.
-name *.csv	This filters to only find a specific file extension (csv).
-print0	This directs output to a file instead of the screen.
|	This is a "pipe" character which separates commands for instruction.
xargs -0	This builds our next command from the previous data as input.
cut -f1,3,4,5	This extracts only the data from columns 1, 3, 4, and 5 from each CSV.
>	This instructs the command to send the data to another file.
Giganews2.txt	This identifies the output file name.

The final result should be a very large file which contains all of the valuable content from within every downloaded file. In a moment, we will use this data to research our targets. The Usenet Historical collection is stored in a different format, so these instructions will need to be modified. The following steps will extract the beneficial data from that collection, and should appear similar to the previous actions. First, we must download the entire archive with the following command.

```
ia download --itemlist usenethistorical1.txt --
glob="*.zip"
```

Next, we must extract the "mbox" files from their compressed containers with the following.

```
find . -name "*.zip" -exec unzip {} \;
```

Finally, we must extract the "From:" line from each file with the following command.

```
rg -aFiN "From: " > UsenetHistorical2.txt
```

This leaves us with a single large file titled UsenetHistorical2.txt. Below are a few lines.

```
alt.2600.mbox:From: "Bill Smith" <momop@mvc.biglobe.ne.jp>
alt.2600.mbox:From: "yosinaga jackson" <purizou@geocities.co.jp>
alt.2600.mbox:From: "yosinaga jackson" <purizou@geocities.co.jp>
```

We do not have as many details within this data set as we did with the Giganews option. However, possessing the names, email addresses, and newsgroups provides a lot of value. Both the Giganews2.txt and UsenetHistorical2.txt files possess many duplicate entries wasting valuable space. You might consider the following command which combines both files into one file while removing all duplicate lines.

```
sort -u -f Giganews2.txt UsenetHistorical2.txt >
UsenetFinal.txt
```

We now possess a single file, quite large in size, which possesses the names, email addresses, and interests of most users of the Usenet system over a thirty-year period. Now, let's discuss ways this can be used during investigations. Assume you are researching a target with my last name. When using Ripgrep, your command is as follows.

```
rg -aFiN bazzell UsenetFinal.txt
```

The result includes the following partial data.

microsoft.public.pocketpc,Steven Bazzell steveill@yahoo.com
sci.bio.microbiology,cbcWiW@UJOMHEH.EDU.TW Wayne A. Bazzell,M.P.S.E
sci.crypt,cbcWiW@UJOMHEH.EDU.TW Wayne A. Bazzell,M.P.S.E
sci.crypt,General Darcy J. Bazzell corporation@incidentally.mi.us

The first line identifies Steven Bazzell in the Usenet group of microsoft.public.pocketpc while using an email address of steveill@yahoo.com. You could search by names, email addresses, partial email addresses, domains, etc. I have successfully used my own Usenet archive in the following scenarios.

- When searching a unique target name, I have uncovered old email addresses which led me to otherwise unknown social network profiles.
- When searching an email address, I have identified various interests of the target, determined by the newgroups to which he or she has posted.
- When searching an email address, it serves as a great way to verify a valid account and establishes a minimum date of creation.
- When searching a domain, I often identify numerous email addresses used by the owner.
- When conducting background checks on targets over the age of 40, I often identify email addresses connected to questionable interests. While investigating a potential police officer, I located evidence he had previously posted images to a child pornography newsgroup. This was confirmed during the interview.

These collections possess over 43 million unique email addresses. You may be questioning the justification of the time it would take to create your own archive. While I encourage you to replicate the steps here as an educational opportunity, I also respect the difficulty involved. Therefore, I have made my own Usenet file available for download at the following URL.

https://mega.nz/file/mSJGGDYI#oNIeyaG2oIHcHQFfGeFuq3zxUp_cCgARVf6bQNqp9ls

The file contains only the name, email address, and newsgroup fields from both collections in order to keep the size minimal. I have removed all duplicates and cleaned the file formatting to exclude any unessential data. It is 12 GB compressed to 3 GB. I hope you find it to be useful.

We have only discussed a few databases stored within archive.org. I promise you there are many others containing sensitive details which would help greatly within your investigations. Conducting searches for any specific breached databases displayed on notification websites such as haveibeenpwned.com/PwnedWebsites should reveal interesting results.

Open Databases

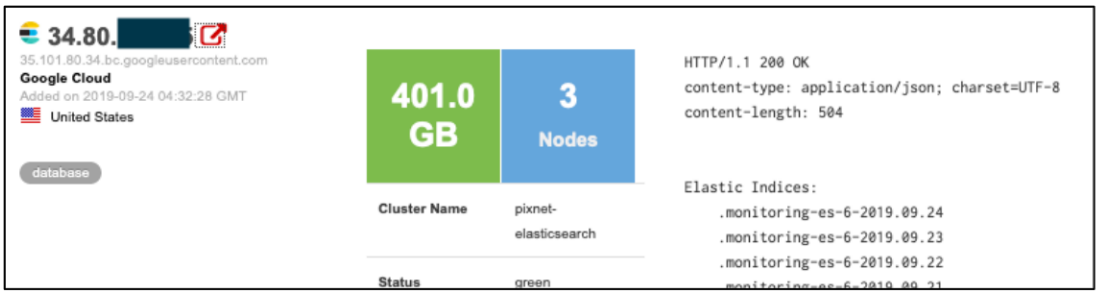
The next time you hear about a "misconfigured server" which exposed millions of customer details, this is also likely a data leak. Our best tool to find this data is Shodan (shodan.io), and you will need to be logged in to a free or paid account to conduct these queries. I will focus only on open Elasticsearch databases for now, which are extremely easy to access. Our Shodan search is as follows.

product:elastic port:9200 [target data]

As a demonstration, I want to search for open Elasticsearch databases which contain an index titled "Customer". Think of an index as the title of a table or section of the database. The following search on Shodan produces over 200 results including an index titled "customer".

product:elastic port:9200 customer

The first result during writing was an open database with 401 GB of data. I have redacted the IP address and modified each address throughout this example. I will use a fictitious address of 34.80.1.1 throughout the entire demonstration. The following figure displays this result. Clicking the red square with arrow next to the redacted IP address connects to the database within your browser in a new tab. The second figure displays this result. This confirms that the database is online and open.



Next, we want to obtain a list of all indexes within this database. These titles are often indicative of the content. The following URL queries this data based on a target IP address of 34.80.1.1. The result can be seen in the following figure.

http://34.80.1.1:9200/_cat/indices?v

We now know there are several indexes which can be viewed. The first is titled "customer", but only contains a small amount of data (68 kb). Near the middle, we see an index titled "memberdisplayname20190903" which contains 124 MB of data including customer usernames. Some of these indexes contain email addresses and other sensitive data. Let's focus on the index titled "bank". We can view the content with the following URL.

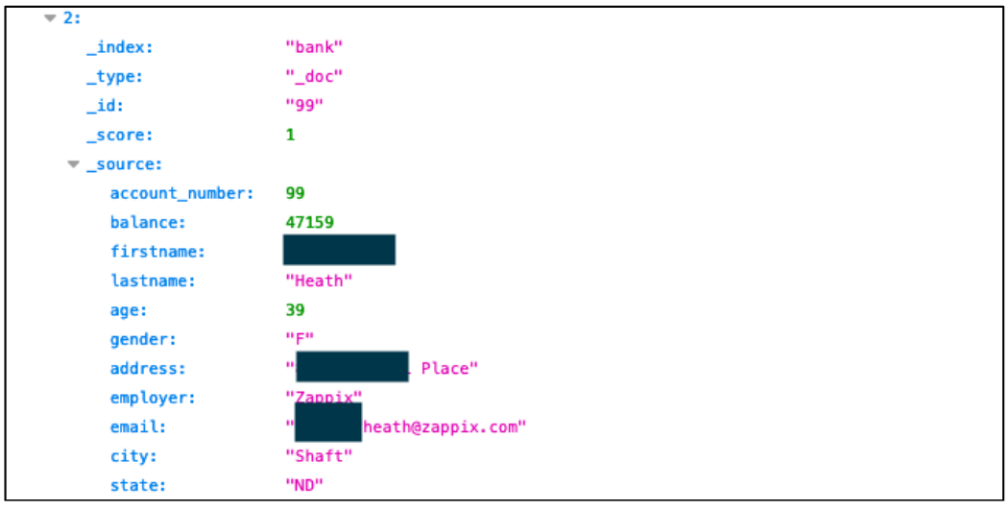
http://34.80.1.1:9200/bank/_search?size=100

This combines the IP address (34.80.1.1), necessary port (9200), name of the index (bank), and instructions to display the first 100 records (/_search?size=100). If we wanted to view the maximum number of entries visible within the browser, we would query 10,000 records with the following URL.

http://34.80.1.1:9200/bank/_search?size=10000

The second figure displays actual redacted results from this query. It identifies the first name, last name, email address, gender, city, state, bank account number, and balance associated with a customer. The rest of the index contains the same information for additional customers.

health	status	index	uuid	pri	rep	docs.count	docs.deleted	store.size	pri.store.size
green	open	customer	xYPN7VCPQHuRdEfP5vEkvg	5	1	10	0	68.3kb	34.1kb
green	open	kkday	3kWWvvGSTOa1AXMVOJa-eg	5	1	3514	115	22.3mb	11.1mb
green	open	minhash2	7aEhgxDDQ-2nu-Vpjpsibg	5	1	135617	2855	1.1gb	593.3mb
green	open	.monitoring-kibana-6-2019.09.18	6aE7CmBBQV2Iqk-8PeVV9A	1	1	74164	0	21.5mb	10.7mb
green	open	memberdisplayname20190903	OARMG9lLSP-rSOA49Pcc0g	1	1	220775	4567	124.6mb	62.3mb
green	open	.monitoring-kibana-6-2019.09.19	Yi7_DE_zTzaY7UCbIwVlWA	1	1	77753	0	22mb	10.9mb
green	open	streamtopic	KRzRxJ8eRd2Cpb5EG-tHXA	5	1	107379	771	211.4mb	105.7mb
green	open	articles20190114	5ZUblCdWQE6jo6JQK2J7sA	10	1	10531578	3885123	45.6gb	22.8gb
green	open	.monitoring-kibana-6-2019.09.20	taiMNlrvT3qMNTULnpFvEw	1	1	77579	0	21.5mb	10.8mb
green	open	articles20190130	4ZCi8yaxQvWN3hj2T0OECQ	10	1	0	0	5kb	2.5kb
green	open	member20181029	_8fDS93vSm-6hEECcEaLuA	5	1	7293911	20	94.1gb	47gb
green	open	.monitoring-es-6-2019.09.19	n0MGgBlbQrSE79xru9KlOw	1	1	346888	3384	461.6mb	231.9mb
green	open	articles20181224	7iO-SW40TPiORKtyj3qkJA	10	1	23422119	141719	114.1gb	57.3gb
green	open	minhash5	t6B40-XeRcq3Vx0NPEVAeA	5	1	223639	0	1.5gb	778.3mb
green	open	test	Q9fEJj9nRs6H4oXNvekTmg	10	1	674654	220802	29.9gb	15gb
green	open	.monitoring-es-6-2019.09.20	1j4OGJUDSJuG7EfSuQp6qA	1	1	354940	3666	478.9mb	239.6mb
green	open	.monitoring-es-6-2019.09.23	KgZ_kQUEThG5Olai27Yc9Q	1	1	346850	3102	487.5mb	245.2mb
green	open	minhash4	G9ablmkIQRCjyPx32-fVIA	5	1	223639	3	1.5gb	811.5mb
green	open	articles20181214	vbVscTkfTiCI64mlmHj44w	10	1	22582851	0	103.2gb	51.3gb
green	open	.kibana	6zZoT6nwTieIp3OlmTCUTw	1	1	4	0	44.8kb	22.4kb
green	open	.monitoring-es-6-2019.09.18	-oWrFwD-RBixBG-m4V2Osw	1	1	347378	3384	453.3mb	225.3mb
green	open	minhash3	yWXB2EJXS4itId5GhT-BLA	5	1	223639	2304	1.6gb	834.9mb
green	open	.monitoring-es-6-2019.09.22	Qg7IefQUSf65DUGlbOcNQA	1	1	347452	2820	474.2mb	236.8mb
green	open	rainytest	D1gkerx_Q0iOl34yDeDePw	10	1	36	6	1.4mb	745.9kb
green	open	.monitoring-es-6-2019.09.24	Z9FjEfovRHe3sBCisb4Zog	1	1	224408	2820	347.4mb	185.2mb
green	open	.monitoring-es-6-2019.09.21	IqK2eVxwSHyX_mIzQLpkfg	1	1	346849	2365	472.5mb	235.6mb
green	open	.monitoring-kibana-6-2019.09.22	PzU8OoxKQJuMBurDPHmbmQ	1	1	77756	0	20.5mb	10.2mb
green	open	.monitoring-kibana-6-2019.09.24	o8yEB59SRZm7GbQjfCpDiw	1	1	50202	0	17.2mb	6.3mb
green	open	minhash	hFJWrKTiTP2RWRjTiRf1_w	5	1	135617	0	2.4gb	1.2gb
green	open	bank	DK8E3aiEQsCZPulCsHWzeA	5	1	1000	0	950.5kb	475.2kb
green	open	customer_service	mCAKa4duQKKSUJYtA4Jsdw	5	1	5677	0	16.2mb	8.1mb
green	open	articles20190917	crlh_EJITQGPq65SKHaaOg	10	1	1016016	0	2.3gb	1.1gb
green	open	.monitoring-kibana-6-2019.09.21	RIFVLfu_RUCh8oThv6sbJA	1	1	77753	0	20.3mb	10.1mb
green	open	.monitoring-kibana-6-2019.09.23	xoeXY0MuSo2yhptUCG29bw	1	1	77753	0	19.8mb	9.8mb



Open Databases Issues

There are many complications with acquisition of open Elasticsearch data. The first is the record limitation. Displaying results within a URL limits the results to 10,000 records. Modifying the URL as described previously presents all data possible within the browser. Saving this page stores all of the content for future queries. However, many of the data sets I find contain millions of records. A Python script which parses through all results and stores every record is most appropriate, and is explained in a moment.

Next are the legal considerations. Technically, this data is publicly available, open to anyone in the world. However, some believe the act of manipulating URLs in order to access content stored within a database exceeds the definition of OSINT. I do not agree, but you might. I believe most of this data should have been secured, and we should not be able to easily collect it. The same could be said for FTP servers, paste sites, online documents, and cloud-hosted files. I believe accessing open databases becomes a legal grey area once you decide to use the data. If you are collecting this content in order to sell it, extort the original owner, or publish it in any way, you are crossing the line and committing a crime. I remind you that the Computer Fraud and Abuse Act (CFAA) is extremely vague and can make most online activity illegal in the eyes of an aggressive prosecutor. Become familiar with the data access laws in your area and confirm that these techniques do not violate any laws or internal policies.

My final reminder and warning is that I am not an attorney. I am not advising that you conduct any of these methods on behalf of your own investigations. I am simply presenting techniques which have proven to be extremely valuable to many investigators. If you believe that accessing an open (public) Elasticsearch database is legal in your area, and does not violate any internal policies, it is time to parse and download all content.

Elasticsearch Crawler (github.com/AmIJesse/Elasticsearch-Crawler)

In early 2019, I was made aware of an open Elasticsearch database which exposed sensitive data associated with 57 million people. In most cases, these records contained personal information such as first name, last name, email address, home address, state, postal code, phone number, and IP address. These types of records are extremely helpful to me as an investigator. Connecting personal email addresses with real people is often the best lead of all online research. I had found the database on Shodan using the methods discussed here. Specifically, I searched for an index titled "Leads" and sifted through any results of substantial size. Once I had located the data, I was desperate to download the entire archive. With a browser limit of 10,000, I knew I would need the help of a Python script.

I reached out to my friend and colleague Jesse and explained my scenario. Within a few moments, he sent me a small Python script. This file is now a staple in my data leaks arsenal of tools. He has agreed to share it publicly, please use it responsibly. It can be installed within Linux with the following commands.

```
cd ~/Downloads/Programs
git clone https://github.com/AmIJesse/Elasticsearch-
Crawler.git
cd Elasticsearch-Crawler
pip install nested-lookup
pip install release
```

You are now ready to download an entire Elasticsearch open database and specify which fields should be acquired. Note that you must open Terminal and navigate to your script in order to launch this utility. The following commands from within Terminal will launch the script.

```
cd ~/Downloads/Programs/Elasticsearch-Crawler
python crawl.py
```

This will present several user prompts to enter the target IP address, index name, port number, and fields to obtain. Let's walk through a real demonstration in order to understand the application.

I logged in to a free Shodan account and searched "product:elastic port:9200 leads" without the quotes. One hit was an Elasticsearch server in India. This database appeared to contain test data, so I will not redact any of the results. The IP address was 111.93.162.238 and the database was approximately 1 GB in size. Clicking the red square within the result on Shodan opened a new tab to the following address.

http://111.93.162.238:9200/

The brief response confirmed that the server was online. The URL discloses the IP address (111.93.162.238) and port (9200). This is the default port and is almost always the same. Now that I had the IP address, I launched a new browser tab and navigated to the following address.

http://111.93.162.238:9200/_cat/indices?v

This connected to the IP address (111.93.162.238) and port (9200), and then conducted a query to display all public indexes (/_cat/indices?v). The result included the following.

index	store.size
imobilestore	1.3mb
testcsv	190.7 kb
easyphix	1.8mb
index_test	7.2 kb
crazyparts	503.5 kb
valueparts	2.3mb
mobilemart	1.7mb
leads	280.8 kb

I usually look at both the names and the sizes. If I see an index titled "Customers", I know it usually contains people's information. If it is only 1 kb in size, I know it is too small to be of any use. When I see any index with multiple gigabytes of data, my curiosity kicks in and I want to see the contents. For this demonstration, let's focus on the index of our original search of "leads". I navigated to the following URL.

http://111.93.162.238:9200/leads/_search?size=100

This connects to the target IP address (111.93.162.238) and port (9200), loads the desired index (leads) and displays the first 100 results (/_search?size=100). This is usually sufficient to see enough target content, but this can be raised to 1000 or 10000 if desired. Below is a record.

"_index": "leads","_type": "leads",
"_id": "PXIhqmUBcHz5ZA2uOAe7",
"_source": {"id": "86",
"email": "test80@agencyinnovations.com",
"first_name": "test80","last_name": "test80",
"phone": "32569874",
"ip": "0.0.0.0",
"orgname": "Sales Arena",
"isDeleted": false,
"created_at": "2018-09-05 19:57:08",
"updated_at": "2018-09-05 19:57:08",

This is obviously test data, but assume it was a record containing a real person's name, email address, and phone number. Also assume there were over a million records within this index, which is quite common. We could save this page, but would be forced to save the undesired fields such as "tags" and "_source". Also, the data would be in a difficult format to search. This is where our new Python script is helpful. You have already launched the crawl.py script, and should have been presented with a prompt for the IP address of the target. The following displays each entry I submitted for this demonstration.

```
IP address: 111.93.162.238
Index name: leads
Port (Default is 9200): 9200
Field values to obtain (submit an empty line when finished):
Value: email
Value: first_name
Value: last_name
Value: phone
Value:
```

After being prompted for the IP address (111.93.162.238), it asked me for the target index name (leads) and port number (9200). It then prompted me to enter the first field I wanted to acquire (email). Since I entered a field, it then prompted for the next field (first_name). The tool will continue to ask for field names for as long as you provide them. Notice there is an empty line in the last "Value". This empty result tells the script you are finished, and it begins collecting the data. When finished, a text file will be saved in the same directory as your script. In this example, it was at ~/Downloads/Programs/Elasticsearch-Crawler/111.93.162.238-leads.txt. The title of the file was automatically created to reflect the IP address and name of the index. The following are the first three lines of this text file.

```
test65@agencyinnovations.com,test65,test65,987485746
test22@agencyinnovations.com,test22,test22,124958616
test69@agencyinnovations.com,test69,test69,2145968
```

If this were real data, you would see millions of people's email addresses, names, and telephone numbers. There are likely hundreds of legitimate databases on Shodan right now, just waiting to be found. The next time you see a news article about a security researcher who found an exposed database containing millions of sensitive records, it is very likely that Shodan and a similar script was used. If your downloaded file contains random text, you have likely encountered a patched version of Elasticsearch. At the time of this writing, Elasticsearch databases version 6.4.0 and newer were blocking the script. Anything older worked fine.

I predict we will see fewer open databases in the future. While we still hear about sensitive leaks every week, word is spreading and companies are locking down their data. This is a good thing for all of us. Until then, I will continue to search.

The JSON Dilemma

If you are lucky, you will always find a nicely organized CSV file with your new data, ready for "cutting" and storing. However, that is not always the case. Many times, you will encounter JSON data. At first, it may seem impossible to clean and minimize. The following is a partial example of JSON data extracted from a large file.


```
"status": 200,
"likelihood": 6,
"data": {
  "id": "qEnOZ5Oh0poWnQ1luFBfVw_0000",
  "full_name": "sean thorne",
  "first_name": "sean",
  "middle_initial": "f",
  "middle_name": "fong",
  "last_initial": "t",
  "last_name": "thorne",
  "gender": "male",
  "birth_year": 1990,
  "birth_date": null,
  "linkedin_url": "linkedin.com/in/seanthorne",
  "linkedin_username": "seanthorne",
  "linkedin_id": "145991517",
  "facebook_url": "facebook.com/deseanthorne",
  "facebook_username": "deseanthorne",
  "facebook_id": "1089351304",
  "twitter_url": "twitter.com/seanthorne5",
  "twitter_username": "seanthorne5",
  "work_email": "sean@peopledatalabs.com",
  "personal_emails": [],
  "mobile_phone": "+14155688415",
```


The full portion for this single entry would have filled four pages in this book. Now imagine that the data set included 100 million entries. That would be 400 million pages if you wanted to print the data. If you were to query this selected data with Ripgrep, it would either only present the line where something was found (without any other data), or the entire entry (several pages of compacted text). If I searched "sean@peopledatalabs.com" via Ripgrep, the result might be "work_email": "sean@peopledatalabs.com" if the data was separated with line breaks. We would not receive all of the other associated content. You could use a lot of sed commands to try and move each piece of data into one line for queries, but that would be time consuming and unnecessary.

When I encounter JSON data, I rely on **jq** (stedolan.github.io/jq/). Since we previously installed jq, we are now ready to launch it within Terminal (or Command Prompt). Assume we have a large file of JSON data, which contains the previous file structure, titled people.json. The following command would begin our extraction.

```
jq --raw-output
'"\(.data.first_name),\(.data.last_name),\(.data.gende
r),\(.data.birth_year),\(.data.birth_date),\(.data.lin
kedin_username),\(.data.facebook_id),\(.data.twitter_u
sername),\( .data.work_email),\(.data.mobile_phone)"'
people.json > people.txt
```

Let's dissect this:

jq	This command launches jq.
--raw-output	This outputs pure text without formatting.
'"	This begins our string.
\(.data.first_name),	This extracts the first name field.
\(.data.last_name),	This extracts the last name field.
\(.data.gender),	This extracts the gender field.
\(.data.birth_year),	This extracts the birth year field.
\(.data.birth_date),	This extracts the birth date field.
\(.data.linkedin_username),	This extracts the linkedin username field.
\(.data.facebook_id),	This extracts the facebook field.
\(.data.twitter_username),	This extracts the twitter field.
\(.data.work_email),	This extracts the work email field.
\(.data.mobile_phone)	This extracts the phone field (no trailing comma).
"'	This ends our string.
people.json	This identifies the input file.
>	This notifies of an output request.
people.txt	This identifies the new output file.

The result appears below.

```
sean,thorne,male,1990,null,seanthorne,1089351304,seant
horne5,sean@peopledatalabs.com,+14155688415
```

If you had a JSON file with 100 million entries, you might have several billion lines of unsearchable data. You would now have a single text file with the 100 million total entry lines, only containing your desired content. Notice in my command that I have "data" preceding each field request (`\(.data.gender)`). That is because those fields are under a section titled "data" within my JSON example on the previous page. If there was a section called "data" which contained a section called "people" which then contained a field called "gender", that field request would appear as follows.

```
\(.data.people.gender),
```

An entire chapter could be devoted to jq and its usage. If you find this tool valuable, I encourage you to visit the Github page for the project. It includes complete usage and tutorials. Now, let's test jq with some real data. The following pages access data leaks or previously public information.

People Data Labs Scrape (PDL)

In 2019, a security researcher claimed to have scraped an open database of over 1.2 billion PDL records including names, locations, and social network content. This data set is still floating around, and it is valuable. I know of someone who joined a channel called "Data stream" on Telegram and searched "people data labs". This person received a result of a torrent file called "PeopleDataLabs_416M.json.7z.torrent". They downloaded this file and opened it via Transmission, which downloaded an 11 GB compressed file. Extracting the file resulted in a new folder with a single file titled PeopleDataLabs_416M.json which contained 67 GB of data. Someone else had already parsed the JSON data to a single line for each entry, which appeared similar to the following.

{"a":"san francisco, california, united states","t":["14155688000"],"e":["sthorne@xxregon.edu","sean@peopledatalabs.com","sean@txxxentiq.co"],"liid":"**seanthorne**","linkedin":"https://www.linkedin.com/in/**seanthorne**","n":"sean thorne"}

This format could be sufficient for most readers, and no cleaning would be required. Since I try to eliminate any unnecessary data, I would consider the following commands, which will get rid of the brackets, quotes, and category identifiers.

```
sed -i "s/[\{]//gI" PeopleDataLabs_416M.json
sed -i "s/[\}]//gI" PeopleDataLabs_416M.json
sed -i "s/[\[]//gI" PeopleDataLabs_416M.json
sed -i "s/[\]]//gI" PeopleDataLabs_416M.json
sed -i "s/\"//gI" PeopleDataLabs_416M.json
sed -i "s/a\://gI" PeopleDataLabs_416M.json
sed -i "s/t\://gI" PeopleDataLabs_416M.json
sed -i "s/e\://gI" PeopleDataLabs_416M.json
sed -i "s/liid\://gI" PeopleDataLabs_416M.json
sed -i "s/linkedin\://gI" PeopleDataLabs_416M.json
sed -i "s/n\://gI" PeopleDataLabs_416M.json
```

Each command will take some time due to the size of the file. Each process on my MacBook Pro required ten minutes to complete. Notice the first four commands include brackets ([]) around the characters I want to remove ({}) and ([]). This is required with some special characters, especially any type of opening or closing bracket. I hope you see by now that many of these demonstrations are not necessarily to help you collect and clean data leaks, but more to teach you the commands which will be most helpful to overcome obstacles in your own data collection journey.

The result should be a 50 GB text file. The following Ripgrep command would search my last name through the data.

```
rg -aFiN bazzell PeopleDataLabs_416M.json
```

LinkedIn 2021 Scrape

In 2021, an unknown group released a huge data set which contained over 400 million user profiles from LinkedIn. Profiles are public, so the headline did not seem interesting at first. However, this data included over 140 million email addresses and numerous cellular numbers associated with the profiles. That caught my attention. It is suspected that the data was acquired by querying email addresses and cellular numbers from previous leaks through the official LinkedIn API in order to identify the associated profiles. Additional public data was then used to connect Facebook and Twitter accounts. The result was 1.4 TB of JSON files which contained the personal email addresses and cellular numbers of many LinkedIn users. I wanted it, but in a better format.

Currently, there is an active Torrent containing the entire data, which consists of 700 compressed files, each approximately 300 MB in size and titled similar to "part-00055.gz", for a total of approximately 200 GB. Once extracted, the data is 1.4 TB, and the files are titled similar to "part-00055" with no file extension. Searching "400M LinkedIn Torrent" on Google should begin your hunt, or searching the previous file name might lead you to a torrent. There is also a copy available on "NitroFlare", but the free download tier can be quite slow. Searching "Linkedin Scrapped Data 2021" should find it. I downloaded the Torrent in an afternoon. I then decompressed all the files.

Each file contains JSON entries which possess a lot of redundant data I do not want in my final copy, similar to the previous example. However, I do want the fields titled first_name, middle_name, last_name, gender, birth_date, linkedin_username, facebook_username, twitter_username, work_email, mobile_phone, location_name, phone_numbers, and emails. I placed all of the "part" files in my Downloads folder and executed the following from that path.

```
jq --raw-output
'"\(.first_name),\(.middle_name),\(.last_name),
\(.gender),\(.birth_date),\(.linkedin_username),\(.fac
ebook_username),\(.twitter_username),\(.work_email),\(
.mobile_phone),\(.location_name),\(.phone_numbers),\(.
emails)"' part* > LinkedIn.txt
```

This created a single file titled LinkedIn.txt, which only contained my desired data. It was 49.68 GB in size and possessed 400,101,052 entries. It contained a lot of entries containing "null" which offered no value, so I removed them with the following.

```
sed -i "s/null\,//gI" LinkedIn.txt
```

The final result was a single 37.46 GB file. I can query this data within a few seconds and only see valuable information. That is a big difference from querying the original 1.4 TB data, which took up to an hour per query and displayed 90% useless data. The following is a slightly modified and redacted (xxx) entry from the new file. It displays my target's first name, middle initial, last name, gender, date of birth, LinkedIn username, Facebook username, Twitter username, location, telephone number, and email address.

```
chelsea, a, jansen, female, 1982-0x-xx, chelsea-
jansen-685axxx, chelsea.
jansen.xxx, chelseaxxx, hartland, connecticut,
+18608191xxx, cjansen@xxx.org
```

I can now associate the name, email address, cellular number, Twitter account, Facebook account, or date of birth with any of the other details. I can query a name and often identify the email address behind it. I can query a Facebook username and possibly identify the true person or date of birth. When I find a target on LinkedIn, I can query the unique portion of the URL (`chelsea-jansen-685axxx`) and see all information from this leak, including a valid email address. While this content does not possess all data within every field, it has been quite beneficial to my investigations. Consider the following examples.

- I searched the LinkedIn profile "seitzjustin" through the data and received the user's full name, Twitter handle, location, and a new email address from a custom domain which I did not previously possess. Historical records of this domain identified a physical address and telephone number.
- While stranded after a canceled flight, I queried the LinkedIn username of the airline's CEO and identified a personal Gmail account and cellular number. I made direct contact requesting a solution.
- A client was receiving harassing messages via Twitter. The Twitter handle of the suspect was associated with a LinkedIn profile and a 33Mail email address. The subdomain of the email address was only assigned to this subject, and a query through additional breach data revealed online profiles associated with the true name.

I hope you can see the power of this type of data. Cleaning the set from 1.4 TB to 37 GB took some time and effort, but I now have a small file for my queries, which completes in less than 30 seconds from an external SSD. I placed this file in the "People" folder of my external drive, which is explained later.

IntelligenceX Scrape

In 2021, a disgruntled IntelX customer "scraped their scrapes". Today, the 80,000 copied documents, which were originally scraped from Pastebin before they terminated their free API, have been merged into one folder possessing sensitive data associated with 46 million email addresses. Many people have successfully downloaded the 1.6 GB compressed file titled "intelx_scraped, pass pompompurin.7z" from the popular Telegram channel called "LEAKS AGGREGATOR". Those who did reported that the file could be decompressed with a password of "pompompurin", but the compression program Keka was required for macOS users. The result was over 7 GB of text files. Let's make them easier to use. Navigate to the folder which stores the 87,813 files and conduct the following.

```
cat * > _IntelX.txt
```

I set you up for failure here. There are too many files for the Cat command. Instead, we must create a command which "finds" all of the files first, and then allows Cat to do its job.

```
find . -name '*' -exec cat {} \; > ~/Downloads/IntelX.txt
```

I instructed my machine to find every file in the directory (*), and then execute (exec) the Cat command, but save the result to a different folder (~/Downloads/IntelX.txt). The resulting file is the same size as the original files, but easier to work with. However, it is a mess of Pastebin scrapes which have no email addresses. I only want lines which include an email address, and no lines which do not include an email address. Therefore, the following command extracts any line which includes "@" and saves it to a new file.

```
cd ~/Downloads
awk '/@/' IntelX.txt > IntelX2.txt
```

The result is a 4.27 GB file. It still includes a lot of junk, but we can now sort for unique entries to remove any duplicates with the following.

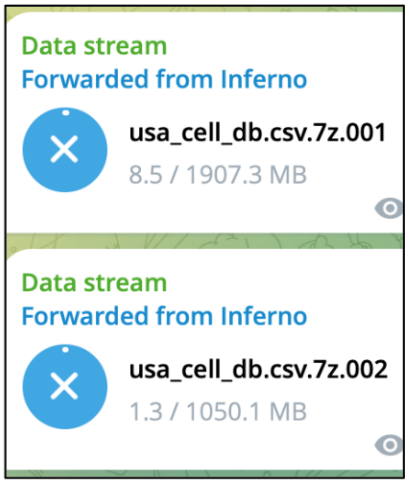
```
LC_ALL=C sort -u -b -i -f -S 80% --parallel=8 IntelX2.txt
> IntelX3.txt
```

This brought us down to 2.5 GB, which is manageable. Searching my last name with Ripgrep revealed hundreds of email addresses and credentials, as seen below (redacted).

Telephone Number Leaks

Some of the leaks previously mentioned include telephone number details, but that type of content was not the main attribute of the data. On the contrary, there are several large leaks which focus exclusively on telephone numbers, and I find these very valuable. In fact, I query target telephone numbers almost as much as I search for email addresses. Many people possess numerous "burner" numbers which are not registered to them, and do not return any results within online searches. However, many of these people will rely on those numbers throughout various social networks and websites, causing them to appear within various leaks, breaches, and logs, which often identify the user. Let's walk through a few examples together.

There are dozens of large telephone number data sets, most of which include "cell" and "phone" within the file names. I searched "USA cell" within Telegram, and immediately received several results. The two visible in the following images appeared because I had previously joined the "LEAKS AGGREGATOR" and "Data stream" channels.



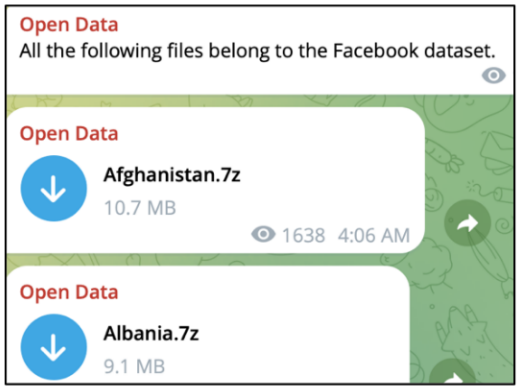
The 2.6 GB compressed file for the 183M USA Cell database was password protected. Downloading the image directly above the download link identified the password (t.me/HadesBeaches), which produced an 11.5 GB CSV file. The 3.1 GB USA Cell files produced a 19.82 GB CSV file. After closer examination, these files contained identical data, but the first option (183M USA CELL) did not contain quotation marks or other unnecessary characters. Therefore, I deleted the second set of data. A random (modified) entry appeared as follows.

MIKE,COLE,105 VALLEY DR,HOUSTON,TX,77042,7135910000,
MIKE.COLE@GMAIL.COM,AT&T MOBILITY,MALE

In this scenario, I now possess the names, physical addresses, and email addresses associated with 183 million cellular numbers in America. As a further value, I can see the service provider for each number. I can use Ripgrep to query through the data.

Facebook Telephone Number Leaks

In 2021, someone published the personal data of 533 million Facebook users, which included cellular telephone numbers associated with most of the accounts. The exposed data included account information from 106 countries, including over 32 million records on users in the US and 11 million on users in the UK. Most records included telephone numbers, Facebook IDs, full names, locations, and birthdates. As you may have guessed, the data is now widely available online. A quick search within the "Open Data" channel for "facebook dataset" on telegram revealed the files for each country, as seen in the following partial image.



Each country possessed its own file, and the "USA.7z" file was 669 MB compressed, which expanded to eight new text files totaling 3.28 GB. I combined all files into one with the following command.

```
cat USA* > Facebook-US.txt
```

A typical entry appeared as follows.

17244540000:9304000:Sara:Rose:female:Pittsburgh, Pennsylvania:Greensburg, :::::

This displays the cellular telephone number, Facebook user ID, name, gender, and location of every victim. I did not see much value in cleaning this data further , so I left it as-is. Adding the Facebook ID to a URL navigates directly to each user. In this fake example, the page at https://facebook.com/9304000 opens the user's profile. I combined all country sets into one large file (82 GB) with the following command.

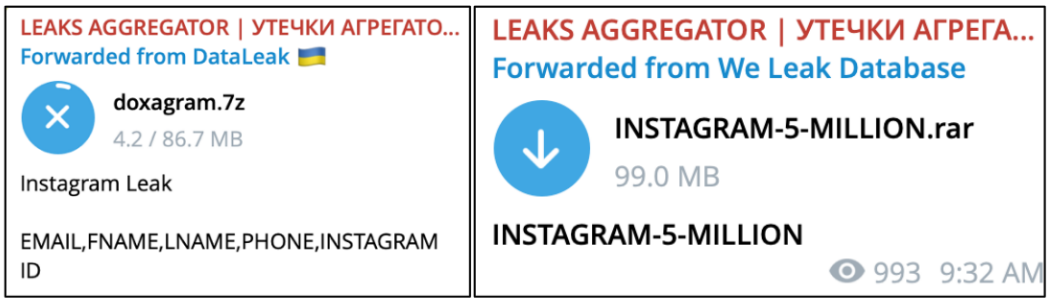
```
find . -name '*.txt' -exec cat > ~/Downloads/FB.txt {} \;
```

I am always skeptical of this type of data, so I conducted a quick test. I copied the telephone number in a random entry, such as the previous example, and searched the number within a reverse caller ID service (as explained in ***OSINT Techniques, 10th Edition***). The result follows, which provided high confidence in the data.

"caller_name": "SARA ROSE",

Instagram Telephone Number Leaks

The previous Facebook leak was due to a poorly secured server. At one time, people could execute bulk queries of telephone numbers against the Facebook Application Programming Interface (API) and receive the profile IDs. Most of these holes have been plugged, but much data still flows freely. I have seen many Instagram telephone leaks which were likely obtained through the same technique. During this writing, I found two compressed files within the LEAKS AGGREGATOR Telegram channel.



The 86 MB file decompressed into five files totaling 288 MB containing 4,939,354 entries. The following is a modified example.

nast@charter.net,nicholas,ast,2013374000,@nast

This produces data which can be queried by email address, telephone number, or Instagram username. The "5 Million" file contained identical data with different file names. This is very common as people republish the data in order to appear as if they possess something special. Attempting to identify redundant data before you add it to your collection is important to reduce query times.

Searching "Instagram" or "IG" within your subscribed Telegram channels can produce valuable data for download. This is why it is so important to join as many data-related channels as possible. This widens the net. You will often identify breaches which have nothing to do with Instagram, yet still possess Instagram user data. During this writing, I received several search results of breached SQL files which contained entries for Instagram profile IDs as part of their data collection about their customers. SQL files are discussed later in the breaches chapter.

Telemetry (telemetryapp.io)

This service requires an account before searching, but allows five free queries per day. You can enter any term and identify Telegram rooms which contain your information. This will never be complete, but it may locate information about your target which could not be found otherwise.

Summary

We have only begun the journey into leaked data. I have personally obtained, cleaned, stored, and queried tens of thousands of valuable data leaks as part of my online investigations. At least once a week, I identify my target's home address through leaked data after all online efforts have been exhausted. I have had countless suspects remove all of their details from people search websites in order to disappear, but they can never edit my own data collection. Aside from the data presented here, there are countless copies of other data sets floating around. It is now up to you to locate and clean the data. Later, we will use a script to help us quickly query all of the data leaks you locate.

CHAPTER SIX
DATA BREACHES

Now that you understand the basics of searching and modifying data, we should discuss the true value of this technique. Until now, we have only experimented with publicly-available data. The real power lies within the stolen databases that are now publicly available which I am about to discuss. There are usually two types of people that download these databases. The first are amateur criminals that use the data to illegally obtain account access to victims' accounts by supplying the usernames and passwords from one service into another service. As an example, if a criminal learns your username and password that you used on LinkedIn, he or she may try that same combination on Twitter. We will never do this, as it is illegal and unethical.

The second group of individuals who download this data are security researchers. Numerous private organizations authorize employees to collect this stolen information and use the data to make their own systems more secure. I personally know of many researchers who download this data, protect it from further distribution, and only use it in an ethical manner. This is the only acceptable use of this data. For those that still believe that we should not access stolen databases released to the public, consider one last argument. Tens of thousands of criminal hackers use this content to "dox" people every day. Doxing is the act of exposing personal information about a victim online. This is a very common tactic used against law enforcement officers when an event such as an officer-involved shooting happens. The government employees' passwords often get leaked on Pastebin or other similar sites. This stolen data is already out there, and it can never be secured. We should embrace the same techniques used against us when researching criminals. Enough warnings. Use your best judgment.

By now you may just want to know where to obtain these databases. The source websites which allow download of this stolen data get shut down often. However, the magic of internet archives can help us regain the data. The next exercise also involves archive.org. Specifically, data from LinkedIn. This version surfaced in 2017 and does not possess any passwords. It contains only LinkedIn user ID numbers and the associated email addresses. While this type of data was included within the original 20 GB LinkedIn breach, it was excluded from the popular password dumps which can be found online today. Since it does not contain passwords, it can be found on the Wayback Machine. At the time of this writing, the direct link to download the entire user ID database could be found at https://archive.org/details/LIUsers.7z.

Decompress this file to your Downloads folder. The text file contains the email addresses and user ID numbers extracted from the full LinkedIn breach. It is nothing more than each email address used to create a specific profile ID number. The following search within Terminal would display the results of a target email address, and the response is directly after the command.

```
rg -aFiN janewilson@microsoft.com linkedin_users.txt
```

1332567, janewilson@microsoft.com

You now know that your target's user ID number is 1332567. This data alone is not very helpful. Let's consider another perspective. Assume you find your target's LinkedIn profile page, and you want to know the email address used to create the account. Right-clicking on the profile allows the option to display the "Source Code" of the profile. Searching the term "member:" within this code presents numerous occurrences of that term. The occurrence toward the end should appear similar to "member:1288635". This tells you that the member ID for your target is 1288635. The following displays the email address associated with that number, with the results immediately below.

```
rg -aFiN 1288635 linkedin_users.txt
```

1288635, John.h.007@gmail.com

I have used this technique numerous times over the past several years. During one investigation, I located the LinkedIn profile of my target. I had previously found no email addresses connected to him. By looking at the source code of the LinkedIn profile, I could see his user ID. Searching that user ID within this LinkedIn data provided the personal email address used to create the account. That email address led to old passwords, which led to additional email addresses, as explained in a moment. Obtaining a confirmed personal email address can lead to numerous new investigation techniques. Note that member numbers only exist for accounts prior to 2015.

Credentials

Now that we are dipping our toes into the waters of "stolen" data, we should consider the big breaches. You have likely heard about LinkedIn, Dropbox, Myspace and Adobe being hacked. All of those breaches have been publicly released by various criminal organizations. At the time of this writing, most remaining sites which possessed this huge collection of stolen data had been shut down. However, the data lives on. Numerous "hacking" groups have collected tens of thousands of breached databases, both large and small, and combined them into credential lists. These lists contain only the email addresses and passwords of billions of accounts. They do not identify which breach each credential originated, but the data is extremely valuable for investigators. Before we analyze the content, we must obtain a full copy of the data. This is where things get tricky (again).

My attorney says I cannot HOST links to any of this content, which I understand. She also says that I cannot display any direct links to an identifiable breach, such as LinkedIn or Adobe. This would make me a target for a lawsuit, and seems like great advice. My only option is to "identify public resources which link to data without attribution of a specific originating source". In other words, I can tell you where you may find this "Combo List" content, but it is up to you to take action to obtain it. Let's connect our VPN; understand any employer policies which might prevent the following actions; and tiptoe into the world of stolen credentials.

COMB (Compilation Of Many Breaches)

In 2021, an unknown group of credential thieves created a huge collection of 3.2 billion credentials consisting only of email address and password combinations. This data set was titled "Compilation Of Many Breaches", otherwise known as "COMB". There have been numerous other combo lists released in previous years, such as Anti-Public, Exploit.in, and others. However, this set included a better search structure and ability to conduct queries within seconds. Some people may find that searching for "CompilationOfManyBreaches.7z" within Google might lead you to a copy. Others report that the following queries will display the torrent file for this data.

site:github.com "CompilationOfManyBreaches.7z"
site:github.com "1.4 billion"

At the time of this writing, the single result from this search possessed a torrent file which could be opened with a torrent software application such as Transmission. This program will download the 20 GB file to your computer. Depending on your internet connection, this entire download can take minutes, hours, or days to finish. If possessing user credentials violates your security clearance or organizational polices, do not proceed with this download.

Let's assume you now have a drive with the entire contents from COMB. The result is a 20 GB compressed file which contains all 3.2 billion credentials. The file itself is titled "CompilationOfManyBreaches.7z". If you chose to download this file, it must be decompressed. I prefer a utility such as 7-zip for Linux and Windows, or Keka for Mac. Upon decompression, you might be prompted for a password. A Twitter post located at https://twitter.com/BubbaMustafa/status/1370376039583657985 claims the required archive password is "+w/P3PRqQQoJ6g". Once decompressed, you should see a new folder titled "CompilationOfManyBreaches" which is 106 GB in size.

Assume you possess a folder called COMB within your external SSD which contains the data downloaded during this exercise. This folder should contain the three files and folder labeled "data". Within Terminal, navigate to that "COMB" folder. My Linux command for this was `cd "/media/$USER/DATA/COMB"`. macOS users would use `cd "/Volumes/DATA/COMB".` In Linux, you could also find the external drive within the Files application, right-click, and choose "Open in Terminal". You should now be within your new data collection folder inside Terminal. We can conduct searches to start evaluating the benefit of this data. Since COMB includes a fast search option, we will start with it. However, we should make sure that our script is executable with the following command (on both macOS and Linux).

```
chmod +x query.sh
```

Execute the following command in Terminal from within the COMB folder.

```
bash ./query.sh michaelbazzell@gmail.com
```

This should result in no hits within a few seconds. Now try the following (notice the period in the email address).

```
bash ./query.sh michael.bazzell@gmail.com
```

The result should appear as follows.

michael.bazzell@gmail.com:password

This identifies a password "password". The first query failed because we did not include the "." within the search parameter. This tool can be fairly unforgiving and requires exact data. The following search provides any email address which begins with "michael.bazzell" with results which follow.

```
bash ./query.sh michael.bazzell
```

michael.bazzell@gmail.com:password
michael.bazzell@us.army.mil:redacted10
michael.bazzelle1970@yahoo.com:redacted201

This search tool is extremely fast. If you know the email address of your target, or at least the first portion of the address, searching through the native COMB query option is best. However, this presents limitations. You cannot use this tool to search a specific domain or password. For that we will once again rely on Ripgrep. The following queries will all assume that you have already opened Terminal and have navigated to the folder where your data is stored. The next search would attempt to identify a specific email address within all the files. Note that results were modified to protect the privacy of the individual. As a reminder, this command applies our parameters and identifies the target data to be searched. The result follows the command.

```
rg -aFiN mikewilson@microsoft.com
```

mikewilson@microsoft.com:bigbucks55

We now know that at some point a password of bigbucks55 was used in conjunction with an online account associated with our target email address. Would this password still work with any online accounts? Possibly, but we will never know. Attempting to log in to an account with this information is a crime. Instead, think about other ways that we could use this data offline. We know our target's work email account, but we may want his personal account. Since many people recycle the same password across multiple accounts, let's conduct a search for the password.

```
rg -aFiN bigbucks55
```

The results include accounts associated with our target, and some that are not. The more unique your target's password, the better quality of associated content you will receive. The following data was presented during my search. Notice the ways that data was displayed based on the search, and how any portion of the results were included. Since we specified our search parameters, we received results regardless of case.

bigbucks551@yahoo.com:towboat@1
bigbucks55@hotmail.co.uk:towboat@1
bigbucks55@hotmail.com:towboat@1
prizeunit@yahoo.com:bigbucks55
mike.wilson5@gmail.com:BigBucks55
mikewilson@microsoft.com:bigbucks55

This tells me that the last two results are very likely my target, since the names are similar and the passwords are almost identical. I can now assume that my target's personal email account is mike.wilson5@gmail.com. The first three accounts could be my target, but are probably not. This is likely just a coincidence. However, we can assume that the owner of one of those accounts is the owner of all three since the passwords are identical. The fourth response is a toss-up, but worth further investigation. We can also use this to query all credentials from a specific domain.

```
rg -aFiN @altonpolice.com
```

One of the results is quite embarrassing, as follows. I promise I have not used that password since the late 90's.

bazzell@altonpolice.com:mb01mb01mb

The tactic of searching leaked passwords and recovering associated accounts is by far the most successful database leaks strategy that I have applied to my investigations. In 2017, I was assisting a federal agency with a child pornography investigation where a suspect email address had been identified. This address was confirmed as being connected to the original person that had manufactured illegal videos of children being molested, but an alias name was used during creation. A search of this address through a breach compilation revealed a unique password associated with the account. Searching this password revealed only one additional email account, which was the personal Gmail account of the suspect (in his real name). The suspect used the same password for both accounts. The primary investigator made an arrest the next day after confirming the connection.

While some will say that we should never download leaked databases that contain personal login credentials, I disagree. There is great potential value in these data dumps that could solve cases on a grand scale, and make a huge impact on the prosecution of serious criminals.

I cannot overstate that this instruction is the very tip of the iceberg. There are tens of thousands of compromised databases online, and more being published every day. If you find "COMB" to be valuable, you may consider researching others. If you invest some time into seeking the sources of this data, you will quickly become overwhelmed at the mass amounts of content to properly obtain, clean, and store. I can only discuss the basics here. It is your job to proceed if you choose.

As a start, you may consider focusing on public credential leaks on Pastebin (pastebin.com). When you search an email address on Pastebin via Google, the results often include paste dumps. Clicking on these links will take you to the original paste document, which likely has many additional compromised credentials. A query of test@test.com on this site will likely present hundreds of data sets ready for download. Any time I see on HIBP that a new public leak has surfaced, I search for that specific data with a custom Google search. If I saw that Myspace has been exposed, I would use the following.

"myspace" ext:rar OR ext:zip OR ext:7z OR ext:txt OR ext:sql

There are many online database resources that will sell you the data. Please avoid these. First, you will likely get ripped off a few times before you find an "honest" seller. Second, you are giving money to criminals, and I don't like encouraging that behavior. Many researchers that I know possess over 100,000 databases which contain over four terabytes of total information, all found on public sites. A single query of this data can take a few minutes, even on a fast machine.

RaidForums Databases

In the previous chapter, I explained a method to open a torrent file which possessed voter registration data from several states. You may still have this torrent loaded into Transmission. If not, revisit that instruction to get it launched. Once opened, you should see 459 GB of databases which originally appeared on RaidForums. You should see data breaches from many familiar companies within this torrent, but I do not want to get sued by conducting demonstrations on them. Instead, I will focus on a few databases which were stolen from criminal marketplaces. I hope you see the irony. Criminals will also steal from other criminals.

I downloaded the databases associated with CrackingForum.com, CrackingItaly.com, and DemonForums.net. All three provide an online community which caters to criminal hackers. Let's jump into each.

The CrackingForum.com folder possesses a single text file which includes the email address and plain-text password of all 469,550 users of the site. This is the easiest data to consume, and the file is ready to be archived to your SSD. There is no cleaning to be done. If your email address target appears here, you know that suspect was a member of this hacking site, and you also see their password.

The CrackingItaly.com folder possesses a single text file which includes the member ID, full name, email address, hashed password, and IP address of all 18,132 users of the site. Again, no cleaning necessary here. This is a fairly minimal database. If your suspect email address appears here, you might learn their full name and an IP address which could identify their home internet provider (and approximate location). I will explain hashed passwords in a moment.

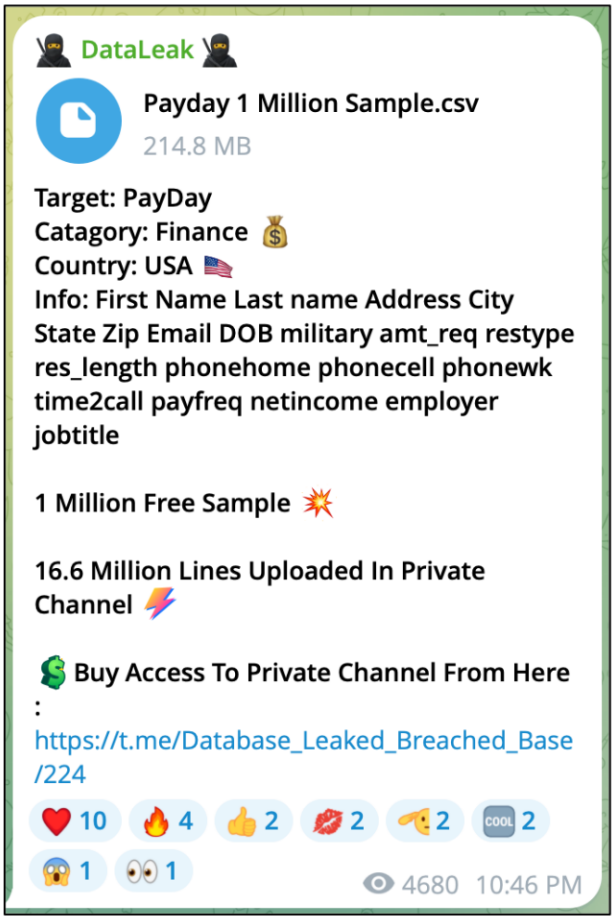
The DemonForums.net folder possesses a single SQL file which is not as straight-forward as the others. It is small enough that we can open it within a traditional text editor. The default option within Linux is fine with me. Apple macOS users might prefer the free version of BBEdit. When I open a SQL file, which is usually an exported backup of a database, I want to find the "Users" section. It is often toward the end of the file. In this scenario, searching "mybb_users" takes you straight to it. I then removed all of the data before this section and after, leaving only the 11,614 lines of user content. I then used my text editor's find and replace feature to remove all of the single quotes, and saved the file. This produced a 6.7 MB file which is quite small. I could have used the Cut command to extract only the desired columns of data, but that is overkill for such a small file. I can now query through usernames and email addresses of all 11,614 users of the service.

Only you can decide if you should proceed with downloading the remaining 450 GB of databases which were originally offered on RaidForums. If you do, many surprises await. Use the overall lessons throughout this guide to clean the data when needed and query the results. Many readers report that this torrent will only download 99% of the data. If that happens, archive as much of the content which is visible and move on.

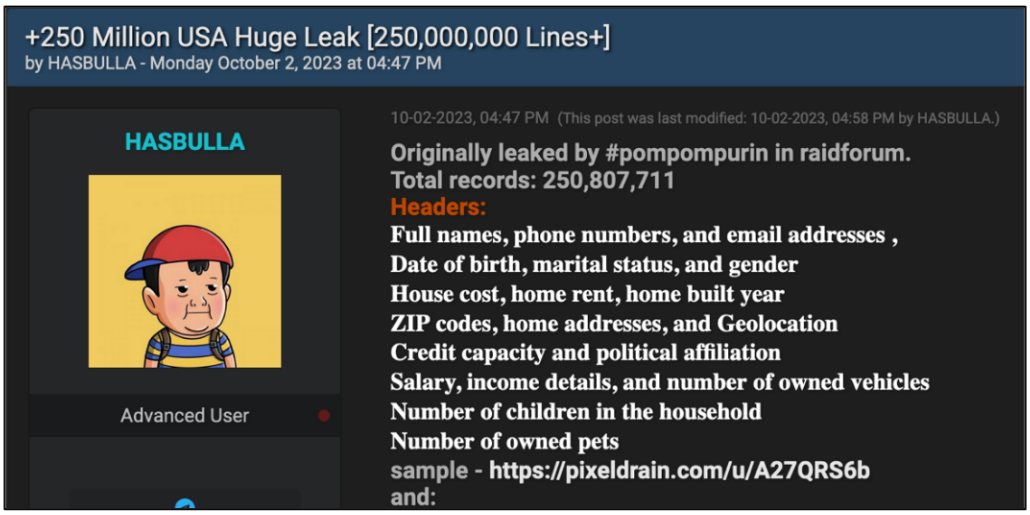
Telegram Search

In the previous chapter, we relied on Telegram to find voter information. We will also rely heavily on Telegram to find stealer logs in the next chapter. Always check Telegram for breach data, as downloading from there is easier than various websites. Subscribing to channels such as "We Leak Database", "LEAKS AGGREGATOR", "DataLeak", "Open Leak", and "Leakbase Official" should present many breaches over the years. It also allows you to query the contents of these rooms with the general search field in Telegram. Anytime I hear about a new breach, searching the company's name within Telegram usually reveals some evidence. Let's conduct a few random demonstrations.

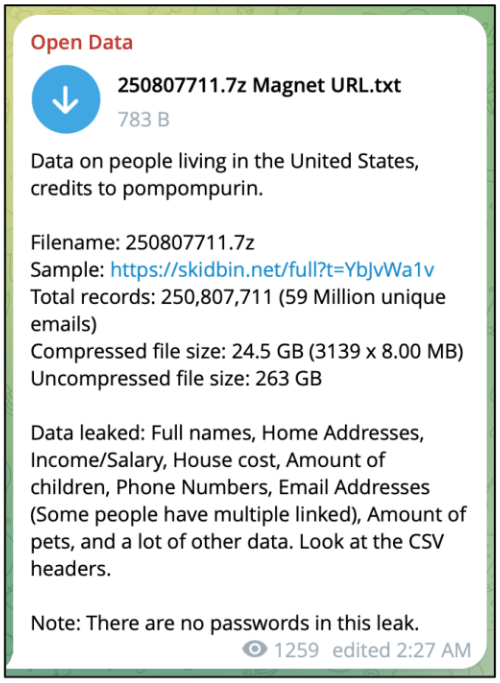
While writing this chapter, I saw a post on BreachForums.is discussing an offer of "USA / Payday Loans Database / 1 Million / 2020". As previously stated, I prefer to never earn, buy, or spend any "credits" on these types of sites. Instead, I searched the term "Payday" within Telegram. The following image displays the first result. Since I was subscribed to the "DataLeak" room, I received a search result which allowed me to download the file immediately, without spending any money or time to earn credits. In April of 2024, it appears this room was either voluntarily or forcefully shut down. This is very common on Telegram. When it happens, move on to the next lead.



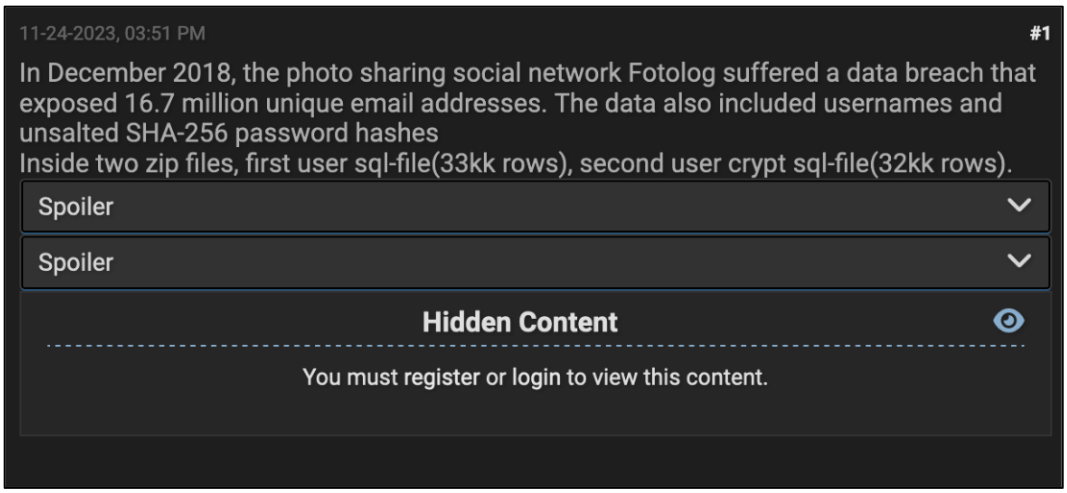
The next post on the BreachForums site was offering to sell a file called "+250 Million USA Huge Leak". The description of this data follows. Notice the total records size of 250,807,711.



I conducted a search of "250 million" on Telegram, and received the following result since I was a member of the "Open Data" channel. The text file contained a torrent link, which I opened within my browser, which launched Transmission, and began downloading the data. Notice that the total record count matches the previous image of someone trying to sell the data. This is another room which many readers have reported as missing, but some readers have found success searching "250807711.7z" "torrent" within Google. Please remember that these examples are to serve as overall search methods, and not to provide data which will be archived forever. The next time you learn about data of interest, begin the hunt on Telegram.



The next post on this page presented a file which contained the Fotolog breach data, as seen in the following image. A search of "Fotolog" on Telegram produced the result seen in the second image, offering this file for free.



We could do this all day. The lessons are to join as many breach data channels as you can find, and search Telegram whenever you see something interesting being sold. You may find enough channels to keep you busy for weeks at the following URL.

https://github.com/fastfire/deepdarkCTI/blob/main/telegram.md

One reader

SQL Files

Some old-fashioned Google queries might find more data breaches and leaks than one can manage. Let's conduct a few examples with SQL files. SQL, often pronounced "sequel", is an acronym for Structured Query Language. It is a computer language used in programming and designed for managing data held in a relational database management system. In other words, most SQL files are databases of some sort. Many of the most popular database breaches were originally released as SQL files. WordPress backups, web forum exports, and other online maintenance files are also stored in this format. Searching for public SQL files can reveal surprising results. Let's search Google for SQL files with the following operators.

`ext:sql`: This returns thousands of results. While some are files with the extension of ".sql", most of the results are web pages ending with ".sql", and are not helpful. Let's modify the search as follows.

`ext:sql "create table":` This returns 26,300 results which include the exact terms "create table". This is a standard statement within SQL files which specifies the names of tables within the database. This filters most of the website names from our search and displays only valid SQL files. Next, let's add to our search with the following modification.

`ext:sql "create table" "@gmail.com":` This returns 5,160 results. Each is an SQL database with at least one entry of "@gmail.com" inside. This indicates that active email addresses are within the files, which is indicative of a true breach or leak. The following search should reveal data worth analyzing.

`ext:sql "create table" "@gmail.com" "'password'":` This returns 3,060 results. Each is an SQL database with at least one entry of "gmail.com" inside and a table titled "password". Note the single quotes around password within double quotes. This tells Google to search specifically for 'password' and not the word alone. 3,060 results are overwhelming, and include a lot of test files stored on Github. Since many people use Gmail addresses as the administrator of test databases, we should add another email domain as follows.

```
ext:sql "create table" "gmail.com" "'password'"
"@yahoo.com" -site:github.com
```

This reveals 67 results. Each are SQL files which contain at least one Gmail address, one Yahoo address, and a table titled password. Furthermore, all results from Github are removed. Most of these results will load as text files within your browser. However, some will be quite large and may crash your browser while loading. The following is a modified excerpt from an actual result, which I found by searching "@gmail.com" within the browser text.

```
(62,'RichardWilson','admin','richard@redactedemail.com
','4d5e02c3f251286d8375040ea2b54e22','Administrator',0
,1,25,'2008-05-28 07:07:08','2009-04-02 13:08:07')
```

This is a very typical structure within SQL files. The following explains each piece.

`62,` (User ID)
`'RichardWilson',` (Name provided)
`'admin',` (Username)
`'richard@redactedemail.com',` (Email address)
`'4d5e02c3f251286d8375040ea2b54e22',` (Hashed password)
`'Administrator',` (Usertype)
`0,1,25,` (Variable internal codes)
`'2008-05-28 07:07:08',` (Registration date)
`'2009-04-02 13:08:07',` (Last visit date)

You could save the entire page as a .txt file (right-click > Save page as…) for your personal data archive. An alternative Google query for text files is as follows.

ext:txt "create table" "gmail.com" "'password'" "yahoo.com" -site:github.com

As I updated this section, I conducted a query of ".sql" in Telegram. Since I was subscribed to the "LEAKES AGGREGATOR" and "Leakbase Official" channels, I received hundreds of search results including recently breached SQL files. Most of these databases each possessed thousands of names, addresses, and often hashed passwords. You could spend weeks going through all of the data waiting to be plucked.

There may be a trove of sensitive information within these files, so use caution and always be responsible. Exercise good defense when browsing any of these sites or downloading any data. Trackers, viruses, and overall malicious software are always present in this environment. Using a macOS or Linux machine and a reputable VPN will provide serious protection from these threats. I search and store leaked and breached databases as a part of every investigation which I conduct.

I can say without hesitation that these strategies are more beneficial than any other online investigation technique of which I know. Some investigators within my circles possess several terabytes of this data from tens of thousands of breaches and leaks. Querying your own offline archive during your next investigation, and identifying unique data associated with your target, can be extremely rewarding. Again, I ask you to be responsible. Never use any credentials to access an account and never allow any data obtained to be further distributed. Use this public data, stolen by criminals, to investigate other criminals.

Hashes

Most websites store passwords in "hashed" form. This guards against the possibility that someone who gains unauthorized access to the database can retrieve the plain-text passwords of every user in the system. Hashing performs a one-way transformation on a password, turning the password into another string, called the hashed password. "One-way" means that it was practically impossible to go the other way and turn the hashed password back into the original password. This was true many years ago, but not so much today. There are several mathematically complex hashing algorithms that fulfill these needs. Some are very insecure and others are nearly impossible to crack. Let's look at a few examples.

Many older databases which have been breached possessed simple MD5 password hashes. The password of "password1234" is as follows as an MD5 hash. The Sha-1 hash of the same password, "password1234", is also as follows. Notice this is substantially longer, and a bit more secure. However, it will be quite easy for us to crack these passwords in just a moment. Below this example is the same password in Sha-256 and Sha-512 format. As you can see, these become increasingly complicated.

MD5: BDC87B9C894DA5168059E00EBFFB9077

SHA1: E6B6AFBD6D76BB5D2041542D7D2E3FAC5BB05593

SHA256:
B9C950640E1B3740E98ACB93E669C65766F6670DD1609BA91FF41052BA48C6F3

SHA512:
8C7C9D16278AC60A19776F204F3109B1C2FC782FF8B671F42426A85CF72B1021887DD9E4FEBE420DCD215BA499FF12E230DAF67AFFFDE8BF84BEFE867A8822C4

In regard to the various "older" breached databases which you are likely to find on the internet, you will most commonly see MD5 and SHA1 hashed passwords. Some of them possess a "salt". This is a small amount of data added to the hashing which makes cracking the password more difficult. If a breach does not possess the salt, the passwords are nearly impossible to crack. If the salt is present, it takes considerable additional resources in order to display the text password. The methods of "cracking" passwords exceed the scope of this book. Fortunately, we do not need the knowledge and computer horsepower to convert these hashes into valuable passwords. We simply need a third-party resource.

Hashes (hashes.org) attempted to reveal the plain-text of your submitted password hash. This was usually done in an effort to assist security professionals to evaluate the security provided by the relevant hash submitted. For us, it provided a new lead to follow. The database contained billions of cracked hashes available via web search and API. If I queried " BDC87B9C894DA5168059E00EBFFB9077" via the search page on the site, I was immediately presented with "password1234" as the password. Unfortunately, the service disappeared in late 2020. However, online archives exist. I consider the following technique to be advanced, and only suitable for those who have a need to reveal hashed passwords as part of their daily operations. The entire archive of hashes and passwords which previously existed on hashes.org is available as a torrent file. The following websites contain a "magnet" torrent link within them. Copying and pasting this link within your browser should launch your default torrent software and begin the 90 GB compressed download. Make sure you have room.

https://pastebin.com/pS5AQNV0
https://old.reddit.com/r/DataHoarder/comments/ohlcye/hashesorg_archives_of_all_cracked_hash_lists_up

Once complete, you should have a folder titled "HashesOrg Archive" with folders of "Hashlists" and "Leaks" inside it. These two folders contain thousands of compressed zip files. While you could issue commands via Terminal to decompress all files, I find it easier to simply select all files within a folder; right-click on them; and select to open with your desired decompression tool. Once you see the ".txt" versions of each file present, you might want to delete the original ".zip" files to free some space. Let's take a look inside the "Leaks" folder and open the file titled "20_casio-com_found_hash_algorithm_plain.txt". A partial excerpt follows.

MD5 00747af6279313863a0319070bdbfb80:168130
MD5 007be02e9bd7eb4402a15f377ad22e9e:zhangker46

The data tells us that this specific breach stored passwords in MD5 format. In this excerpt, we know that the MD5 hash of "00747af6279313863a0319070bdbfb80" reveals a password of "168130". We also know that a password of "zhangker46" was used within the Casio website at the time of the breach. Both of these pieces of data will be valuable to us. Let's conduct an investigation.

Your target has an email address of "badguy42@gmail.com". A query of this email address within the COMB data reveals a password of "verybad1234". However, this does not tell you which breach is associated with this address. Within Terminal, after navigating to the "HashesOrg Archive" folder, the following command is issued.

```
rg -aFiN verybad1234
```

The result appears similar to the following.

Leaks/713_forums-nodoubt-com_found_hash_algorithm_plain.txt.zip
MYSQL5 887b0eb63dbc543991567864efc0b05aad5a8ab2:verybad1234

We now have circumstantial evidence that a user on a web forum for the band No Doubt was using a password of "verybad1234" and the MYSQL5 hash of that password is "887b0eb63dbc543991567864efc0b05aad5a8ab2". Does this prove that "badguy42@gmail.com" was using this forum? No, but it is a solid lead. It could also be someone else using the same password. Since Hashes.org does not share the username or email address, we must continue the investigation with the OSINT methods previously explained. Is that email address associated with any conversations about the band? That would give me more confidence with the result. We should look at the typical way that one would use the Hashes.org data set. Assume that you have identified the following data within a breach, leak, or online website.

badguy42@gmail.com:14FDF540E39F0F154C8D0B3BD82ACE100B779DFA

Searching "14FDF540E39F0F154C8D0B3BD82ACE100B779DFA" through the hash identification website **TunnelsUp** (tunnelsup.com/hash-analyzer) reveals the following.

14FDF540E39F0F154C8D0B3BD82ACE100B779DFA - Hash Type: SHA1

We know this is a SHA1 hash which represents a password, and can execute the following.

```
rg -aFiN 14FDF540E39F0F154C8D0B3BD82ACE100B779DFA
```

The results appear below.

Leaks/706_forums-utorrent-com_found_hash_algorithm_plain.txt.zip
SHA1 14FDF540E39F0F154C8D0B3BD82ACE100B779DFA:stillverybad1234

Leaks/1182_prowrestlingfans-com_found_hash_algorithm_plain.txt.zip
SHA1 14FDF540E39F0F154C8D0B3BD82ACE100B779DFA:stillverybad1234

We now know that "badguy42@gmail.com" likely used a password of "stillverybad1234" at some point in time. We also know that someone using the password of "stillverybad1234" used that password on a uTorrent forum and a wrestling website, both of which suffered a breach. Are these all the same person? We cannot definitely conclude that. However, these are great leads. My next search would be the following.

```
rg -aFiN stillverybad1234
```

This may identify more data to be analyzed. However, this is all circumstantial. If a password is unique and complex, I have more confidence in the relationship to my suspect. If the password is "password1234" and appears on hundreds of sites, this has no value.

The **HashMob** (hashmob.net) community now offers a paid search service and free downloads of lists similar to those previously found at Hashes.org. Registration is not required to download "Found" lists from the "Official lists" menu. These files are very similar to the Hashes.org data, but updated often. I downloaded all 2,326 official (non-combo) hash lists with the following command.

```
wget --content-disposition -i
https://inteltechniques.com/data/hashlists.txt
```

WARNING: The result was 90 GB of data. A typical file was titled "330.3200.found", which identified the ID number of the breach (330) and the type of hash (3200-bcrypt). Each file contained numerous entries similar to the following.

00092f0f900824707b3508abc78aa6bb89f1e61bca81b0bcb01bb2efc6a1057:yanks2008

Everything to the left of the colon is the hash, and everything right of it is the password. Searching "330" (the ID of the breach) on the official HashMob website at **https://hashmob.net/api/v2/hashlist/official** reveals this is the hash list from a breach at Zildjian.ru.

Just like the tutorials within the Hashes.org section, this data allows us to search for either a hash or a password to identify if either is present within known breaches. It also allows us to query a hash in order to identify its plain-text password. After witnessing Hashes.org disappear, I prefer to possess my own archive of this free data. However, I also maintain a paid account with HashMob which allows real-time queries of hashes within their website, or via their API, which is much easier and immediately identifies passwords. If you need to query hashes often without downloading all the data, a paid account may be justified, and the results are updated daily. Next, let's take a closer look at some popular types of hashes.

MD5

Some older data breaches possess passwords hashed with MD5, an extremely insecure method. These hashes can be cracked very quickly. Below are two entries retrieved from various files.

174@gmail.com:482C811DA5D5B4BC6D497FFA98491E38
185@gmail.com:22F4182AAE2784FB3D1A432D44F07F46

Everything before the colon is the username or email address, and everything after is the MD5 hash. Searching these hashes in your Hashes.org data produces the following results. We now know that the first password is "password123" and the second is "reader12".

482c811da5d5b4bc6d497ffa98491e38:password123
22f4182aae2784fb3d1a432d44f07f46:reader12

MD5 + Salt (Vbulletin)

Some hashes will contain a "salt". This is usually a small piece of data after the hash, and both pieces are required to translate into plain-text. One of the most popular examples of this are the hundreds of Vbulletin online forums which have been infiltrated by various hacking groups. In the following examples, the salt is the final three characters after the last colon.

Aanas,menball@aol.com:9d9e3c372d054c0769bd93181240be36:tye
Traor,tranr@optusnet.com.au:9274583d060b3efb464115e65a8c1ead:vv#

Searching these hashes and salts through your Hashes.org data provides the following.

9d9e3c372d054c0769bd93181240be36:tye:eliza!%
9274583d060b3efb464115e65a8c1ead:vv#:runner

In the first example, you see the target password is "eliza!%". This proves that semi-complex passwords containing special characters are often dehashed and ready for conversion. This is where offline hashes stick out from the crowd. There are many online reverse-hashing tools, but most of them focus only on one format, such as MD5, and use minimal dictionaries to crack the hashes. Let's try one more demonstration.

SHA1

This format is slightly more secure than MD5, but still extremely easy to crack. The passwords obtained from the LinkedIn breach were all hashed in SHA1. Below are two examples.

174@gmail.com:403E35A2B0243D40400AF6BB358B5C546CDDD981
185@gmail.com:B1C4BBC4D7546529895CFABF8C1139CA7E486E18

The results from our offline hash collection follow. These two passwords are identical, but with different case. The first is all lowercase while the second contains uppercase. Notice that these create a completely different hash because of this. Keep this in mind when investigating passwords of your target.

403E35A2B0243D40400AF6BB358B5C546CDDD981:letmein!
B1C4BBC4D7546529895CFABF8C1139CA7E486E18:LetMeIn!

The golden days of data breaches from 2017 through 2019 are over. Sure, those databases are still floating around and very valuable, but recent breaches do not typically possess MD5 or SHA1 password hashes. We often see much more secure hashing algorithms such as BCRYPT or SHA512. Again, cracking these passwords with dedicated rigs exceeds the scope of this book. Searching "older" hashes can be time consuming, but the results can be extremely beneficial. It is quite a commitment to download hundreds of gigabytes of data for this purpose. Take time to think about your own needs.

Online Hash Search Resources

The following websites will convert MD5 and SHA1 hashes into passwords in some scenarios. The password must be within their limited system and should not be very complex. These are very limited, but may offer immediate data.

https://osint.sh/md5/
https://md5decrypt.net/en/Sha1/
https://www.md5online.org/md5-decrypt.html
https://www.dcode.fr/sha1-hash
https://md5decrypt.net/en/
https://md5hashing.net/hash/sha1

If you do not want to build, store, and maintain your own hash data set, I recommend **Name That Hash** and **Search That Hash** (github.com/HashPals/) over the online options. You can install it within your macOS or Linux machine (as long as you installed Pip, as previously explained) with the following steps.

```
sudo pip install -U search-that-hash
sudo pip install -U name-that-hash
```

Within Terminal, you can now execute the following to search a hash within multiple online converters.

```
sth --text "5f4dcc3b5aa765d61d8327deb882cf99"
```

The result appears as follows.

5f4dcc3b5aa765d61d8327deb882cf99
Text : password Type : MD5

You can also search for only the type of hash with a command of `nth --text "5f4dcc3b5aa765d61d8327deb882cf99"`. The result appears as follows.

Most Likely MD5, HC: 0 JtR: raw-md5 Summary: Used for Linux Shadow files.

We now know that the hash value is a MD5 representation of the password "password". This process is included within the script titled "Breaches/Leaks Tool" included in the OSINT VM. I launch this application daily. When it cannot identify the password, I rely on my Hashes.org data set. If all of this is simply too complicated, we can always rely on more online services, as explained soon.

Hopefully, you now have a basic understanding of breaches, passwords, hashes, database files, and how all of it can work together to be a part of your online investigations. You might have a copy of the COMB data set which includes over a billion email addresses and associated passwords, but that was created for convenience. You may have encountered some hashes online and can reveal passwords hidden beneath, but those will be based on specific target queries. While the data discussed so far is easily assessable, new breaches might not be so easy to find and digest. Let's start over with two breach examples from data recently stolen.

As stated previously, I am not allowed to provide direct links to any breach data. However, I am allowed to discuss ways in which some people use OSINT to find this data. Let's take another look at Torrent search engines. If you were to search "Guron" on many Torrent sites, including The Pirate Bay, you would likely be presented a Torrent file which contains 241 GB of data titled "MyCloud". Expanding the data should reveal 1,315 files, each of which are a breached database obtained between 2016 and 2021. This is far from everything breached during that time, but it may serve as another introduction into this type of content. Warning: the **compressed** content is 241 GB, so expect the decompressed to reach a terabyte. I plucked two small random examples from this data set in order to explain the data within.

DogForum.sk: The file titled "Dump_public_dogforum.sk_2021.01.13.rar" is a 2.3 MB compressed file which decompresses to a 12.5 MB SQL database dump. This site's database was allegedly stolen on January 13, 2021. Opening the SQL file within a text editor reveals a table titled "phpbb3_users". The following displays only the ten most interesting fields for my investigations within this table.

```
CREATE TABLE `phpbb3_users` (
  `user_id` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `user_ip` varchar(40) COLLATE utf8_bin NOT NULL DEFAULT '',
  `user_regdate` int(11) unsigned NOT NULL DEFAULT '0',
  `username` varchar(255) COLLATE utf8_bin NOT NULL DEFAULT '',
  `user_password` varchar(40) COLLATE utf8_bin NOT NULL DEFAULT '',
  `user_passchg` int(11) unsigned NOT NULL DEFAULT '0',
  `user_email` varchar(100) COLLATE utf8_bin NOT NULL DEFAULT '',
  `user_birthday` varchar(10) COLLATE utf8_bin NOT NULL DEFAULT '',
  `user_lastvisit` int(11) unsigned NOT NULL DEFAULT '0',
  `user_timezone` decimal(5,2) NOT NULL DEFAULT '0.00',
```

A redacted example of one user entry is below (there are over 17,000 users).

```
(8419,'178.40.120.170',1275503507,'Ode','$H$9ftKyo59RKktw7Idj2TGaVweMH2IT
G.',1350132160,'xxxxblackjack@gmail.com',' 0- 0-   0',1371625109,+3,)
```

We can now assume the following about this user based on this data.

8419: The user number for this user on this forum.
178.40.120.170: The IP address of the user during registration.
1275503507: The UNIX time the user registered (June 2, 2010 18:31:47 GMT).
Ode: The username for this user.

H9ftKyo59RKktw7Idj2TGaVweMH2ITG.: The hashed password.
1350132160: The UNIX time the user last changed the password (October 13, 2012 12:42:40 GMT).
xxxxblackjack@gmail.com: The email address of the user.
0- 0- 0: The date of birth of the user (none was provided).
1371625109: The UNIX time of the user's last visit (June 19, 2013 06:58:29 GMT).
+3: The user's time zone (+3 GMT - Middle East).

Please take a moment to digest this example. This entry is only one of 17,000 within this single stolen breach. The Torrent mentioned possesses over 1,000 breaches, and there are tens of thousands of known breaches out there. There is the potential to collect this type of information from billions of accounts. I hope you see the possibilities for your own investigations. Every day, we digest dozens of breaches into our own system, which I will discuss later, and we see this type of result with almost every query we conduct. Sometimes, we receive this type of data from within several breaches associated with our single target email address. It is quite powerful. Let's conduct another example.

EscortReviews.com: The file "Dump_public_escortreviews.com_2021.01.27.rar" is a 51.7 MB compressed file which decompresses to a 223.3 MB SQL database dump. This site's database was allegedly stolen on January 27, 2021, and was publicly reported a month later. Opening the SQL file within a text editor reveals a table titled "user". The following displays only the most interesting fields for my investigations within this table.

```
CREATE TABLE `user` (
  `userid` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `username` varchar(100) NOT NULL DEFAULT '',
  `password` char(32) NOT NULL DEFAULT '',
  `passworddate` date NOT NULL DEFAULT '0000-00-00',
  `email` char(100) NOT NULL DEFAULT '',
  `homepage` char(100) NOT NULL DEFAULT '',
  `joindate` int(10) unsigned NOT NULL DEFAULT '0',
  `lastvisit` int(10) unsigned NOT NULL DEFAULT '0',
  `birthday` char(10) NOT NULL DEFAULT '',
  `ipaddress` char(15) NOT NULL DEFAULT '',
```

A fictitious example of one user entry is below (there are over 470,000 users in this database).

```
(201,bobforfun,65184286F793B5E70ACA525E63DAE54F,2011-06-
12,bob_love@ymail.com,
bobforfun.com,1238309520,1307841316,06-02-1983,76.187.13.130,2)
```

We can now assume the following about this user based on this data.

201: The user number for this user on this forum.

Bobforfun: The username for this user on this forum.
65184286F793B5E70ACA525E63DAE54F: The hashed password for this account.
2011-06-12: The date the user last changed the password.
bob_love@ymail.com: The email address associated with this account.
bobforfun.com: The personal website associated with this account.
1238309520: The Unix time the user joined (March 29, 2009 at 06:52:00 GMT).
1307841316: The Unix time the user last accessed (June 12, 2011 at 01:15:16 GMT).
06-02-1983: The provided date of birth for this user.
76.187.13.130: The IP address of the user during registration.

This site was used heavily by escorts, pimps, and johns in order to promote prostitution. Anyone who conducts human trafficking investigations might benefit from this data on 470,000 registered members. The hash for this fictitious example is MD5. If you were to place that hash into our Search-That-Hash tool previously discussed, you would receive the following result.

65184286F793B5E70ACA525E63DAE54F
Most Likely
MD5, HC: 0 JtR: raw-md5 Summary: Used for Linux Shadow files.
Text : escorts

You would now know that this user was using a weak password of "escorts" for this site in 2011. These example files are fairly small, but there is still a lot of content inside them which is unnecessary. You could clean these files using the previous techniques, but that may be overkill for your needs. Since I collect terabytes of this data, I try to minimize every file as much as possible. Reconsider the "DogForum.sk" file from earlier. The following commands would extract only the columns we desire and remove the unnecessary tabs.

```
cut -d, -f1,6,7,8,10,11,13,15,16,29 dogforum.sk.sql > DogForum.txt
sed -i ''s/(ctrl-v-tab)//gI" DogForum-Cleaned.txt
```

This action shrank the 12.5 MB file to 2.4 MB and compressed each line to fit nicely within Terminal. That may not seem like much, but it can make quite a difference when dealing with thousands of files.

Archive.org Breach Data

Archive.org possesses a surprising number of data breach dumps. As one example, conduct a search on their site for "nulled.io" and click the "nulled.io_database_dump" page. It should present a download option for the entire 10 GB database from the Nulled hacking forum, as uploaded by "ntgrg". If you were to click on this profile, you would likely find other similar uploads. You could spend weeks on Archive.org identifying breach data.

Cit0day Collection

Next, let's take a look at an enormous data set and discuss the challenges of ingesting such content. This section will combine several techniques and commands which I have discussed throughout the previous chapters. In 2020, a criminal marketplace offering breached databases called Cit0day.in had allegedly been shut down by various government agencies. The same day the site displayed a seizure notification, the 23,618 hacked databases (containing 226 million credentials) from the site were provided for free download via a file sharing site. The government takedown of Cit0day was never proven, and most researchers speculate the seizure notice on the website was fake. However, the leaked data was real. Today, these files are still abundant on the internet for anyone to obtain.

If you were to search "cit0day", with a zero (0) instead of an "o", within the main Telegram search field, you would likely identify a channel called "cit0day collection in tg". Visiting this room should display thousands of these compressed breach data files within the channel. This room allows download of each of these individual hacked databases, or you can download the entire collection within three compressed files titled "Full Clouds Cit0day_2" near the end of the list. These files consisted of 9 GB of compressed data, which expanded to a single 9.1 GB compressed 7z file, which decompressed a folder titled "Cit0day_RF" containing 23,564 folders. From there, each folder possessed multiple files associated with a specific breached victim company. The data was 30 GB in size and consisted of 72,780 files. What a mess. Let's work through it together.

Some folders are easy. If you were to look at the contents of the folder titled "agekuda.net {8.882} [NOHASH] (Shopping)", you would see that only one file exists, titled "agekuda.net {8.882} [NOHASH].txt". This file possesses all 8,882 users of this site, and the contents appear similar to the following.

```
mge.com@ezweb.ne.jp:sundial
qbq@ezweb.ne.jp:apo11
```

All of the credentials within the file are plain-text passwords and there are no hashed passwords. This file is ready to be added to your collection. However, let's take a look at the folder titled "agenziaporto.it {6.405} [HASH+NOHASH] (Shopping)". It possesses the following five files.

```
NotFound.txt
Rejected.txt
Result.txt
Result(HEX).txt
www.agenziaporto.it {6.405} [HASH].txt
```

The "NotFound.txt" file contains all of the usernames and hashed passwords for this breach which were not dehashed into plain-text passwords. Everything within this file is redundant to the last "[HASH]" file, and this file is not needed. The "Rejected.txt" file is also redundant. The "Result.txt" file contains all of the hashed passwords from

the final "[HASH]" file which were turned into plain-text passwords. This file has value, but the "Result(HEX).txt" file is not necessary for our needs. The final file titled "www.agenziaporto.it {6.405} [HASH].txt" contains the most important content. It contains the 6,404 usernames and hashed passwords from this breach.

If desired, you could simply copy these 23,618 folders into a directory called "Databases" within your SSD and query the data whenever needed. There is no shame in that, and you could move on to more data. However, I cannot stomach this. The amount of unnecessary data and redundant entries would bother me. It may not seem like much now, but when you have 200,000 breached databases which take several minutes to query, you may wish you had cleaned the data before archiving it. The first step I took was to open Terminal and navigate to the folder which stored all of this data. For me, the command was as follows.

```
cd ~/Downloads/Telegram\ Desktop/Cit0day_RF/Data/
```

Once there, I removed all files which included "Rejected.txt", "NotFound.txt", "Result(HEX).txt", "*not found*.txt", or "final" in the name. This eliminated much of the redundant data.

```
find . -iname 'Rejected.txt' -delete
find . -iname 'NotFound.txt' -delete
find . -iname 'Result(HEX).txt' -delete
find . -iname '*not found*.txt' -delete
find . -iname '*final*.txt' -delete
```

Below is the breakdown of the commands.

find	This tells Terminal to find something.
.	This tells Terminal to search everything below the current folder.
-iname	This tells Terminal to search for content without case sensitivity.
'*final*.txt'	This tells Terminal to find any text file with "final" in the name.
-delete	This tells Terminal to delete whatever matches our command.

Next, I wanted to combine all of the "decrypted", "result", and "no hash" files into one large file. These all contain redundant usernames which are already in the main file for each breach, but they also include plain-text passwords which could be valuable. The following commands combined all of the data and then deleted the unnecessary files. Notice that I removed the "i" from the "no hash" commands because I do not want to delete uppercase entries containing "NO HASH".

```
find . -iname '*result*' -exec cat {} \; > Pass1.txt
find . -iname '*decrypt*' -exec cat {} \; > Pass2.txt
find . -name '*no hash*' -exec cat {} \; > Pass3.txt
find . -iname '*result*' -delete
find . -iname '*decrypt*' -delete
find . -name '*no hash*' -delete
```

```
cat Pass1.txt Pass2.txt Pass3.txt > Cit0DayPass.txt
rm Pass1.txt Pass2.txt Pass3.txt
```

You would now have a file called Cit0dayPass.txt which contains several gigabytes of usernames and plain-text passwords from this collection. If desired, you could use the Sort command previously explained to remove duplicates from this file, as there are many. Since these usernames are already within the files we will deal with next, you do not need to know of which breach these entries originated. I will query this file in a moment. Finally, each folder should now be left with only one file which should be similar to one of the following naming structures.

3bbwifi.com {1.687} [HASH] [NOHASH].txt
agendadelasmujeres.com.ar {25.541} [HASH].txt

We could just save these to our Databases folder, but I want them cleaner. The following command renames every file to remove the unnecessary information within parentheses and brackets. We will need the Rename application, which can be installed with the commands.

macOS: `brew install rename`
Linux: `sudo apt install rename`

Once you know you have Rename installed, consider the following long command.

```
find . -type f -execdir rename
's/\{[^\}]+\}//g;s/<[^<]+>//g;s/\[[^\]]+\]//g;s/\([^\)
]+\)//g' {} \;
```

This basically finds all files in all subdirectories and renames them to remove any parentheses and brackets, and all data between them. I use this command often when I download thousands of files with long unnecessary names. The previous files should now appear as follows.

3bbwifi.com .txt
agendadelasmujeres.com.ar .txt

This is much nicer, and I can now move all of these files into a single folder with the following commands. In this scenario, I am moving them to a folder called Databases within my SSD. If you named your SSD the same as mine, this should work for you too.

```
mkdir /Volumes/DATA/Databases/
find . -type f -name '*.txt' -execdir mv {}
/Volumes/DATA/Databases/  \;
```

The result should be all 22,762 text files within the Databases folder on your SSD, with a total size of 20.37 GB (down from 30 GB). This also moved the Cit0daypass.txt

file which we previously created. You may notice that many of the files possess a trailing space in the file name. If this bothers you, navigate to your new Databases folder and execute the following.

```
rename "s/ //g" *
```

Finally, let's conduct a query of a fictitious target.

```
rg -aFiN test9876@gmail.com
```

The results would appear as follows.

friuligol.it.txt
test9876@gmail.com:ef5381e56a457e62bf22976b5b4f4f8f

Cit0DayPass.txt
test9876@gmail.com:mypassword

This displays the presence of our target email address in the original breach (friuligol.it) with a hashed password, and the dehashed plain-text password (mypassword) from the file we previously created from the "results" and "decrypted" files. You would now possess this data in a format which is smaller, easier to digest, and faster to query. You can delete the entire "Cit0day_RF" folder once you know you have the necessary data on your SSD. If desired, you can conduct the following to see the total number of lines within all files combined.

```
find . -name '*.txt' | xargs wc -l
```

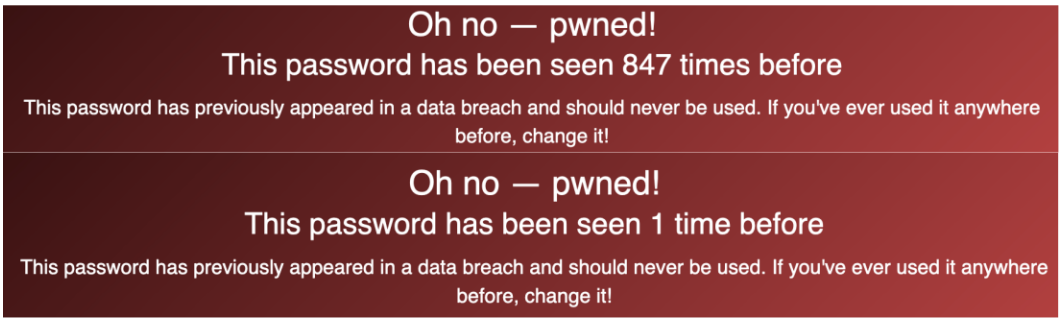
The result appears as follows.

50727432 total

We now know that the original report of 226 million credentials was likely based on all of that redundant data. A total of 50 million victims is more accurate. Journalists will often exaggerate the truth in order to get more clicks. Trust your own analysis. This is still not bad for a single data breach collection, which can be downloaded and cleaned in an hour. The following image displays a very small portion of this collection.

Password Commonality

If you encounter a password for your target, you may want to know if it was unique. As an example, I would assume my previous example of the password "escorts" is fairly common. I can confirm this through the Have I Been Pwned password site at haveibeenpwned.com/Passwords. The following figures display the result identifying 847 breaches containing that password (top). However, what if your target's password was "escorts!"? Surpassingly, there is only one reported breach which had this password, as seen in the second image.



Online Breach Search Resources

The following provides direct URL submission options for several breach data search websites. This will be vital for our Data Tool mentioned at the end of this chapter.

Email Address (test@test.com)

https://dehashed.com/search?query=test@test.com
https://intelx.io/?s=test@test.com
https://psbdmp.ws/api/search/test@test.com
https://check.cybernews.com/chk/?lang=en_US&e=test@test.com
https://portal.spycloud.com/endpoint/enriched-stats/test@test.com
https://cavalier.hudsonrock.com/api/json/v2/preview/search-by-login/osint-tools?email=test@test.com
https://api.proxynova.com/comb?highlight=1&query=test@test.com

Username (test)

https://dehashed.com/search?query=test
https://psbdmp.ws/api/search/test
https://api.proxynova.com/comb?highlight=1&query=test

Domain (inteltechniques.com)

https://dehashed.com/search?query=inteltechniques.com
https://psbdmp.ws/api/search/inteltechniques.com
https://intelx.io/?s=inteltechniques.com
https://www.hudsonrock.com/search?domain=inteltechniques.com

Telephone (6185551212)

https://dehashed.com/search?query=6185551212
https://psbdmp.ws/api/search/6185551212

IP Address (1.1.1.1)

https://dehashed.com/search?query=1.1.1.1
https://psbdmp.ws/api/search/1.1.1.1
https://intelx.io/?s=1.1.1.1

Name (Michael Bazzell)

https://dehashed.com/search?query=michael Bazzell
https://psbdmp.ws/api/search/Michael%20Bazzell

Password (password1234)

https://dehashed.com/search?query=password1234
https://psbdmp.ws/api/search/password1234
https://www.google.com/search?q=password1234
https://api.proxynova.com/comb?highlight=1&query=password1234

Hash (BDC87B9C894DA5168059E00EBFFB9077)

https://hash.ziggi.org/api/dehash.get?hash=BDC87B9C894DA5168059E00EBFFB9077&include_external_db
https://decrypt.tools/client-server/decrypt?type=md5&string=BDC87B9C894DA5168059E00EBFFB9077
https://md5.gromweb.com/?md5=BDC87B9C894DA5168059E00EBFFB9077
https://www.nitrxgen.net/md5db/BDC87B9C894DA5168059E00EBFFB9077
https://dehashed.com/search?query=BDC87B9C894DA5168059E00EBFFB9077
https://www.google.com/search?q=BDC87B9C894DA5168059E00EBFFB9077

Miscellaneous Sites

The following websites do not allow submission via URL, but a manual search may be beneficial. Please remember that all of these sites come and go quickly. By the time you read this, some of these services may have shut down. It is equally possible that new resources are waiting for your discovery.

HIBP: https://haveibeenpwned.com
LeakPeek: (leakpeek.com)
Beach Directory: (breachdirectory.org)

H8Mail (github.com/khast3x/h8mail)

H8Mail attempts to combine many of the breach services we have explored into one utility. It should never take the place of a full manual review, but the embedded automation can be beneficial to an investigation. Conduct the following within Terminal.

```
sudo pip install -U h8mail
cd ~/Downloads
h8mail -g
sed -i 's/\;leak\-lookup\_pub/leak\-lookup\_pub/gI'
h8mail_config.ini
```

Launching the following from the Downloads directory will produce minimal, if any, results.

```
h8mail -t bill@microsoft.com
```

Providing API keys from services such as Snusbase, Leak-Lookup, HaveIBeenPwned, Emailrep, Dehashed, and Hunterio will provide MANY more results, but these services can be quite expensive. If you rely on breach data every day; can afford premium services; and do not want to collect your own breach data; there may be value for you within this option. After you have obtained API keys from your desired services, open the Files application, enter the Downloads folder, and double-click the file named "h8mail_config.ini". You should see text similar to the following partial example. Add your API keys within the appropriate lines, similar to the entry for "leak-lookup pub", and remove any semicolons within lines you want to be used. If a semicolon is at the beginning of a line, that option is ignored. At the minimum, make sure the semicolon is removed from the "leak-lookup pub" line and execute another query. You should see new results. If you conducted the previous "sed" command, this should already be configured for you.

```
;hibp =
leak-lookup_pub = 1bf94ff907f68d511de9a610a6ff9263
;leak-lookup_priv =
;emailrep =
```

After this new configuration file modification, I executed a search for test@email.com. The result was a file which contained 172 results. The following partial view confirms that this address exists within breaches from TicketFly, TrueFire, and Tumblr. Eliminating this modification returned no results.

```
test@email.com,LEAKLOOKUP_PUB,ticketfly.com
test@email.com,LEAKLOOKUP_PUB,truefire.com
test@email.com,LEAKLOOKUP_PUB,tumblr.com
```

Sample Investigation

We have covered a lot so far within this chapter. Let's pause and conduct an investigation using this data. Assume that you possess the databases mentioned previously, especially "COMB". Your investigation identifies a suspect with an email address of johndoe1287@gmail.com. This appears to be a burner email account, as you find no evidence of it anywhere online. It was used to harass your client and likely created only for devious activity. Our first step is to query the address through all the databases you have acquired. Assume these are all stored on your external SSD, which is plugged into your host computer. Let's take each step slowly.

Open Terminal, type cd, then space, and drag and drop the external drive from your Files (Finder) application into Terminal. This makes sure you are in the correct path. My command appears similar to the following.

```
cd '/media/osint/DATA/'
```

I can now conduct a query within Terminal against all of my collected data. The following searches our target email address. Each query could take several minutes if you possess a lot of data and a slow drive.

```
rg -aFiN johndoe1287@gmail.com
```

This results in one entry as follows.

johndoe1287@gmail.com:H8teful0ne45

We now know that he used the password of H8teful0ne45 on a site. We should next conduct a search of that password to see if it is used anywhere else. The following query is appropriate.

```
rg -aFiN H8teful0ne45
```

This returned the following results.

johndoe1287@gmail.com:H8teful0ne45
johndoe@gmail.com:H8teful0ne45
johndoe1287@hotmail.com:H8teful0ne45

These addresses are likely controlled by our target since the passwords are the same and the addresses are similar. We now have new search options. However, this search only queries for this exact text password term. If you possess a database which has not been dehashed, your target password could be present within an MD5, SHA1, or other hash. Therefore, let's convert this password into the most commonly used hashes with the following websites, displaying the output below each.

https://passwordsgenerator.net/md5-hash-generator/
9EF0EC63E2E52320CB20E345DCBA8112

https://passwordsgenerator.net/sha1-hash-generator/
D15FB15C1BC88F4B7932FD29918D1E9E9BBE7CA5

https://passwordsgenerator.net/sha256-hash-generator/
37A790A268B9FE62B424BABFC3BCAB0646BFB24B93EC1619AAE7289E0D7086DB

We can now submit each of these throughout all of our data with the following three commands.

```
rg -aFiN 9EF0EC63E2E52320CB20E345DCBA8112
rg -aFiN D15FB15C1BC88F4B7932FD29918D1E9E9BBE7CA5
rg -aFiN 37A790A268B9FE62B424BABFC3BCAB0646BFB24B93EC1619AAE7289E0D7086DB
```

The first query returned the following result.

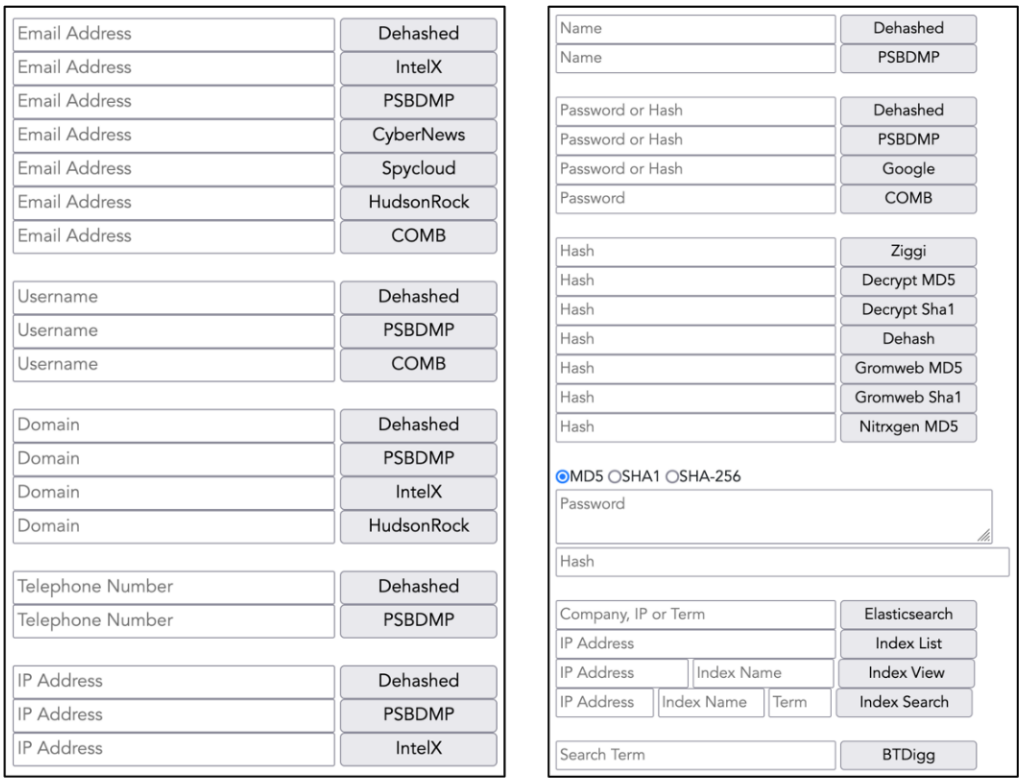
Leaks/1183_houstonastros-com_found_hash_algorithm_plain.txt.zip
SHA1 D15FB15C1BC88F4B7932FD29918D1E9E9BBE7CA5:H8teful0ne45

This tells us that a user with a password of "H8teful0ne45" was present on a breach about the Houston Astros. Is this the same person? It could be. It could also be a coincidence. The more unique a password is, the more confidence I have that it is the same individual. This definitely warrants further investigation. I might next try to locate the original breach data, which would likely include any email addresses associated with that password hash. All of these steps are designed to lead us to the next step. All of these results give me more confidence that these accounts are owned by the same person. The variant of the "hateful" password and presence of "johndoe" within the original email address and the new password convinces me we are on the right track. I would now target this new email address and replicate the searches mentioned within previous chapters. We should also check the online breach resources previously explained.

As previously stated, your biggest frustration may be the speed of each query. There is a lot to digest in this chapter. I want to state again that breach data has been the absolute biggest aid in my online investigations. I rely on it way more than Facebook, Twitter, and Instagram combined. In a future chapter, I will offer additional storage and query techniques which should assist with immediate access to your data.

IntelTechniques Breaches & Leaks Tool

This search tool combines most of the online search options mentioned throughout the chapter. The breach data resources are split into categories based on the target data (email, username, etc.). The last feature allows entry of found passwords and immediately generates an MD5, SHA1, and SHA256 hash for further research. The following displays the site at **https://inteltechniques.com/tools/Breaches.html**.



CHAPTER SEVEN
STEALER LOGS

I am absolutely fascinated by stealer log data. While I have always prioritized breach data as a vital part of my investigations, stealer log data presents a whole new world. This chapter offers two unique ways of acquiring stealer logs. The first is the manual process which collects more data, but requires a lot of effort. The second obtains only the most beneficial details with minimum effort, but misses some data you might find valuable. First, let's revisit the basics. This is my favorite chapter.

Stealer logs are created once a malicious virus has been installed to a computer (typically Windows). The victim may be tricked into installing a program after visiting a malicious website, or the virus could be included within an unauthorized application, such as pirated software. The virus sniffs through the computer to identify, extract, and collect any valuable data. These "logs" are then transferred from the host machine and distributed within shady online networks.

The most common stealer logs we would find in previous years were labeled as Raccoon Stealer, Redline Stealer, and Vidar Stealer. Today, every stealer log group rebrands the data they steal from other groups in order to present it as their own. There is no honor amongst thieves. The groups which offer data stolen via these programs are limitless. A new group pops up every week. Criminal marketplaces trade this data as a commodity. They use the stolen data to unlawfully access online accounts, steal cryptocurrency, make unauthorized purchases, and wreak havoc on innocent people's digital lives. Our systems ingest over one million logs every day which are being shared online.

Let's take a look at some real data. In the following example, I extracted a random log which was generated by the Redline Stealer and uploaded to a criminal marketplace. I redacted much of the content. First, let's understand the file structure. The following is a tree of a single log file for one victim.

```
└── US[xx4909xC4Ex3C4x008x57AxD60BBCAF6] [2022-06-19T08_49_35]
    ├── -----000.txt
    ├── 00000000000.jpg
    ├── Autofills
    │   ├── Google_[Chrome]_Default.txt
    ├── Cookies
    │   ├── Google_[Chrome]_Default Network.txt
    │   └──Microsoft_[Edge]_Default Network.txt
    ├── DomainDetects.txt
    ├── ImportantAutofills.txt
    ├── InstalledBrowsers.txt
    ├── InstalledSoftware.txt
    ├── Passwords.txt
    ├── Screenshot.jpg
    └── UserInformation.txt
```

Now, let's walk through each folder and file.

```
└── US[xx4909xC4Ex3C4x008x57AxD60BBCAF6] [2022-06-19T08_49_35]
```

This top folder presents a two-letter abbreviation of the country of the victim (United States) followed by a unique Hardware Identifier (HWID) and the date and time of capture. The HWID is a security measure used by Microsoft upon the activation of Windows. This unique HWID is generated when the operating system is first installed. This will be vital in a moment. The date and time allow us to know the likely accuracy of the data.

```
├── -----000.txt
```

This is an information file about the thief. It often includes generic contact information, pricing for stolen data, and online communities associated with the product.

```
├── 00000000000.jpg
```

This is the logo of the Redline product.

```
├── Autofills
│   ├── Google_[Chrome]_Default.txt
```

This is where things get interesting. Your browser has likely asked you if you would like to store information which was entered into an online form. This could include your name, address, email, or other unique detail which gets entered into websites often. If you allow your browser to store this data, stealer logs easily collect it into their systems. The following is partial (redacted) data extracted from an actual victim. The original logs clearly displayed all data.

```
Name: email
Value: loxxx@gmail.com
===============
Name: username
Value: skxxxx22
===============
Name: lastname
Value: Loxxx
===============
Name: first-name-field
Value: Alixxx
===============
Name: address
Value: 20xx xxxx Lane
===============
```

Name: city
Value: Moxxx
==========
Name: phone
Value: 209-xxx-xxx
==========
Name: dob
Value: 10/xx/20xx
==========
Name: VIN
Value: kmhtcxxxxxxxxx
==========
Name: keyword
Value: 20xx Subaru xxxx
==========
Name: card-name
Value: Alixxx Loxxx
==========
Name: expiration-date
Value: 0x/2x
==========

I now have the name, DOB, home address, email, cell, vehicle, and partial credit card details of the victim.

```
├── Cookies
│   ├── Google_[Chrome]_Default Network.txt
│   └──Microsoft_[Edge]_Default Network.txt
```

Your browser stores temporary internet files about your credentialed sessions. I may not have your password to your email account, but possessing the cookies from your browser could allow me to steal your credentialed session and replicate access to your account. The following details could be beneficial to a criminal (obviously redacted and abbreviated).

.paypal.com TRUE / FALSE 196941xxx cookie_check yes
.paypal.com TRUE / FALSE 168533xxx cookie_prefs
P%3D1%2CF%3D1%2Ctype%3Dimplicit
.paypal.com TRUE / FALSE 196941xxx d_id
9ebfcxxxe8545ae9a39xxx2d228xxx116537xxx
.paypal.com TRUE / FALSE 1969xxxG8
KN2xxx0aJZzhbL_R4HkiO_kb5_yMTkUPrF-Ml6xxx
.paypal.com TRUE / FALSE 1685334379 X-PP-ADxxxYsNaAuNxxxHBuQ9dI
.paypal.com TRUE / FALSE 1716870381 _ga GA1.2.137xxx.165379xxx
.paypal.com TRUE / FALSE 1716912230 login_email lopxxxgmail.com
.paypal.com TRUE / FALSE 1969417578 rmuc KhGxxxmVv_x1Oo9gQ7axxxUk

├── DomainDetects.txt

This file offers immediate access to the priority domains which exist in the overall record. This allows criminals to quickly identify logs of interest.

PDD: [Amazon] amazon.com (2), [Games] steamcommunity.com (2)
CDD: [PayPal] paypal.com (40), [Amazon] amazon.com (14), [Games] battle.net (11), [Games] epicgames.com (1), [Games] steamcommunity.com (8)

├── ImportantAutofills.txt

This file parses data from the stored form fields which will be most beneficial to a criminal. The last two lines of my example suspect appear as follows. I also see full credit card details presented here often.

dob: 06/xx/xx
ssn: 248xx2xxx

├── InstalledBrowsers.txt

This file identifies all installed browsers and versions.

1) Name: Google Chrome, Path: C:\Program Files (x86)\Google\Chrome\Application\chrome.exe, Version: 102.0.5005.115
2) Name: Internet Explorer, Path: C:\Program Files\Internet Explorer\iexplore.exe, Version: 11.00.22000.1 (WinBuild.160101.0800)
3) Name: Microsoft Edge, Path: C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe, Version: 102.0.1245.44

├── InstalledSoftware.txt

This file presents all applications installed within the machine. While this may not be extremely valuable to a criminal, it is gold to an investigator. If my target possesses a stealer program, I get to monitor a lot of details about the person's computer usage. In the following example, I would know which VPN my target uses, hardware details, and preferred games. That could lead to quite a social engineering attack.

1) Adobe Acrobat Reader DC [22.001.20117]
2) Adobe Creative Cloud [5.5.0.617]
3) Adobe Genuine Service [7.7.0.35]
4) Adobe Photoshop 2021 [22.1.1.138]
5) Adobe Refresh Manager [1.8.0]
6) Epic Games Launcher [1.1.279.0]
7) ExpressVPN [7.7.12.4]
8) ExpressVPN [7.7.12.4]
9) Google Chrome [102.0.5005.115]
10) HP Audio Switch [1.0.179.0]
11) HP Connection Optimizer [2.0.17.0]

12) HP PC Hardware Diagnostics UEFI [7.6.2.0]
13) Intel(R) Chipset Device Software [10.1.18295.8201]
14) Launcher Prerequisites (x64) [1.0.0.0]
15) LOOT version 0.16.0 [0.16.0]
16) McAfee LiveSafe [16.0 R27]
38) Minecraft Launcher [1.0.0.0]
39) NVIDIA Texture Tools Exporter for Adobe Photoshop [2020.1.3]
40) Razer Synapse [3.7.0531.052416]
41) Red Dead Redemption 2 [1.0.1436.31]
42) Rockstar Games Launcher [1.0.59.842]
43) Rockstar Games Social Club [2.1.3.7]
44) Steam [2.10.91.91]
45) UE4 Prerequisites (x64) [1.0.14.0]

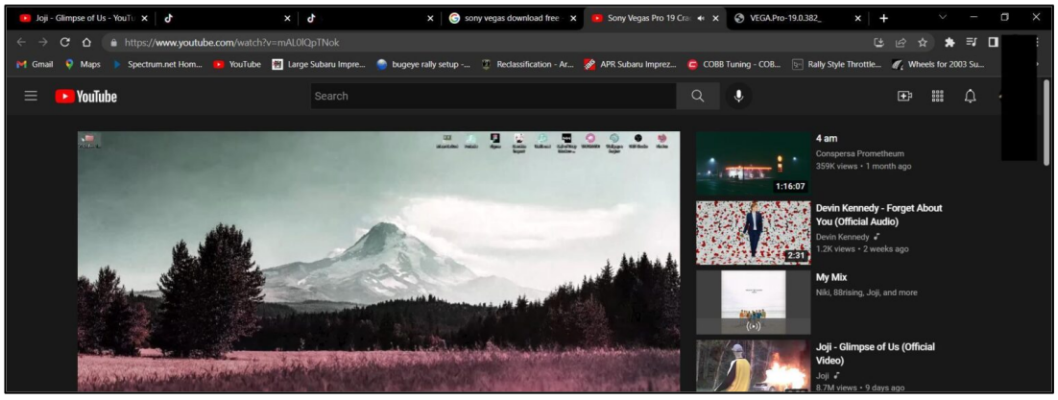
├── Passwords.txt

This file presents all of the passwords stored within all browsers. This is why it is so important to only use reputable password managers, and never the native browser password storage option. Below is one of 56 examples for this victim, redacted. The original file displays all passwords in plain-text.

URL: https://www.amazon.com/ap/signin
Username: xxxxxxx
Password: xxxxxxx

├── Screenshot.jpg

This is one of the most interesting pieces. It is a screen capture of the victim's machine at the time of infection. The following figure is an actual example (slightly redacted) which identifies the person's video interests, TikTok favorites, toolbar shortcuts, Google avatar (redacted), and an overall state of the computer at the time. This is quite invasive and can be a great lead in the investigation.



└── UserInformation.txt

The last file displays general details about the system, including the victim's IP address, hardware, location, and date. An example is below.

Build ID: REDLINEVIP
IP: 192.xx.xx.xx
FileLocation: C:\Windows\Microsoft.NET\Framework\v4.0.30319\InstallUtil.exe
UserName: xxx
Country: US
Zip Code: xxx
Location: xxx, Texas
HWID: 72xxxxxxxxxxxxxxCAF6
Current Language: English (United States)
ScreenSize: {Width=1920, Height=1080}
TimeZone: (UTC-07:00) Mountain Time (US & Canada)
Operation System: Windows 10 Home x64
UAC: AllowAll
Process Elevation: False
Log date: 19.06.2022 8:49:35
Name: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz, 6 Cores
Name: NVIDIA GeForce RTX 2060, 4293918720 bytes
Name: Total of RAM, 12126.75 MB or 12715814912 bytes

You may be wondering why I get so excited about these logs when they expose sensitive details about innocent victims. I have two main reasons. First, it helps us defend our clients. We have had over a dozen clients who unknowingly installed a stealer virus. Our systems caught the infection within a few days and allowed us to contact the client to make notification. We have helped numerous organizations which had unknown infections within their network. Some of our staff calls this our "Pre-Crime Unit".

Next, it could be a priceless investigation tool. I have had numerous suspects within my own investigations who were victims of stealer logs. When this happens, I can see all of their email addresses, usernames, passwords, and computer information. This has revealed the true owner of many alias names and other deceitful tactics used by the suspect. If my target is the victim of stealer logs, my entire investigation is about to be wrapped up quickly. If I have the HWID, I can search through our troves of logger data to identify even more vital info about the suspect.

Because of this, we aggressively collect stealer log data every day. Some days we ingest over a terabyte of this data. The following Google queries might present interesting information, but most results lead to shady criminal marketplaces which require an account to see download links. Use extreme caution here, and I will present better options in a moment.

"stealer logs" "download"
"stealer logs" "Redline"

It is now time to dive into some very shady criminal communities. Ensure that your organization's policies allow this behavior; never publish or disseminate anything you find; understand any local, state, or federal laws which could disallow these actions; and always focus on the good usage of this data. Never attempt to log in to anyone's account and make sure your computer is well protected from online threats. The following is presented as an educational demonstration, and I take no liability for your own actions taken.

Full Stealer Logs via Telegram

Telegram is a cross-platform and cloud-based instant messaging service which provides group chat, optional end-to-end encrypted chats, VOIP calling, file sharing, and several other features. We installed it within a previous chapter and you may have already obtained some data from it. It has been scrutinized as a tool used by drug dealers, burglars, terrorists, child abusers, and many fringe groups. The software and services were developed, and are currently maintained, by a Russian company. I would never install this software on any personal or OSINT investigations machine. I only execute it on my dedicated leaks, breaches, logs, and ransomware desktop. Of any technique in this book, the following tutorial is the most likely to infect a machine with a virus. You have been warned.

However, I believe Telegram is quite possibly the best resource for Stealer Logs. Within seconds, you have access to terabytes of data without the need for links to third-party download providers which enforce slow download speeds for free accounts. We can download most of the data directly from Telegram servers at the full bandwidth of our internet connection. Let's start with a demonstration.

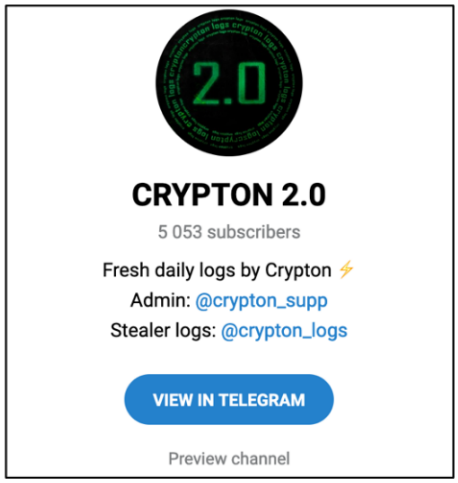
There are dozens of Telegram channels which offer free stealer logs in order to promote their paid services. I never recommend paying for any type of stolen data, but I will eagerly download and analyze any content they are sharing via their promotional channels. CRYPTON 2.0 is one of these services. If you were to search "CRYPTON 2.0" in Google, you will likely only see a few BreachForums posts offering logs, which have since been removed. However, if you were to navigate directly to the "CRYPTON 2.0" Telegram channel, you might find an overwhelming amount of free stealer logs. Next, we need to understand Telegram URLs. Telegram links might appear as the following fictious example, **which is not me**.

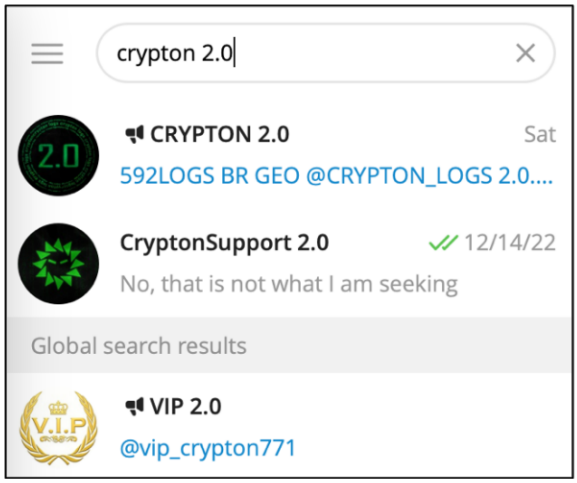
https://t.me/inteltechniques

If you navigate to a URL like this, your browser will likely prompt you to either choose an application or cancel the request to open an external program. If you have Telegram installed, you can select it and allow your browser to forward any channel links directly to Telegram Desktop. If you do not have Telegram installed, many channels will allow you to see a preview of the content. The URL for this browser-based preview would appear as follows.

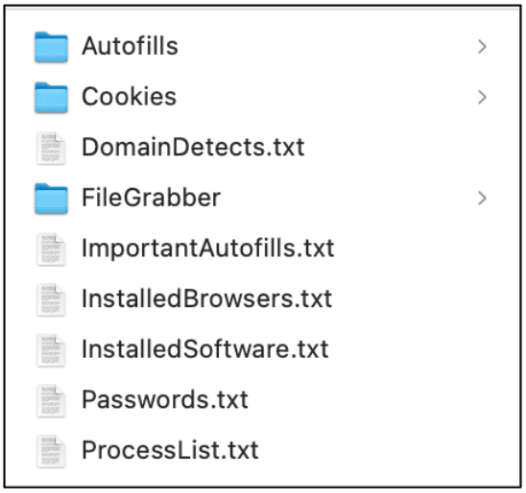
https://t.me/s/inteltechniques

In these two examples, I had my name (inteltechniques) in the URL. You would replace inteltechniques with whatever channel you are seeking (channels start with "@"). However, searching a room name within the Telegram application is always the easiest route. Remember, I cannot provide any direct hyperlinks to any stolen data, but you should be able to adapt my following demonstrations in order to access a room such as "CRYPTON 2.0". If you were to find this channel through a URL, you would see a page within your browser similar to the following image to the left. Searching within Telegram should appear as the following image to the right.





Clicking the "View In Telegram" option (URL) or room name (search result) should open this channel within Telegram and give you the opportunity to join. Notice this channel is promoting purchase of access to their logs collection, but also offering compressed samples for free. The following figure (left) displays how this would appear. In this image, clicking the downward arrow would download the 677 MB compressed archive of logs. Extracting the compressed file should result in a new folder which contains several (possibly thousands) of new folders. Each of these folders will possess a folder structure similar to that seen in the image to the right.



The "Autofills", "Cookies", "Passwords", and "Browsers" files contain the information as previously explained. The "FileGrabber" folder contains all documents stored within the default "Documents" and "Desktop" directories of the victim. These are typically full of Microsoft Word documents and PDFs. In this one sample file, we now have the passwords, documents, and other sensitive details of 1,101 victims. While this is scary and awful, it is also powerful as an investigative tool.

There are always dozens of active stealer logs channels, but you might prefer to only visit a room which attempts to combine free logs into one place. A channel called "Moon Cloud | Free Logs", which has a username of "Moon_Log" could be appropriate for you. At the time of this writing, it offered downloads of logs from DaisyCloud, LulzsecCloudLogs, Tiny Cloud, and dozens of others. There is easily over a terabyte of compressed logs in this channel alone, which presents the logs of hundreds of thousands of victims.

You could easily fill a 4 TB SSD with the logs you collect in one day. Some people save everything, but that can be a challenge. You could download as much as you can fit and then query the data with Ripgrep. However, what about the new 50 GB of compressed data which will be posted tomorrow? What about the next day? Collecting and storing this data can be a full-time job. This is where I offer two paths to consider. You can continue to download the full logs and clean them, or rely on other people to do your dirty work. Let's understand both options, and start with the manual approach. Afterward, I present a much easier solution which collects less data.

Full Stealer Log Cleansing

Stealer logs are large. I estimate we now have over 40 TB of stealer log data. That is not only a lot of data to store, it can take a long time for Ripgrep to query. At my company, we place everything within a custom database, as explained later, but that is not feasible for most people. While writing this section, I downloaded over 25 compressed files which contained over 30 GB of data including over 37,000 victim logs. That was just as an afternoon demonstration. Imagine how quickly this would pile up if you do it every day. Because of this, I offer an option for those without the need for all of the data included within a typical log.

Most people find the email addresses, usernames, and passwords to be the most valuable part of this data. While the screen captures, documents, and cookies are beneficial, they take up the most space. If desired, you could just keep all of the password files and delete the rest. Some fellow data enthusiasts just cringed at that, but I want to be realistic. Searching through millions of files in order to locate the email addresses and passwords of a suspect may be overkill for some. Instead of saving the individual password files, let's merge them all into one file. Imagine you have thousands or millions of stealer log files decompressed within your Downloads folder. The following Terminal command would navigate to each "passwords.txt" file; extract all of the text data; and compile it into one file titled Passwords.txt. Make sure you are in the right folder within Terminal, which is usually your "Telegram Desktop" folder.

```
find . '*pass*.txt' -exec cat {} \; >
~/Downloads/Passwords.txt
```

We have a problem. Some stealer software titles the passwords file "Passwords.txt" with an uppercase "P". Therefore, the following command is better. It ignores the case of the file and would extract data from both "passwords.txt" and "Passwords.txt". We are using '*pass*' to pick up all files including "pass" anywhere in the name.

```
find . -iname '*pass*.txt' -exec cat {} \; >
~/Downloads/Passwords.txt
```

If desired, you could replicate this for " ImportantAutofills" files with the following.

```
find . -iname 'importantautofills.txt' -exec cat {} \;
> ~/Downloads/ImportantAutofills.txt
```

I executed both commands against my test data, which produced a 200 MB Passwords.txt file and a 20 MB ImportantAutofills.txt file. These could be queried quite easily. If I had a terabyte of logs, my passwords file might be only 8 GB. While we keep all of the data we acquire, we also create an ever-growing Passwords.txt file which can be quickly queried for times we may not have immediate access to our entire database. If you were to execute the following command, the date would be appended to your new file, such as Passwords.2023-11-28.txt.

```
timestamp=$(date +%Y-%m-%d) && find . -iname
'*pass*.txt' -exec cat {} \; >
~/Downloads/Passwords.$timestamp.txt
```

The results within this large text file are split over several lines. This will require a variation of our Ripgrep command. As an example, the following text might be present within the file.

URL: https://www.amazon.com/ap/signin
Username: test123@gmail.com
Password: pass456

If we conduct a traditional Ripgrep search for "test123@gmail.com", the result would be as follows.

Username: test123@gmail.com

This is because Ripgrep only displays the line which possesses the data, and not the lines immediately before or after. The data within stealer logs typically splits the results over multiple lines. Therefore, if you want to query this type of data, I recommend the following Ripgrep command.

```
rg -aFiN -A2 -B2 test123@gmail.com
```

This conducts our typical Ripgrep query, but adds "-A" and "-B" followed by "2". This tells Ripgrep to display the two lines before and after each hit, which would appear as follows with our test data.

==========
URL: https://www.amazon.com/ap/signin
Username: test123@gmail.com
Password: pass456
==========

I suspect some readers are overwhelmed at the amount of data available for investigative use. I get it. I am overwhelmed every day. However, I believe that is part of the fun. If you need to keep screen captures of victim machines and any documents absorbed by the viruses, you will need to download, extract, and store entire stealer logs. Eventually, you will need a server with tens of terabytes of storage. That is what I do, but it is not necessary for most readers. This is where stealer log summaries enter the scene, and may be a more appropriate solution for you.

Stealer Log Summaries via Telegram

Now that there are so many stealer log channels on Telegram, we are seeing new rooms which only offer compressed text-only versions of stealer logs. These channels offer text file downloads of all URLs, usernames, and passwords recently posted within the full stealer logs which I just explained. There are benefits and disadvantages for each, so consider the following.

- Full stealer logs contain screen captures, cookies, and other documents which are not present within the stealer log summaries. This can identify priceless information about your target.
- Full stealer logs assign a folder to a specific victim. You can use this to associate multiple email addresses to a single user. Summaries combine all credentials into one big data dump.
- Stealer log summaries have already been cleaned for you. You simply need to download the text files, combine them, and sort for unique entries. You will only download the important data without the rest of the garbage. Instead of downloading a 1 GB compressed logs file, you can download a 10 MB summary of the credentials.

A typical entry within full stealer logs is as follows:

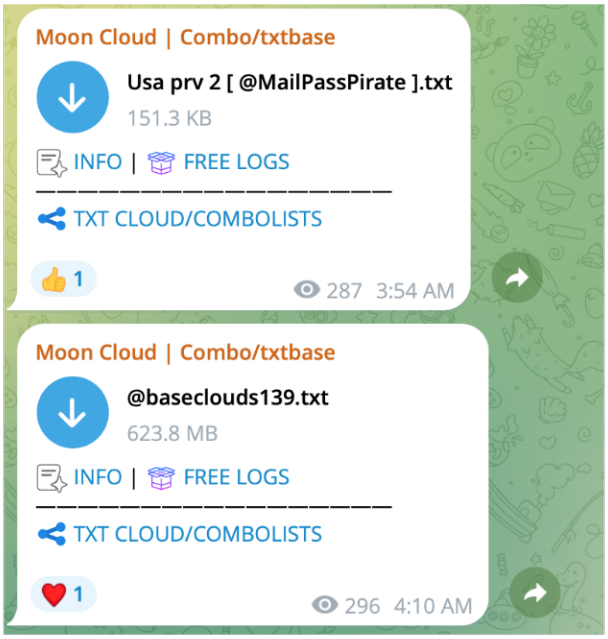
URL: https://www.amazon.com/ap/signin
Username: test123@gmail.com
Password: pass456

The same entry would appear as follows in a summary file.

https://www.amazon.com/ap/signin:test123@gmail.com: pass456

I will now assume you do not want to collect several terabytes of compressed files and parse through all of them simply to identify the credentials within stealer logs. Instead, we will download stealer log summaries. Much like the previously mentioned channel of combined log sources called "Moon Cloud | Free Logs", we will focus on another channel called "Moon Cloud | Combo/txtbase" to download summaries.

There are two types of files in this room. The first are stealer log summaries, which tend to be at least 40 MB text files. The other are combo files from these summaries which tend to be less than 20 MB in size, often only a few kilobytes. The following image displays an example. The top file is only 151 KB and includes email addresses and passwords which were acquired from stealer logs. The second file contains 623 MB of URLs, email addresses, and passwords from stealer logs. For this section, we will focus on the latter.

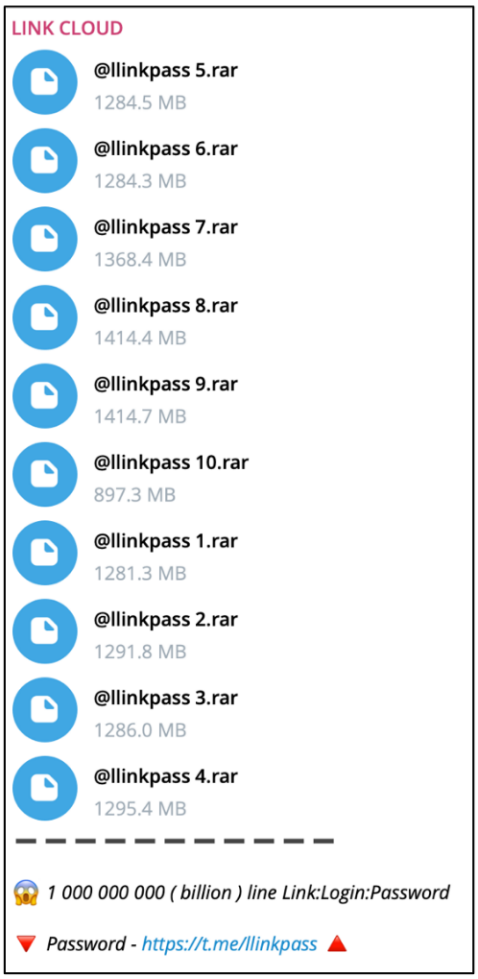


Let's start with the file seen in the previous image. After download, I can immediately access the data without decompression. The file is small enough I can open it within a text editor. The following is a redacted view of the first line.

kevinsaenxxxxient:kevinsaenxxxxientexx:https://www.netflix.com/ar/Login

This indicates that a person with that email address has been infected with a stealer virus; they stored their credentials directly to their web browser; they have a Netflix account; and the username and password combination is used to access that account. If you worked for Netflix, you could use this information to disable the account and prompt the user for a password reset. If you were a criminal, you would illegally use this to access free movies. Let's do another example.

Occasionally, a group will publish a huge collection of stealer log data as compressed files. The channel "Link Cloud" with a Telegram username of "@linkpass" is notorious for this. While writing this chapter, they posted 6.7 GB of compressed files which decompressed to 71.1 GB of text files containing stealer log summaries. Note the password required to decompress the data in the following image. All files should be decompressed before proceeding.



When combined with the posts from the other room, I possessed 76.5 GB of text data. This is quite a feat, and many might think the work is done. While you could offload all of these files onto a SSD for queries, you have two problems. First, many files will have the same file name but unique content. Therefore, saving all files downloaded over a year in one folder will be an issue. The bigger concern is duplicates. Since these groups steal log data from each other, that 76.5 GB of data most certainly contains redundant information. There is no justification to store duplicate entries.

Therefore, I will share my daily and weekly ritual with you. This method will download all available summaries every day; combine the daily haul into one file without any duplicates; then merge the data into your primary collection weekly, avoiding redundant data. While I hope you follow along and replicate my steps, I offer a script soon that will do all of the work on your behalf.

Daily Tasks

On today's date, I downloaded a total of 80 GB of decompressed text files containing stealer logs. That is higher than normal, but not completely out of the ordinary. I then executed the following commands within a single Terminal window, with an explanation of each. A replica of these will be used in our script.

The following creates a time stamp which we can use to name our file later. This is important so that we do not overwrite our collected data every day.

```
timestamp=$(date +%Y-%m-%d-%H-%M)
```

The following changes the active Terminal directory to the folder used by Telegram by default for downloads. Your folder may vary.

```
cd ~/Downloads/Telegram\ Desktop
```

The following command combines all text files into one file called Logs.csv.

```
cat *.txt > Logs.csv
```

The following commands delete all the original text files, then renames our working combined file to Logs.txt.

```
rm *.txt && mv Logs.csv Logs.txt
```

Next, we need to eliminate the duplicates with the following command.

```
LC_ALL=C sort -u -b -i -f -S 80% --parallel=8 Logs.txt
> Logs-$timestamp.txt
```

The result was only 57 GB of data within my new text file (down from 80 GB originally). Finally, the following commands remove our original Logs.txt file and moves our new file to our Downloads directory. This is to ensure that the next time we run this series of commands, we do not delete the file we just created.

```
rm Logs.txt && mv Logs-$timestamp.txt ~/Downloads/
```

Your daily task is complete and you would replicate these steps tomorrow with any new stealer log summaries posted to your joined Telegram channels.

Weekly Tasks

If you download the stealer log summaries every day or two and conduct these steps, you will soon have a substantial collection of stealer log data. I often find myself with a folder of files such as the following.

Logs-2023-11-25-09-39.txt
Logs-2023-11-26-09-48.txt
Logs-2023-11-27-09-19.txt
Logs-2023-11-28-08-41.txt
Logs-2023-11-29-09-00.txt

I can see from these file names that I tend to complete this process during the 9:00 hour. I try to collate these files every week or two in order to query them from my SSD. The following two commands merge all sorted logs files within the Downloads folder into a single file which should be free of any duplicates.

```
cd ~/Downloads && timestamp=$(date +%Y-%m-%d-%H-%M)
LC_ALL=C sort -u -b -i -f -m Logs* > Merged-
$timestamp.txt
```

This creates our timestamp and takes any SORTED files in the current directory which begin with "Logs" and MERGES (-m) them into a single file. Finally, we can delete the original Logs files which are no longer needed with the following command.

```
rm Logs*
```

You can now move this merged file into your SSD. My SSD is labeled DATA and I have a folder at the root of the drive titled "Logs". The following command moves my new file to that folder if using macOS. The second command would move it if using Linux.

```
mv Merged* /Volumes/DATA/Logs
mv Merged* /media/$USER/DATA/Logs
```

You could collect these "Merged" files in your SSD and query them. There is no harm in that, as each file possess a unique name due to the timestamp. However, you will find the duplicates start piling up again. Several stealer logs will upload the same data every day from actively-infected computers. Therefore, I prefer to keep my primary stealer log collection free of redundancies.

Assume you now have a file similar to Merged-2023-11-23-09-11.txt (November 23, 2023) in your SSD Logs folder. It possesses all of the stealer log summaries which you collected over a week. It is now one week later, and you have seven new sorted Logs text files in your Downloads folder after conducting the "Daily Tasks" section previously mentioned. As also previously stated, you could merge them all and copy the new file to your SSD, but then we have redundancy issues. The following four

macOS commands merges all of the sorted log files in your Downloads directory with your current primary merged file(s) in your SSD Logs folder to create a new file which possesses all of the data without any redundancies. The original files are deleted and this new file is copied to your SSD Logs folder.

```
cd ~/Downloads
LC_ALL=C sort -u -b -i -f -m Logs* /Volumes/DATA/Logs/* >
Current.txt
rm Logs* /Volumes/DATA/Logs/*
mv Current.txt /Volumes/DATA/Logs/Current.txt
```

The following commands for Linux are slightly different.

```
cd ~/Downloads
LC_ALL=C sort -u -b -i -f -m Logs* /media/$USER/DATA/Logs/*
> Current.txt
rm Logs* /media/$USER/DATA/Logs/*
mv Current.txt /media/$USER/DATA/Logs/Current.txt
```

That was a lot. Let's summarize everything from the beginning. We collected a bunch of stealer log summaries from Telegram into the default Telegram downloads folder. After the daily downloads completed, we sorted all of the text files into a single file without any duplicates, and moved it to our Downloads folder. It appeared similar to Logs-2023-12-01-09-52.txt. We repeated this every day. Whenever these sorted files started to pile up, we merged them all into one file without any duplicates. We moved this file to our SSD for future queries. The next week, we repeated the process and had several new sorted Logs files in our Downloads directory. We executed a command to combine them with the file on our SSD to make a new primary logs collection file with no duplicates. We moved that to the SSD and eliminated the unnecessary files. You could repeat this every week (or month) as desired. Your "Current.txt" logs file will slowly grow without duplicates on your SSD.

Was it worth all the effort? Only you can decide. I currently possess a folder with 300 GB of unique stealer logs which I query almost every day. There are billions of credentials in there. It is priceless to me. At the time of this writing, the "Moon Cloud" and "Link Cloud" channels possessed many terabytes of stealer log summaries, in text format, dating back to August 2023 (with logs dating back to 2015). After downloading them all and removing duplicate entries, you should be able to replicate my entire 300 GB collection of most known stealer log credentials in one day. In fact, you will likely possess the exact same data as the numerous new companies selling stealer logs to organizations as a defense technique. Why pay for partial access when you can possess all of the data yourself?

In a moment, I will present a script which displays two sorting options. The first conducts your daily tasks for you while the second completes the weekly tasks. This script will also query all logs data. If you plan to only dabble in this type of data, these tutorials serve as an example to the threats and investigative benefits of stealer logs. The only way you could truly benefit from them within your own investigations is to

download everything you find daily. That is quite a burden. I hired a full-time employee to do just that. He constantly acquires new leaks, breaches, logs, and ransomware, and then imports into our custom database for use by my staff. That comes with quite an expense, but we see results which justifies the activity.

Once you have built a massive collection of these logs, you can use Ripgrep to query them as previously mentioned. In 2021, I was investigating an unknown person harassing one of my clients. He was using a throwaway email address which seemed impossible to trace. It was not present within any breach data. However, it appeared within my stealer logs, which included a device name similar to Desktop-u3ty6. Searching that device identifier presented dozens of email addresses and passwords in use on that machine. This quickly revealed my suspect, a 15-year-old kid. Further investigation confirmed his computer became infected after downloading a pirated version of anti-virus software. The irony.

One week prior to writing this, I was investigating a financial account takeover attempt on behalf of a client. Someone switched her bank account to a "burner" email address. This criminal apparently possessed a stealer virus, because searching that email address immediately hit upon the entire logs for his computer. This included his "FileGrabber" folder which contained a recently modified resume. I was able to provide a full dossier on the suspect to my client, and the entire investigation took less than five minutes.

I suspect my attorney will scold me if I continue to provide more examples. Instead, I will leave you with a list of known Telegram stealer log rooms for you to conduct your own research. These include Base Cloud, Boosty, BradMax, Bugatti Cloud, CBank, Cloud Logs, Cloud Redhat, Crypton, DB_Cloud, Eternal Logs, Expert Logs, Hub Head, HUBLOGS, Keeper Cloud, Link Cloud, Link:Login:Pass, Logs Arthouse, Logs Cloud, Luffich Cloud, Luxury Logs, Magican Cloud, Moon Cloud, NinjaByte, Observer Cloud, OneLogs, Redlogs Cloud, Ruban Cloud, Satanic Cloud, Snatch Logs, Syxap's Cloud, Tor Logs, Wild Logs, xcloud, and X-Cloud. This is only a sample, and there are many others which emerge and disappear daily. Start with Google.

I offer one final recommendation for Telegram: take advantage of the search field. As I wrote this, I noticed an announcement about a new breach at a cryptocurrency provider. Searching the name of the service within Telegram immediately presented numerous free download options. Follow as many breach-related channels in order to increase search results of interesting data. You can also search sites such as BreachForums for logs, as a lot of members post direct download links which do not require purchased or earned credits. The following query would identify posts on BreachForums which may be helpful.

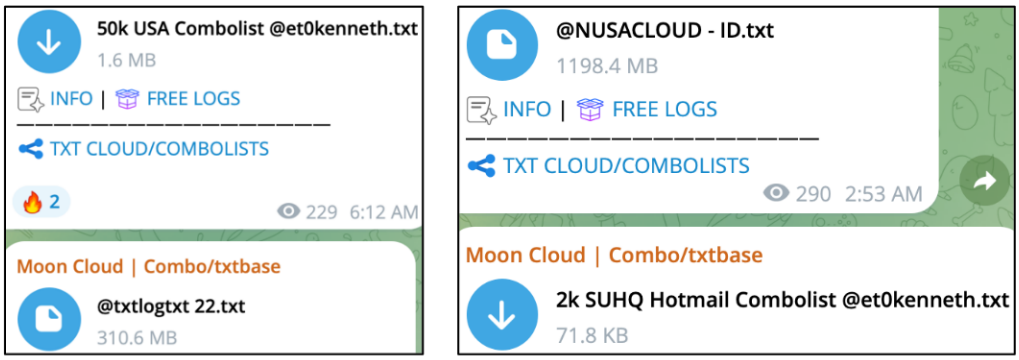
site:breachforums.is "Url-Log-Pass"
site:breachforums.is "mega.nz" "stealer log"

You may also have success with the following queries.

site:anonfiles.com "logs"
site:mediafire.com "logs"

Combo Files

If you were browsing the stealer logs offerings within the "Moon Cloud | Combo/txtbase" channel, you likely noticed a lot of smaller files which did not contain stealer log summaries. These are "Combo" files, which are typically the usernames and passwords extracted from the stealer log summaries. Most of the time, the content within these files is also available within the stealer log summaries, but that is not always the case. I have downloaded many combo files which contained data not present within any stealer logs. Consider the following screen captures.



In the first image, the top file is only 1.6 MB while the lower is 310 MB. This is the first sign that the larger file contains stealer logs while the smaller possesses only usernames and passwords. In the second image, these are reversed. The top is a large stealer log summary while the bottom possesses only 2,000 Hotmail usernames with passwords. These channels will often extract credentials from stealer logs for a specific service (in this case Hotmail), and offer them to people who may want to target that service with illegal actions.

You may want to download all of these combo files for your own collection. I always recommend doing this after you have acquired and sorted the logs for the day. Otherwise, the combo files will get mixed into your logs. The process is the same. Download all of the desired combo files into an empty Telegram Downloads folder; remove duplicates; and move the result out of your Telegram directory. The following modified commands should work for macOS and Linux users. Note that the sort command is on two lines.

```
timestamp=$(date +%Y-%m-%d-%H-%M)
cd ~/Downloads/Telegram\ Desktop
cat *.txt > Combos.csv
rm *.txt && mv Combos.csv Combos.txt
LC_ALL=C sort -u -b -i -f -S 80% --parallel=8
Combos.txt > Combos-$timestamp.txt
rm Combos.txt && mv Combos-$timestamp.txt ~/Downloads/
```

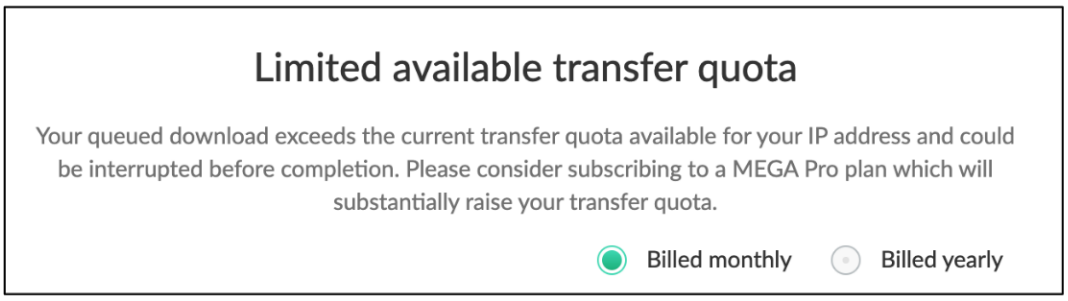
If you can only collect one type of data, I highly recommend logs over combos. Most combo files will display thousands of lines in the format of EMAIL:PASSWORD.

Mega.nz Issues

As you continue your journey into data collection, you will likely find several links on the popular file sharing website Mega (mega.nz). This service provides 50 GB of free storage, and is often used by criminals to store and share leaks, breaches, and logs. Download speeds are typically fantastic, but we have a new problem. Free users have a limited amount of bandwidth they are allotted within a 24-hour period. Even worse, those of us using a VPN, as highly recommended, are sharing IP addresses with people abusing Maga's free tiers. This often results in blocks from Mega encouraging us to purchase a paid plan. I cannot completely eliminate these hurdles, but I can make them much more tolerable. For this, I recommend Megatools, which can be installed with the following steps.

macOS: `brew install megatools`
Linux: Download and install from https://megatools.megous.com/builds/builds/
Windows: `choco install megatools`

Once installed, we can download a Mega link directly via terminal. A demonstration should explain the benefits of this method over a traditional web browser download. I opened Telegram and searched "mega.nz". I was presented a post which claimed to offer 63 GB of stealer logs hosted on Mega. I clicked the link and was immediately presented the following warning from Mega.



I then opened Terminal; changed the directory to my Downloads folder; and entered `megatools dl` followed by the same Mega file URL. The download began immediately and it continuously disclosed the download speed and percent of completion. Whenever I encountered a bandwidth restriction, the Terminal command kept my place and retried to resume every couple of minutes. When that happened, I simply switched VPN servers and the download continued. This would not have helped with the browser download, as the session cookies monitor your overall usage, regardless of your IP address.

I use this method for every large Mega download I encounter. Not only does it bypass some of the restrictions, it is not prone to browser memory problems which can result in corrupted or incomplete downloads.

Criticism

I know what some readers are probably thinking. You may believe that sharing these types of details is irresponsible and will lead to further abuse of the victims of these attacks. I respect your opinion and understand your concern. I also believe that thinking is naive. The criminals who are going to abuse this data do not need a tutorial from me. If anything, they would laugh at my basic understanding and explanation of their world. I am sure they could teach me better ways of doing all of this. I do not believe anyone is going to buy this book and begin further abusing this type of data.

Hardware stores sell hammers; websites sell lock picks; and local stores sell guns. All of these can (and are) abused. That does not mean that they should be restricted from those who are responsible with the items. I would rather take the chance that one person will misuse this information if it helps a thousand people use it for good. The more analysts and researchers who understand and collect this data, the better we will all be. If we had enough large companies monitoring and acting upon this data, we may see it go away because it is no longer useful to the criminals. Ignoring these techniques allows the criminals to continue their games and win. If I was the CEO of PayPal, I would dedicate employees to this task; have them identify any presence of PayPal credentials; and disable them before they are abused. You could replace PayPal with any other financial institution in those statements.

There are books about detailed computer hacking, malware creation, and various "black hat" lessons. They are used by security researchers, penetration testers, and other professionals who are tasked with responding to bad situations. My hope is that this book can help those who can use this data for good (defense and offense). I would have loved to have all of this when I was investigating cyber-crime for the FBI, but they probably would not have allowed me to use it. We can either pretend there is not a problem or we can take action to stop some of this mess. We can either allow criminals to have all of the power with this data or we can use it against them. I have witnessed countless examples of criminals accidentally infecting their own machines and exposing their own data.

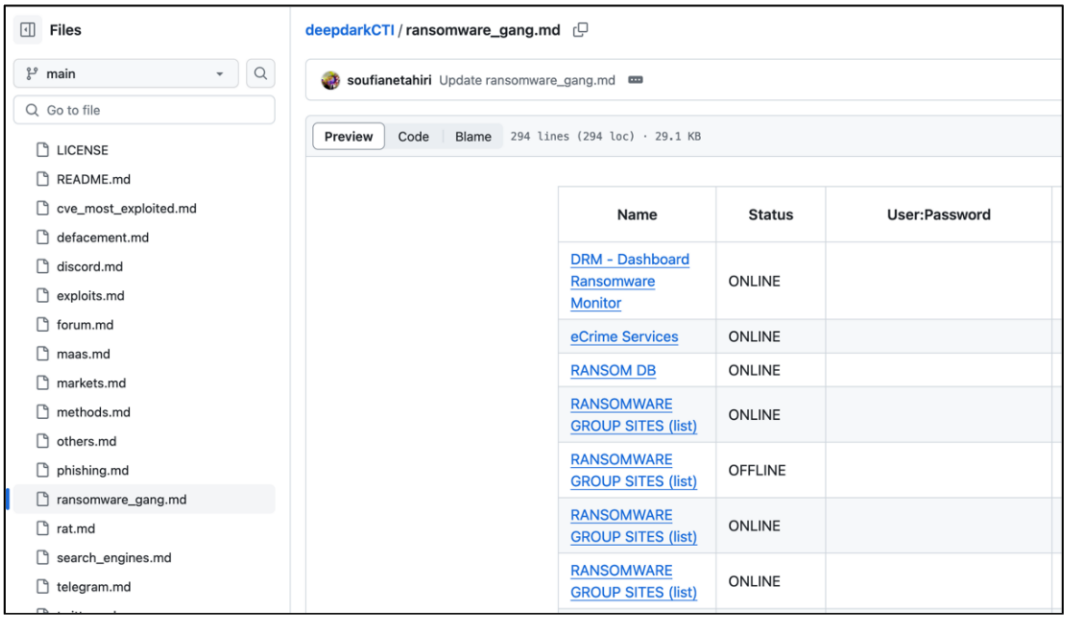
CHAPTER EIGHT
RANSOMWARE

If you thought I was pushing the boundaries of ethical data acquisition, the following might make you uncomfortable. You have likely heard about ransomware infection. It is the illegal activity by criminals which steals data from a company, encrypts all of their files, and demands a ransom to gain access to the unusable data remaining on their own servers. When companies began creating better backups which eliminated the need to pay the ransom, the criminals took a new route. If the victims do not pay, all of their data is uploaded to the internet via Tor websites for anyone to download. This generates terabytes of private documents, email messages, databases, and every other imaginable digital file. Is this OSINT data? I don't think so. However, an investigator working on behalf of a victim company should know where to find this information.

Most sites which announce and distribute ransomware will require the Tor Browser. However, we will also retrieve recent data posted within clear-source options such as Telegram. Let's start with a demonstration using the Tor browser. The first step is to identify the current URLs for the various ransomware groups. I present the following resource, which should be all you need to identify your targets.

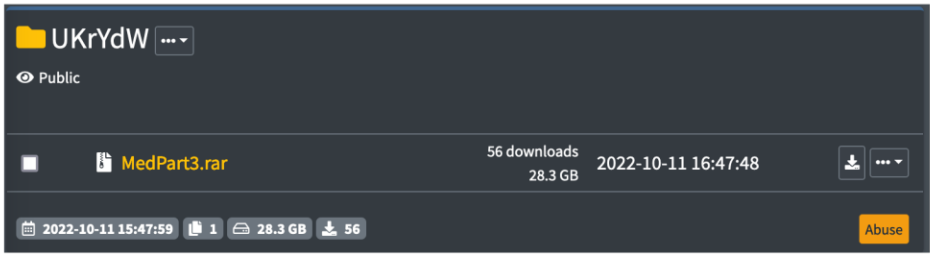
https://github.com/fastfire/deepdarkCTI/blob/main/ransomware_gang.md

This collection does not require access to the Tor network, but many of the sites included will. This site attempts to provide a current status of various criminal groups; direct links to otherwise hidden sites offering data; and identify sites which have gone offline. The following figure displays a current partial view of this service.



You should have already installed Tor Browser by following the steps presented in a previous chapter. If you did not, navigate to https://www.torproject.org/download/ and install the Tor Browser for your operating system. You should then be able to copy any links from the previous Github page and paste them into Tor Browser to see the content. Any website ending in ".onion" will require the Tor Browser.

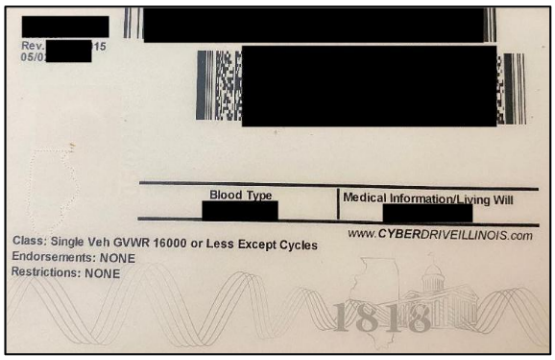
I copied the Everest link to navigate to the official Everest ransomware distribution page. I then saw dozens of victims listed within their page. I will not identify any victims throughout this chapter, and will focus only on the data and type of organization. I clicked on the first victim, which was a non-profit health care organization which was hit with ransomware in October 2022. The page offered three links hosted at gofile.io, each approximately 30 GB compressed. The following figure displays one of the download options. Notice we can see the number of successful downloads. This could be vital if you are investigating this breach on behalf of the company. This confirms people are actively downloading the stolen data from this group.



If you were to download these files, you would have 120 GB of data which was stolen during the breach. Most of the data is of no use to an online investigator. We intentionally destroy anything which contains intellectual property or internal housekeeping affairs. The breach provided over 37,000 medical documents including patient records. This breach also included many copies of driver's licenses (front and back), insurance cards, and other forms of identification. This data is quite invasive and you may feel uncomfortable accessing any of it. I completely understand. We digest it into our system in order to make immediate notifications to any of our clients which are present within the breach. We only use this data for defensive practices, and mark all medical data to be ignored when staff conducts OSINT queries into the data.

Recently, we took in a large ransomware data set which included scans of the front and back of thousands of customers' driver's licenses. Our systems analyze all images, documents, and PDFs, and extract any text content via Optical Character Recognition (OCR). This allows us to query or monitor our clients' details, which can notify us when their identification cards or financial data is present online. This process has weaknesses, especially when scanned images are of poor quality. The text must be properly identified in order to receive any alerts about client data.

We also started scanning all documents for any type of barcode, including QR codes and license barcodes. This has paid off greatly, and this recent breach turned out to be quite beneficial. Our automated barcode scanning element immediately identified the text details of all customers, which allows us to query by name, address, DOB, DL, etc. The following figure (left) displays a redacted back of a scanned DL from this recent breach.



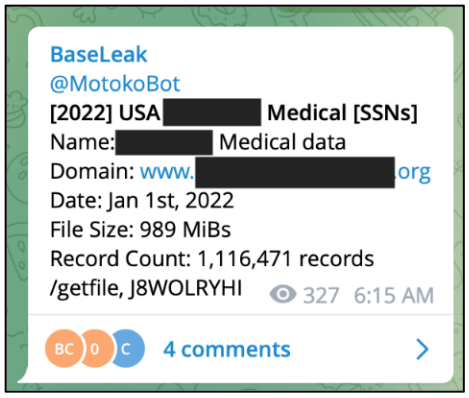

```
<last e="DCS">Sc          /last>
<first e="DAC">J          /first>
<middle e="DAD">          le>
<dob e="DBB">199          </dob>
<eyes e="DAY">BL          >
<sex e="DBC">F</
<height e="DAU">          eight>
<weight e="DAW">          ight>
<street e="DAG">          berleaf Ct</street>
<city e="DAI">O           /city>
<state e="DAJ">I          e>
<postal e="DAK">          000</postal>
<country e="DCG"          ountry>
<id e="DAQ">S636          6</id>
<issued e="DBD">          -21</issued>
<expires e="DBA"          5-02</expires>
```

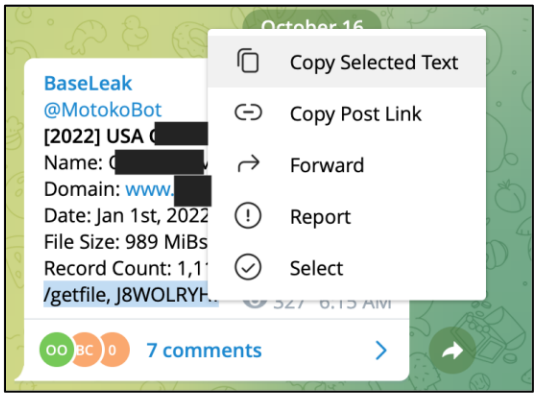
The figure (right) displays the (redacted) automated text conversion of that barcode, which allows us to ingest text data into our overall breach database for easy access. The first lesson here is that barcode analysis of entire data sets can reveal much more data about the victims within the breach. In this example, our systems ingested the images; performed OCR on all text; scanned the barcodes; populated all text data within our database; and alerted us that a client's driver's license was within the breach, all without any manual effort from us.

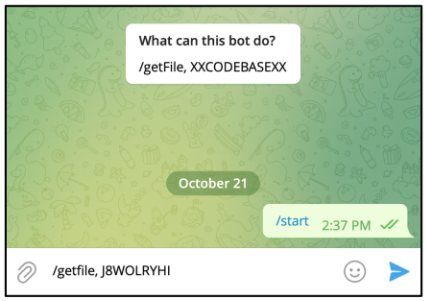
The second lesson is to never allow companies to scan your identification. It will never be stored securely and will be leaked during the next attack. The barcode on the back of a passport card only includes the passport number, and may be much safer whenever required to hand over ID. My driver's license possesses a vinyl sticker on the back which contains a new barcode. Scanning it reveals a stern text message about consumer rights. I have yet to allow anyone to scan it though.

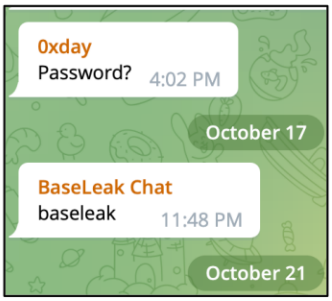
The Everest site has terabytes of data published for anyone to see. Add that to the other dozens of groups and you have more data than you likely want to store. We typically eliminate large files such as PDFs and Office documents once our system has ingested the text from the data. Let's conduct another example on the clear web and look at some pure text data which would be much easier to analyze.

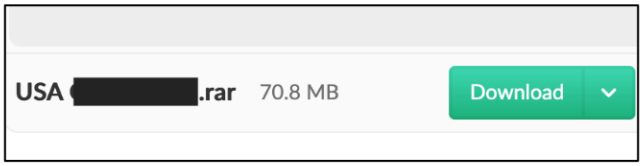
In October of 2022, I reviewed a Telegram channel called "baseleak". This channel was offering a 1 GB download of recently stolen medical data in text format. The first figure displays the original post. However, there was no instant download option. Please note I have redacted these posts in order to protect the identity of the victim organization. This group adds a layer of complexity in order to retrieve and analyze the information. I had to copy the text on the last line, as visible in the second figure. I then had to click "@MotokoBot" to load the view seen in the third figure), where I had to paste this text. The password to access the file was present in the comments of the post, as seen in the fourth figure. This finally unlocked the link, which is visible in the fifth figure, which connected me to the download site Mega, as seen in the final figure. The entire process took only a couple of minutes.











The 70 MB compressed file extracted to an almost 1 GB CSV file titled "data.csv". It contained the following 92 fields of data.

Master Patient ID, Name, Sex, Date of Birth, DOB Approximate, Age, AgeInDays, AgeText, PrematureDays, GestationalAge, BSA, Ideal Body Weight, Height, Weight, HeightMod, WeightMod, SSN, BMI, Brand Preference, Safety Caps, Race, Language, Religion, Pregnant, Breast Feeding, Smoker, Address 1, Address 2, City, State, ZIP, Last Room, Last Hospital#, Last Admit Date, Last Discharge Date, Last Visit #, AllergyList, DiagnosisList, PhoneNumber, AssigningAuthority, Visit #, Medical Record #, Hospital #, Room #, CurrentOrPreviousRoom, Room, Prev Room #, Admit Date, Discharge Date, PlannedDischargeDate, PCU, Facility Code, Service Type, Service Type Description, CommentsTextID, Physician Code, Physician, Unverified Orders, CrCl, AdminRecord, Diet Code, Diet Description, Clinical Service, PassStart, PassStop, Encounter Number, AccountNumber, AllowLateCharges, AccountNumberAssignAuth, MedicalRecordNumAssignAuth, HospitalNumberAssignAuth, InternalCommentsTextID, InternalCommentsText, CommentsText, org_int_id, loc_lvl_3_id, loc_lvl_4_id, loc_lvl_5_id, pcu_int_id, pat_grp_int_id, psn_int_id, Patient Type Group, SerumCreatinine, PatientCategory, VisitActive, out_prt_lc, pat_cat_cd, vst_sta_cd, visit_status_code, PatientInBed, Complaint, FinancialClass

That is a lot of invasive and sensitive content. We avoid possessing explicit medical data, such as medical conditions or patient data. However, we do collect and store names, DOBs, SSNs, addresses, and phone numbers in order to make proper notifications to our clients. Therefore, I executed the following command to extract only the fields important to me (and my clients), as previously explained.

```
cut -d, -f1,2,3,4,6,17,27,31 data.csv > data-cleaned.txt
```

This minimized my CSV to a 228 MB txt file. I noticed there were many duplicate entries, so I executed the following, also as previously explained.

```
LC_ALL=C sort -u -b -i -f  data-cleaned.txt > Data-Final.txt
```

I then noticed a lot of extra spaces within the lines, so I executed the following to replace every two spaces with only a single space.

```
sed -i 's/  / /gI' Data-Final.txt
```

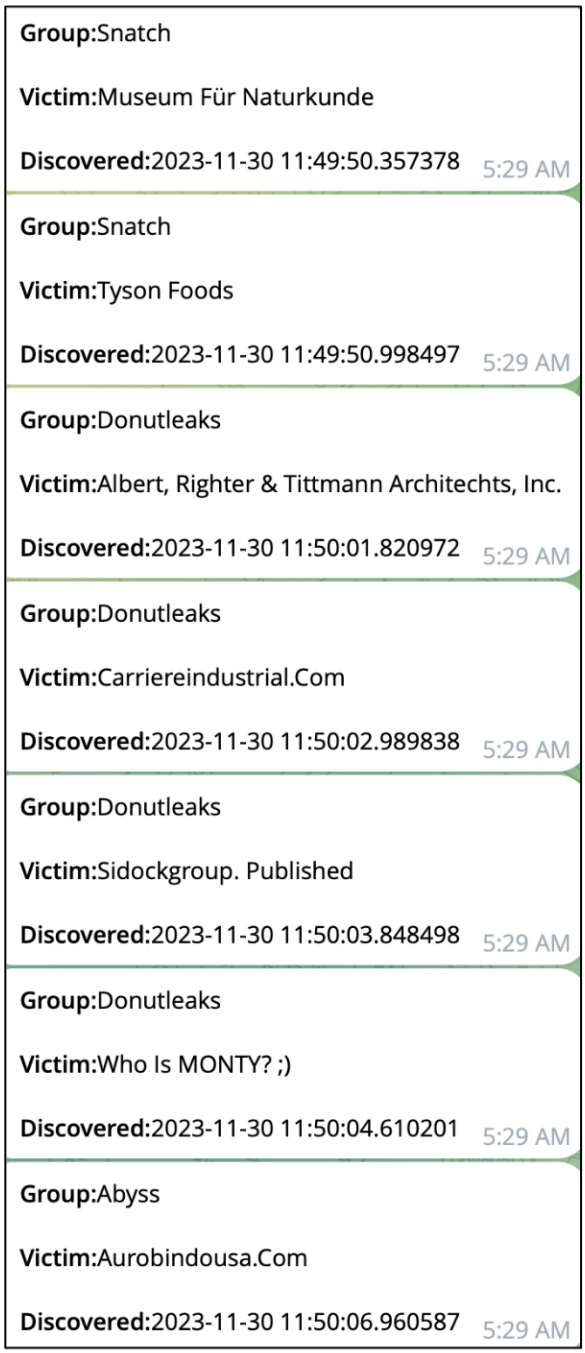
This resulted in a 31 MB text file with 375,000 unique entries. A redacted example of the text follows.

Master Patient ID, Name, Sex, Date of Birth, Age, SSN, Address 1, ZIP
00xx3, PRESTON, JAxx, F, 1982-xx-xx, 459xxxxxx,47xx Rxxxxx RD, ROxx, TX, 77xxx

This file contains extremely sensitive stolen information about 375,000 people. If any of our clients are within this data, we will receive an alert and can make notification. I hope you now see why it is so important to protect any breach data which you acquire. While it is publicly available content, you must ensure that you do not leak it further. This is why we never store any breach data within the cloud. It is all stored on a physical server in our office, and access is tightly controlled. We also monitor all activity for abuse.

Telegram Ransomware Search

Monitoring every ransomware group within their dedicated Tor websites is exhausting. Fortunately, we have an easier option. I highly recommend joining a Telegram channel called RansomWatcher. This channel monitors all of the active ransomware groups and posts a notification when a victim has been infected. The following image represents only a fraction of the victims posted the day I wrote this section. This channel does not possess or offer any sensitive data.



I can now search the name of any company within this channel to see if they appear in any known ransomware attack. If anything is present, I can research the group responsible for the attack on the previous GitHub page. I can then navigate to the group's Tor page and see if any data has been published from the attack. Another alternative to this type of monitoring is https://ransomwatch.telemetry.ltd.

If your employer tasks you with the investigation of content stolen from your organization, I want you to know what to do. I want you to locate, obtain, and analyze the exact same data which criminals are abusing. We can no longer keep our heads in the sand and hope no one notices the leakage of sensitive data. We must think like the criminals and attack from their mindset.

If you are an online investigator who can responsibly benefit from this type of data, I want you to know what to do with the content you find. I also want you to protect the data and make sure you cause no further harm. Every day, this type of data assists me with positive identification of the true identity of stalkers, harassers, scammers, and other unsavory people. We can either allow this data to be solely used by criminals to make our lives more difficult, or we can use it as a tool to expose those criminals which would otherwise remain hidden.

CHAPTER NINE
QUERIES, SCRIPTS, DATABASES, & BACKUPS

This brings us to the elephant in the room. How does someone new to breach data collection organize and query all of it? The obvious optimal solution is to create a database and then identify any fields which should be indexed when importing each breach. Elasticsearch would work well for this. However, maintaining an internal database would be a full-time job and database maintenance far exceeds the scope of this book. I believe large databases are overkill for most individual data collectors. Where does that leave us? I have simpler options.

While I was writing this section, I put my feet in the shoes of a reader new to this type of data collection. I pretended I did not have access to our investigations server and grabbed a couple of terabytes of data similar to that which has been previously discussed here. Conducting the Ripgrep queries discussed throughout the guide worked fine, but they were dreadfully slow across 2 TB of data. I needed to isolate the scope of each query to eliminate unnecessary search time.

I decided to create a script which would manually query the data in the most efficient way possible, absent a structured database. First, I categorized all files in order to limit the data for each query. I then placed everything on my SanDisk 4 TB Extreme Portable SSD (amzn.to/3GO0cP2). The drive was labeled "DATA" and possessed the following folders at the root of the drive. After each folder name, I include a brief description of the content within parentheses.

COMB (The COMB user:pass database previously explained)
Combos (Additional user:pass files)
Databases (Breached databases containing SQL, txt, csv, and other formats)
Hashes (The Hashes.org and HashMob data sets)
International (International data which may not apply to every daily investigation)
Logs (Stealer logs acquired using the previously-explained techniques)
People (Various people search database leaks and breaches)
Ransomware (Any ransomware content including txt, pdf, csv, and other data)
Voter (All voter data acquired during the previous tutorials)
Whois (All Whois data acquired during the previous tutorials)

You can either manually create these folders on your own drive, or copy and paste the following commands to replicate the structure on your own drive labeled "DATA" on your macOS machine. It is vital that your drive possesses this internal name for all of this to work.

```
mkdir /Volumes/DATA/COMB/
mkdir /Volumes/DATA/Combos/
mkdir /Volumes/DATA/Databases/
mkdir /Volumes/DATA/Hashes/
mkdir /Volumes/DATA/International/
mkdir /Volumes/DATA/Logs/
mkdir /Volumes/DATA/People/
mkdir /Volumes/DATA/Ransomware/
mkdir /Volumes/DATA/Voter/
mkdir /Volumes/DATA/Whois/
```

Linux users would conduct the following.

```
mkdir /media/$USER/DATA/COMB/
mkdir /media/$USER/DATA/Combos/
mkdir /media/$USER/DATA/Databases/
mkdir /media/$USER/DATA/Hashes/
mkdir /media/$USER/DATA/International/
mkdir /media/$USER/DATA/Logs/
mkdir /media/$USER/DATA/People/
mkdir /media/$USER/DATA/Ransomware/
mkdir /media/$USER/DATA/Voter/
mkdir /media/$USER/DATA/Whois/
```

You now have some structure for your data. You can place the files you obtain in the folder most appropriate for that type of data. Once my data was split into these folders, I created a script to query all of it. The entire text from the macOS script is presented here soon. However, let's download and prepare it with the following commands.

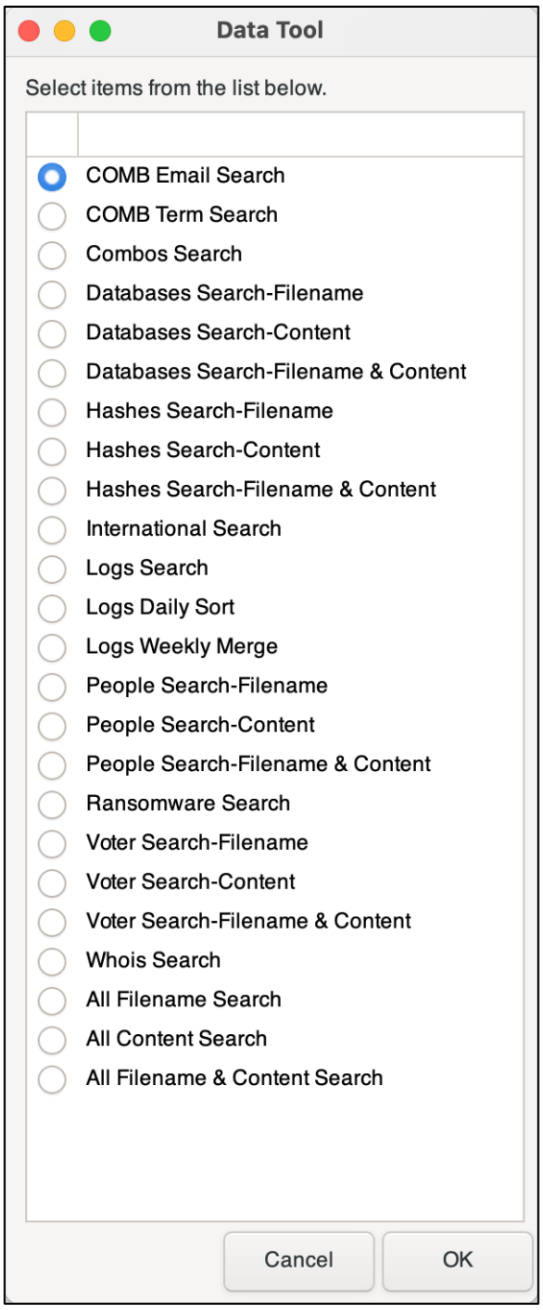
macOS Users:

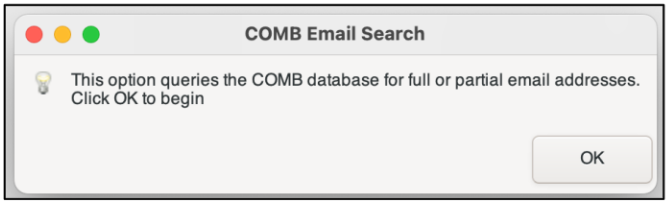
```
cd /Applications
curl -O https://inteltechniques.com/data/DataTool
chmod +x DataTool
```

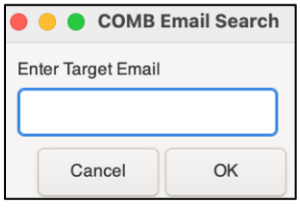
Linux Users:

```
cd /home/
sudo curl -O  https://inteltechniques.com/data/DataTool.sh
sudo curl -O https://inteltechniques.com/data/DataTool.desktop
sudo chmod +x DataTool.sh
sudo mv DataTool.desktop /usr/share/applications/
```

This will present an executable script within the Applications folder (macOS) or the Show Applications menu (Linux) called "DataTool". Clicking the file launches the script and presents the following menu. Make sure you have installed all necessary software from the previous chapters. The following images display the menu, an information notice, and query window.







The following six pages presents the entire macOS script, which is almost identical to the Linux option previously downloaded.

```
opt1="COMB Email Search"
opt2="COMB Term Search"
opt3="Combos Search"
opt4="Databases Search-Filename"
opt5="Databases Search-Content"
opt6="Databases Search-Filename & Content"
opt7="Hashes Search-Filename"
opt8="Hashes Search-Content"
opt9="Hashes Search-Filename & Content"
opt10="International Search"
opt11="Logs Search"
opt12="Logs Daily Sort"
opt13="Logs Weekly Merge"
opt14="People Search-Filename"
opt15="People Search-Content"
opt16="People Search-Filename & Content"
opt17="Ransomware Search"
opt18="Voter Search-Filename"
opt19="Voter Search-Content"
opt20="Voter Search-Filename & Content"
opt21="Whois Search"
opt22="All Filename Search"
opt23="All Content Search"
opt24="All Filename & Content Search"

menu=$(zenity --list --title "Data Tool" --radiolist --column "" --column "" TRUE "$opt1" FALSE
"$opt2" FALSE "$opt3" FALSE "$opt4" FALSE "$opt5" FALSE "$opt6" FALSE "$opt7" FALSE
"$opt8" FALSE "$opt9" FALSE "$opt10" FALSE "$opt11" FALSE "$opt12" FALSE "$opt13"
FALSE "$opt14" FALSE "$opt15" FALSE "$opt16" FALSE "$opt17" FALSE "$opt18" FALSE
"$opt19" FALSE "$opt20" FALSE "$opt21" FALSE "$opt22" FALSE "$opt23" FALSE "$opt24" --
height=700 --width=300)

case $menu in

$opt1 )
zenity --info --text="This option queries the COMB database for full or partial email addresses. Click
"OK" to begin" --title="COMB Email Search"
data=$(zenity --entry --title "COMB Email Search" --text "Enter Target Email")
cd /Volumes/DATA/COMB
bash ./query.sh $data
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt2 )
zenity --info --text="This option queries the COMB database for any term. Click "OK" to begin" --
title="COMB Term Search"
data=$(zenity --entry --title "COMB Term Search" --text "Enter Target Data")
cd /Volumes/DATA/COMB
rg -aFiN $data
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;
```

```
$opt3 )
zenity --info --text="This option queries combo files for any term. Click "OK" to begin" --
title="COMB Term Search"
data=$(zenity --entry --title "Combos Search" --text "Enter Target Data")
cd /Volumes/DATA/Combos
rg -aFiN $data
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt4 )
zenity --info --text="This option queries all file names in the Databases folder. It is used to identify
the exact file name to be searched next. Click "OK" to begin" --title="Databases Search-Filename"
data=$(zenity --entry --title "Databases Search-Filename" --text "Enter Partial Filename")
cd /Volumes/DATA/Databases
find . | grep -i $data
echo
echo "Select and copy any file path and press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt5 )
zenity --info --text="This option queries all Databases for any term. Click "OK" to begin" --
title="Databases Search-Content"
data=$(zenity --entry --title "Databases Search-Content" --text "Enter Target Data")
cd /Volumes/DATA/Databases
rg -aFiN $data
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt6 )
zenity --info --text="This option queries only a specific Database file for any term. Click "OK" to
begin" --title="Databases Search-Filename & Content"
data1=$(zenity --entry --title "Databases Search-Filename & Content" --text "Enter File Path")
data2=$(zenity --entry --title "Databases Search-Filename & Content" --text "Enter Target Data")
cd /Volumes/DATA/Databases
rg -aFiN $data2 $data1
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt7 )
zenity --info --text="This option queries all file names in the Hashes folder. It is used to identify the
exact file name to be searched next. Click "OK" to begin" --title="Hashes Search-Filename"
data=$(zenity --entry --title "Hashes Search-Filename" --text "Enter Partial Filename")
cd /Volumes/DATA/Hashes
find . | grep -i $data
echo
```

```
echo "Select and copy any file path and press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt8 )
zenity --info --text="This option queries all Hashes for any term. Click "OK" to begin" --
title="Hashes Search-Content"
data=$(zenity --entry --title "Hashes Search-Content" --text "Enter Target Data")
cd /Volumes/DATA/Hashes
rg -aFiN $data
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt9 )
zenity --info --text="This option queries only a specific Hashes file for any term. Click "OK" to
begin" --title="Hashes Search-Filename & Content"
data1=$(zenity --entry --title "Hashes Search-Filename & Content" --text "Enter File Path")
data2=$(zenity --entry --title "Hashes Search-Filename & Content" --text "Enter Target Data")
cd /Volumes/DATA/Hashes
rg -aFiN $data2 $data1
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt10 )
zenity --info --text="This option queries International files for any term. Click "OK" to begin" --
title="International Term Search"
data=$(zenity --entry --title "International Search" --text "Enter Target Data")
cd /Volumes/DATA/International
rg -aFiN $data
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt11 )
zenity --info --text="This option queries Logs files for any term. Click "OK" to begin" --title="Logs
Search"
data=$(zenity --entry --title "Logs Search" --text "Enter Target Data")
cd /Volumes/DATA/Logs
rg -aFiN $data
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt12 )
zenity --info --text="This option combines all log files within the Telegram download folder and sorts
for unique entries. Click "OK" to begin" --title="Logs Daily Sort"
```

```
timestamp=$(date +%Y-%m-%d-%H-%M)
cd ~/Downloads/Telegram\ Desktop
cat *.txt > Logs.csv
rm *.txt
mv Logs.csv Logs.txt
LC_ALL=C sort -u -b -i -f -S 80% --parallel=8 Logs.txt > Logs-Sorted-$timestamp.txt
rm Logs.txt
mv Logs-Sorted-$timestamp.txt ~/Downloads/
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt13 )
zenity --info --text="This option merges all sorted Logs files within the Downloads folder with the
primary collection on the SSD. Click "OK" to begin" --title="Logs Weekly Merge"
cd ~/Downloads
LC_ALL=C sort -u -b -i -f -m Logs* /Volumes/DATA/Logs/* > Current.txt
rm Logs* /Volumes/DATA/Logs/*
mv Current.txt /Volumes/DATA/Logs/Current.txt
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt14 )
zenity --info --text="This option queries all file names in the People folder. It is used to identify the
exact file name to be searched next. Click "OK" to begin" --title="People Search-Filename"
data=$(zenity --entry --title "People Search-Filename" --text "Enter Partial Filename")
cd /Volumes/DATA/People
find . | grep -i $data
echo
echo "Select and copy any file path and press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt15 )
zenity --info --text="This option queries all People data for any term. Click "OK" to begin" --
title="People Search-Content"
data=$(zenity --entry --title "People Search-Content" --text "Enter Target Data")
cd /Volumes/DATA/People
rg -aFiN $data
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt16 )
zenity --info --text="This option queries only a specific People file for any term. Click "OK" to
begin" --title="People Search-Filename & Content"
data1=$(zenity --entry --title "People Search-Filename & Content" --text "Enter File Path")
data2=$(zenity --entry --title "People Search-Filename & Content" --text "Enter Target Data")
cd /Volumes/DATA/People
```

```
rg -aFiN $data2 $data1
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt17 )
zenity --info --text="This option queries all Ransomware data for any term. Click "OK" to begin" --
title="Ransomware Search"
data=$(zenity --entry --title "Ransomware Search" --text "Enter Target Data")
cd /Volumes/DATA/Ransomware
rg -aFiN $data
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt18 )
zenity --info --text="This option queries all file names in the Voter folder. It is used to identify the
exact file name to be searched next. Click "OK" to begin" --title="Voter Search-Filename"
data=$(zenity --entry --title "Voter Search-Filename" --text "Enter Target Data")
cd /Volumes/DATA/Voter
find . | grep -i $data
echo
echo "Select and copy any file path and press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt19 )
zenity --info --text="This option queries all Voter data for any term. Click "OK" to begin" --
title="Voter Search-Content"
data=$(zenity --entry --title "Voter Search-Content" --text "Enter Target Data")
cd /Volumes/DATA/Voter
rg -aFiN $data
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt20 )
zenity --info --text="This option queries only a specific Voter file for any term. Click "OK" to begin"
--title="Voter Search-Filename & Content"
data1=$(zenity --entry --title "Voter Search-Filename & Content" --text "Enter File Path")
data2=$(zenity --entry --title "Voter Search-Filename & Content" --text "Enter Target Data")
cd /Volumes/DATA/Voter
rg -aFiN $data2 $data1
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt21 )
```

```
zenity --info --text="This option queries all Whois data for any term. Click "OK" to begin" --
title="Whois Search"
data=$(zenity --entry --title "Whois Search" --text "Enter Target Data")
cd /Volumes/DATA/Whois
rg -aFiN $data
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt22 )
zenity --info --text="This option queries all file names. It is used to identify the exact file name to be
searched next. Click "OK" to begin" --title="All Filename Search"
data=$(zenity --entry --title "All Filename Search" --text "Enter Partial Filename")
cd /Volumes/Data/
find . | grep -i $data
echo
echo "Select and copy any file path and press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt23 )
zenity --info --text="This option queries all data for any term. Click "OK" to begin" --title="All
Content Search"
data=$(zenity --entry --title "All Content Search" --text "Enter Target Data")
cd /Volumes/Data/
rg -aFiN $data
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;

$opt24 )
zenity --info --text="This option queries only a specific file for any term. Click "OK" to begin" --
title="All Filename & Content Search"
data1=$(zenity --entry --title "All Filename & Content Search" --text "Enter File Path")
data2=$(zenity --entry --title "All Filename & Content Search" --text "Enter Target Data")
cd /Volumes/Data/
rg -aFiN $data2 $data1
echo
echo "Press Enter to return to menu"
read data
exec /Applications/DataTool
;;
esac
```

Let's break down the sections. The following identifies the text file as a bash script.

```
#!/bin/bash
```

The following begins our menu entries as they will appear on launch.

```
opt1="COMB Email Search"
```

The following builds the menu content and size, and selects the first option as default.

```
menu=$(zenity --list --title "Data Tool" --radiolist --column "" --column "" TRUE "$opt1"
FALSE "$opt2" FALSE "$opt3" ... FALSE "$opt24" --height=700 --width=300)
```

The following begins our set of function options.

```
case $menu in
```

The following presents the first menu option.

```
$opt1 )
```

The following presents an informational dialogue to the user.

```
zenity --info --text="This option queries the COMB database for full or partial email
addresses. Click "OK" to begin" --title="COMB Email Search"
```

The following presents a menu to collect a piece of data to query.

```
data=$(zenity --entry --title "COMB Email Search" --text "Enter Target Email")
```

The following changes our target directory to the desired folder on the SSD.

```
cd /Volumes/DATA/COMB
```

The following executes a command and enters the collected target query data.

```
bash ./query.sh $data
```

The following presents instructions to return to the menu when finished.

```
echo "Press Enter to return to menu"
```

The following returns to the main menu in macOS.

```
exec /Applications/macos
```

The following represents the end of a function and then the end of the script.

```
;; esac
```

Next, let's walk through each query option within this menu.

COMB Email Search: This option prompts the user for an email address and then queries the COMB data set using the included database. Make sure the three files and "Data" folder are directly in the COMB folder at the root of the SSD. Results should appear almost immediately. While this will also accept username queries, password searching will fail.

COMB Term Search: This option prompts the user for any term and queries the same COMB data, but uses Ripgrep. The results will take more time, but you can query any portion of the data, including passwords or partial content.

Combos Search: This option prompts the user for any term and queries all data within the Combos folder with Ripgrep. The speed of results will depend on the amount of data within this folder.

Databases Search-Filename: This option prompts the user for any term and searches all filenames (not content) within the Databases folder. This identifies files of interest which can be specified in order to decrease query time. You would copy the entire path of any result of interest. If I searched "LinkedIn" and received a full path of "/Volumes/DATA/Databases/LinkedIn2022.txt", I would copy that entire path and paste it into the upcoming menu option to isolate my search results only to that file.

Databases Search-Content: This option prompts the user for any term and queries all data within the Databases folder with Ripgrep. The speed of results will depend on the amount of data within this folder. If you have thousands of files in this folder, you may want to focus on the next option.

Databases Search-Filename & Content: This option prompts the user for both the file path (identified in a previous search) and any term for a specific query. In this scenario, you would paste the "/Volumes/DATA/Databases/LinkedIn2022.txt" result from a previous query; submit it; then provide any term which should be queried within that file. This drastically reduced search time when otherwise searching a folder with thousands of large files.

Hashes Search-Filename: This option prompts the user for any term and searches all filenames (not content) within the Hashes folder. This identifies files of interest which can be specified in order to decrease query time. You would copy the entire path of any result of interest. If I searched "LinkedIn" and received a full path of "/Volumes/DATA/Hashes/LinkedIn2022.txt", I would copy that entire path and paste it into the upcoming menu option to isolate my search results only to that file.

Hashes Search-Content: This option prompts the user for any term and queries all data within the Hashes folder with Ripgrep. The speed of results will depend on the amount of data within this folder. If you have thousands of files in this folder, you may want to focus on the next option.

Hashes Search-Filename & Content: This option prompts the user for both the file path (identified in a previous search) and any term for a specific query. In this scenario, you would paste the "/Volumes/DATA/Hashes/LinkedIn2022.txt" result from a previous query; submit it; then provide any term which should be queried within that file. This drastically reduced search time when otherwise searching a folder with thousands of large files.

International Search: This option prompts the user for any term and queries all data within the International folder with Ripgrep. The speed of results will depend on the amount of data within this folder.

Logs Search: This option prompts the user for any term and queries all data within the Logs folder with Ripgrep. The speed of results will depend on the amount of data within this folder.

Logs Daily Sort: This option conducts all of the steps within the "Daily Tasks" section of the Stealer Logs chapter. You only need to make sure that all of your decompressed log files are in the default Telegram Desktop folder within Downloads.

Logs Weekly Merge: This option conducts all of the steps within the "Weekly Tasks" section of the Stealer Logs chapter. Your final result should be a new "Current.txt" file within the Logs folder of your SSD, free of any duplicate entries.

People Search-Filename: This option prompts the user for any term and searches all filenames (not content) within the People folder. This identifies files of interest which can be specified in order to decrease query time. You would copy the entire path of any result of interest. If I searched "Experian" and received a full path of "/Volumes/DATA/People/Experian2015.txt", I would copy that entire path and paste it into the upcoming menu option to isolate my search results only to that file.

People Search-Content: This option prompts the user for any term and queries all data within the People folder with Ripgrep. The speed of results will depend on the amount of data within this folder. If you have thousands of files in this folder, you may want to focus on the next option.

People Search-Filename & Content: This option prompts the user for both the file path (identified in a previous search) and any term for a specific query. In this scenario, you would paste the "/Volumes/DATA/People/Experian2015.txt" result from a previous query; submit it; then provide any term which should be queried within that file. This drastically reduced search time when otherwise searching a folder with thousands of large files.

Ransomware Search: This option prompts the user for any term and queries all data within the Ransomware folder with Ripgrep. The speed of results will depend on the amount of data within this folder.

Voter Search-Filename: This option prompts the user for any term and searches all filenames (not content) within the Voter folder. This identifies files of interest which can be specified in order to decrease query time. You would copy the entire path of any result of interest. If I searched "Florida" and received a full path of "/Volumes/DATA/Voter/Florida2022.txt", I would copy that entire path and paste it into the upcoming menu option to isolate my search results only to that file.

Voter Search-Content: This option prompts the user for any term and queries all data within the Voter folder with Ripgrep. The speed of results will depend on the amount of data within this folder. If you have thousands of files in this folder, you may want to focus on the next option.

Voter Search-Filename & Content: This option prompts the user for both the file path (identified in a previous search) and any term for a specific query. In this scenario, you would paste the "/Volumes/DATA/Voter/Florida2022.txt" result from a previous query; submit it; then provide any term which should be queried within that file. This drastically reduced search time when otherwise searching a folder with thousands of large files.

Whois Search: This option prompts the user for any term and queries all data within the Whois folder with Ripgrep. The speed of results will depend on the amount of data within this folder.

All Filename Search: This option prompts the user for any term and searches all filenames (not content) within the entire SSD. This identifies files of interest which can be specified in order to decrease query time. You would copy the entire path of any result of interest. If I searched "Facebook" and received a full path of "/Volumes/DATA/People/Facebook2022.txt", I would copy that entire path and paste it into the upcoming menu option to isolate my search results only to that file.

All Content Search: This option prompts the user for any term and queries all data within the entire SSD with Ripgrep. The speed of results will depend on the amount of data within this folder. If you have thousands of files in this folder, you may want to focus on the next option.

All Filename & Content Search: This option prompts the user for both the file path (identified in a previous search) and any term for a specific query. In this scenario, you would paste the "/Volumes/DATA/People/Facebook2022.txt" result from a previous query; submit it; then provide any term which should be queried within that file. This drastically reduced search time when otherwise searching a folder with thousands of large files.

Having all of these options reduces your query time, and gives you greater control over the data included. Instead of waiting for Ripgrep to search through 3 TB of data, you can focus on either one folder or one specific file. Once you master this technique, your search results should return quickly.

Notifications

Every time you select an option within this menu, you will be greeted with an informational notice identifying the steps which are about to take place. I believe these are important until they become a nuisance. If you are sick of these notices, you can simply remove every line which begins with "zenity --info" from your script. If that seems too tedious, you can complete the entire process with the following commands for each macOS and Linux, respectively.

```
sed -i "s/zenity \-\-info/\#\#zenity \-\-info/gI"
/Applications/DataTool

sudo sed -i "s/zenity \-\-info/\#\#zenity \-\-info/gI"
/home/DataTool.sh
```

Internal Storage

If you plan to host all of your breach data within your internal drive for the fastest queries possible, you only need to change the paths within the script. As an example, each path on the script typically begins by changing the directory to the SSD as follows.

```
cd /Volumes/DATA/
```

If you wanted to host your breach data within a directory labeled "Breaches" within your Documents folder, you would change every instance of this to the following. Make sure to maintain any subfolders such as COMB or Voter exactly as they appear in the script. You only want to change the part of the path from the SSD to internal. This is completely optional, and only appropriate for those with huge internal drives.

```
cd ~/Documents/Breaches/
```

Zenity

While writing this chapter, I was often annoyed with the way macOS implements Zenity into its system. Zenity is the software which presents selectable menus for our script. The Linux version is much smoother and more functional. I intentionally used the default macOS version of Zenity in order to keep everything similar. However, I currently use a different version of Zenity on my macOS device. If you are a macOS user, and are also annoyed at the fluidity of the menu for this script, you can change to an alternative version with the following commands.

```
brew uninstall zenity && brew install ncruces/tap/zenity
```

If you dislike the alternative version worse than the official option, you can reverse this change with the following commands.

```
brew uninstall zenity && brew install zenity
```

Qgrep

Until this point, we have been using Ripgrep to query our files. This is much faster than traditional Grep queries, but it can still take a long time to sort through a large amount of data. The speed of your drive will determine if this takes minutes or hours. Creating a proper database from scratch can quickly exceed the scope of this book, but we do have one easy option if we want to create simple databases which can be queried in a fraction of the time it would take Ripgrep. This is not a magic bullet, as it has some caveats, but it can be a game-changer for querying folders with thousands of files in seconds. You have already installed Qgrep in an earlier chapter, so you should be ready to go. We must first navigate to the folder where we installed the program. The following should apply to macOS and Linux users who followed my guide.

```
cd ~/Documents/Qgrep
```

We can now launch Qgrep and create a new entry called "Voter", which will create a database with all content in the Voter folder on our SSD, with the following commands for macOS and Linux.

```
./qgrep init Voter /Volumes/DATA/Voter/
./qgrep init Voter /media/$USER/DATA/Voter/
```

This created a configuration file within a hidden folder titled ".qgrep" in our Home folder. However, this file contains no voter data. It only has the settings for our new database. You should be prompted to update this database, as it does not have any data in it yet. The following command should complete the process.

```
./qgrep update Voter
```

You should receive a result similar to the following in a couple of minutes.

[100%] 42 files, 34228 Mb in, 10926 Mb out
+42 files; 68454/68454 chunks updated in 165.80 sec

This tells us that we now have a database of all Voter information which is ready to query with the following command.

```
./qgrep search Voter i bazzell
```

Let's break this down.

./qgrep	This is the command to run the application.
search	This tells the program we want to search data.
Voter	This identifies the database we want to search.
i	This informs Qgrep to run without case sensitivity.
bazzell	This is the term we want to query.

This process completed in a few seconds, and produced all lines within any files inside the Voter folder on our SSD which matched the search term. The same query with Ripgrep would have taken much longer. You may be wondering why we are not building these simple databases for all of our data. Well, there are issues.

- Every database you create requires additional storage of approximately 1/3 of the total data indexed on your host drive. This can add up quickly, and may introduce undesired redundancies. If you have 3 TB of data, you would need 1 TB of internal storage for the indexed database.
- The update process can only tolerate files which are less than the size of your available RAM. If every file you are indexing is less than 32 GB on a machine with 32 GB of RAM, you might be OK. The moment your stealer logs file (or any other large data set) crosses that line, Qgrep will crash.
- Every time you add any files to a folder indexed by Qgrep, you must update the program in order to receive search results for all data. This can become a nuisance if you add data often, however, it is quite fast.

By default, Qgrep stores the databases it creates on your internal drive in a hidden directory called ".qgrep" in your home folder. If you are able to view hidden files, you can navigate to this folder to see the contents. If you cannot see it, you must conduct the following to enable hidden file view.

macOS: Press the "Command" + "Shift" + "." (period) keys at the same time.
Linux: Open Files and enable "Show Hidden Files" in the upper-right menu.

You should now see the contents of the .qgrep folder. Mine appeared as follows.

Voter.cfg
Voter.qgd
Voter.qgf

My Voter folder on my SSD had 36 GB of data, and the Voter.qsd file was 12 GB. This matches with our 1/3 rule. If you decide to use Qgrep for most of your data, I would consider moving the databases to your SSD and execute the queries from there. Let's conduct a thorough demonstration of all options to determine the best query speed for you.

My initial query of the Qgrep Voter database from my internal drive took 8 seconds. A second query was only 2 seconds. This is because Qgrep had stored much of my data within RAM. That is very fast for querying 36 GB of original data. Every query I conducted within this Terminal session consistently took 2 seconds to complete. A Ripgrep search of this same data took 39 seconds. That is a substantial difference.

Next, let's move our Qgrep database to our external SSD. I created a new folder at the root of the SSD titled .qgrep. I moved the three Voter files from my internal drive to this external SSD. I made sure the files no longer existed internally with the following commands.

```
cd ~/Documents/Qgrep
./qgrep search Voter i bazzell
```

This resulted in an error since the files are no longer present in the place where Qgrep thinks they should be. Instead, I must specify the location of this Voter database with the following command.

```
./qgrep search /Volumes/DATA/.qgrep/Voter.cfg i bazzell
```

This specified the path to our configuration file which is now stored on our SSD. This query took 19 seconds, over twice as long as the initial query when the database was on our internal drive, but still half the time as Ripgrep. Additional queries were slightly faster, but did not make a huge difference.

Whenever I update any files within my Voter folder, I must execute the following to also update the Qgrep database since I moved the files.

```
cd ~/Documents/Qgrep
./qgrep update /Volumes/DATA/.qgrep/Voter.cfg
```

Overall, you must decide what is most important to you. Consider the following.

- If speed is your ultimate priority; you need to query specific data often; and you have the extra internal disk space, then default Qgrep is a great solution for you.
- If speed is a priority; you need to query specific data often; you do not have the extra internal disk space; and you do have extra SSD space, then Qgrep with databases stored on your SSD is a great solution for you.
- If simplicity is your priority and you can tolerate longer query times, then strict text files is better for you.

I do a mix of both. Over 90% of my queries utilize the standard script previously presented. That works fine for most needs. However, I have built a few Qgrep databases. One is for a folder of text files identifying the owners of cellular telephone numbers obtained from a huge data leak from a caller ID database provider. This folder has over 300 GB of text files, none of which are over 32 GB in size. Any time I identify new leaks or breaches which contain phone owner info, I put them in this folder on my SSD. I have created a Qgrep database using the previous commands, and update the database whenever needed. When I conduct a Qgrep query on this data, Qgrep is using a compressed version of that data which exists on my internal drive (now with 100 GB of extra data). The results take only a few seconds to arrive and my SSD does not need to be connected to my machine. I conduct the following any time I need to search a number from my internal drive database.

```
cd ~/Documents/Qgrep
./qgrep search Phone i 6185551212
```

SQLite

After the initial publication of this guide, several readers requested a tutorial for creation and maintenance of a traditional database. I have always resisted this because I am not a database expert, and I even hired someone to handle our extensive in-house database needs. However, I respect that a proper database can be the most appropriate tool when handling large amounts of data. I want to preface this section by stating that we will only dive into the basics of database creation using only one of many options, but you might see the value with a brief tutorial. I believe the best database software for a beginner is SQLite. You may already have this software of your machine, but let's make sure with the following commands.

macOS: `brew install sqlite`
Linux: `sudo apt update && sudo apt install sqlite3 -y`

SQLite is also available for Windows, and the following commands should work the same way as within macOS and Linux. Since my demonstration involves stealer log data, I will refrain from any specific Windows commands due to the high risk of virus infection. The following commands navigate to the Documents folder and creates our first SQLite database called "Logs.db" within that folder.

```
cd ~/Documents && sqlite3 Logs.db "PRAGMA journal_mode =
OFF;PRAGMA cache_size = -64000;PRAGMA synchronous = 0;"
```

You should now see "`sqlite>`" within your Terminal, which confirms we are working within our new database. Let's create a table called "Logs" and a single column called "log" with the following command. This command instructs SQLite to make sure no duplicates ever appear within this database. Make sure you see "`sqlite>`" on your Terminal line, as these commands will do nothing if pasted directly into a standard Terminal session. The following should be copied and pasted in its entirety.

```
CREATE TABLE Logs(log TEXT PRIMARY KEY,UNIQUE(log))
WITHOUT ROWID;;
```

We now have an empty database. I like to enter the following two commands which are likely already set by default, but they make sure that data headers are off (for stealer logs) and that the mode is set to list any entries one line at a time.

```
.headers off
.mode list
```

The next command creates a data separator as a character which we should never encounter within our data. This is because we only want to import one line of stealer log content into our only column, and not have things sporadically split into additional columns which may be ignored by our database. Since this demonstration will involve stealer logs which are typically separated by a colon (:), we do not want SQLite trying to split the lines by colon, which would be common with a CSV file. This could create

a huge mess when the database encounters lines which begin with "https:" or include a colon, comma, pipe, or other character within a password. The following command is a way to cheat and tell SQLite to avoid a separating character when importing data.

```
.separator "\a"
```

We are now ready to import data. Assume you have a stealer log text file titled "logs.txt" within a folder called "Logs" on your SSD. The following SQLite command imports this file into our Logs table. Since this table only has one column and we have no valid separator set, this should just import each line directly into the column.

```
.import /Volumes/DATA/Logs/logs.txt Logs
```

Depending on the size of the file, this process can take seconds or several minutes. You might see several lines which display "`INSERT failed: UNIQUE constraint failed: Logs.log`". This is a great error to see. It informs you that SQLite detected an entry which you already possessed and ignored the duplicate. This unique designation does two things. It prevents duplicates, but it also creates an index of your data. This provides faster search queries, but with some caveats which are discussed in a moment. We can now see our progress with the following.

`.databases` - This displays your database location details.
`.table` - This displays the table details.
`.indexes` - This displays the name of any indexes.
`SELECT COUNT(*) FROM Logs;` - This displays the total rows of a database.

During this writing, I imported 25 GB of stealer log text files into my new database, which took about ten minutes. Since the data was imported and then indexed for unique entries, the database was 30 GB in size. That is larger than the data itself, but that is the nature of indexed data. If you are following my tutorial, this database is stored within the Documents folder of your internal drive, so be aware of internal storage requirements. This database file can be created or moved anywhere desired, and you would only need to navigate to that directory to open it. Finally, we can query our data. The following SQLite command queries the "log" column within the "Logs" table for any entry which includes "inteltechniques" within the line.

```
SELECT log FROM Logs WHERE log LIKE '%inteltechniques%';
```

My query of the 25 GB of original data (30 GB database) took over fifteen minutes to complete from the internal drive, and the results appeared within the Terminal in real time. That seems very slow for an indexed query. This is because the previous command did not actually take advantage of the index at all because we used a wildcard (%) at the beginning and end of the query. We can use the following command to see what will happen behind the scenes for our first query without searching any data.

```
EXPLAIN QUERY PLAN SELECT log FROM Logs WHERE log LIKE
'%inteltechniques%';
```

The result should be similar to "--SCAN Logs". This tells us that SQLite will go through each line, one at a time, and display any results. Let's try again with the following command which tells us the search plan for a query for any line BEGINNING with "https://inteltechniques".

```
EXPLAIN QUERY PLAN SELECT log FROM Logs WHERE log GLOB
'https://inteltechniques*';
```

The result should be similar to the following.

```
--SEARCH Logs USING COVERING INDEX sqlite_autoindex_Logs_1
(log>? AND log<?)
```

This confirms that our fast index will be used instead of scanning all of the data one line at a time. This is because we are specifying the first part of the line and not using a wildcard. Now, let's execute the actual query with the following. Note that "*" is a wildcard to replace "%" from our previous query.

```
SELECT log FROM Logs WHERE log GLOB 'https://inteltechniques*';
```

This query was much faster and completed within one second. It displayed every line in my stealer logs which BEGINS with "https://inteltechniques". If you know exactly how your target data begins, such as a URL, this index can be the fastest option. If you are searching data which may appear randomly within a line, as will be most common for our queries, the index will not assist with the speed of your search. However, it still assists with making sure duplicate data is never imported into your database.

You may be wondering why we do not separate each field of the URL into multiple columns for better indexing. Unfortunately, all of the stealer logs we acquire will have various formats. Some will appear similar to URL:LOGIN:PASS while others will be LOGIN:PASS:URL. Some will use semicolons, pipes, or other separators. Many will be formatted as USER@PASS:URL. **There is no standard**. Therefore, creating a properly separated and fully indexed database will take a lot of time, and will never be perfect. You could create multiple columns with unique indexing and import all data with a separator, but I think you will be displeased with the size of your database and the errors encountered due to poor formatting of the data. This would also delete duplicate email addresses for various sites. This is why my demonstration only focuses on a single column of data. You may be able to justify the extra effort, but the exact steps to configure your own perfectly-indexed custom database exceed our scope. While testing this data, I downloaded new stealer log files from Telegram. I conducted the "Daily Task" as previously explained, and then opened my database and imported the "new" data with the following single command.

```
sqlite3 Logs.db "PRAGMA journal_mode = OFF;PRAGMA
cache_size = -64000;PRAGMA synchronous = 0;" ".headers
off" ".mode list" ".separator \a" ".import
~/Downloads/Logs/logs.txt Logs
```

I immediately observed that most of this data was already present within my SQLite database. Since I configured this column to only accept unique data, I never need to "sort" for unique entries. Every time I import data, only the unique entries are added. In my experience, importing unique entries into SQLite is more accurate than sorting for unique entries with the Sort command. Once your database becomes substantially larger than the content, you may want to optimize the data with the VACUUM command. This will attempt to optimize and shrink the size of the database. While working within your SQLite database in Terminal, execute the following command.

```
VACUUM;
```

Note that you must have at least the size of your database available as free space. Results will vary. My 30 GB test database only reduced to 29 GB, but it was fairly new. A 250 GB database that has seen hundreds of imports might benefit more. You can always export a current copy of the text data with the following SQLite command from the folder of your database. This can be valuable if you are using SQLite to import non-duplicate stealer logs in order to keep the data to a minimum, but also want a non-database backup of the data. I can state from experience that databases eventually become corrupt. Having a backup of the database is great, but having a text-only backup of the raw data is even better. If you rely on SQLite databases, make sure you have updated and archived data.

```
sqlite3 -header -csv Logs.db "select * from Logs;" >
~/Desktop/Logs.csv
```

For most readers, I think Qgrep and SQLite are overkill. If you have a specific data folder which you query many times every day and you can no longer stomach the wait times, then you should test these strategies. For those who conduct an occasional daily query, the previous search script is a much simpler solution. If you need to query terms which could appear anywhere within your data, Qgrep is faster than Ripgrep, and Ripgrep is faster than SQLite. If your query appears at the beginning of an entry, SQLite is fastest, followed by Qgrep then Ripgrep. If your priority is to eliminate redundant data, SQLite is the best daily solution. Remember that Qgrep will only ingest files which are smaller than the available RAM, but SQLite should be able to import any size. If you want to create a Qgrep database out of a 100 GB Stealer Logs file, you will need to use the Split command to create multiple smaller files. You could then search the entire data set in seconds, but could no longer merge these smaller files with new data to eliminate duplicates with the Sort Merge feature.

If you collect stealer logs every day and you want an easier way to import them into your collection without duplicate entries, a SQLite database is great. You may find that the space required to store an indexed database is too much more than that required for the original data alone. Ripgrep may be better for your needs. Only you can decide which route is best for your data. I prefer the simplicity and speed of Qgrep with the immediate delivery of keyword queries, but respect the ability to remove duplicates while importing into SQLite. We have only briefly discussed SQLite databases. Tweaking your configuration, indexing, compression, and queries can open even more doors. If this is of interest to you, I highly recommend researching SQLite further.

Duplicate Files

If you collect enough breach data, you will likely find many duplicate files with different names. This could be due to poor file naming or deliberate trickery in order to distribute inaccurate data. Regardless, I find it beneficial to eliminate duplicate files frequently. The following commands install fdupes.

macOS: `brew install fdupes`
Linux: `sudo apt-get install fdupes`

The next command launches fdupes, scans recursive files within the Downloads directory, and prompts you to select which duplicate file to keep.

```
fdupes -r -d ~/Downloads
```

I placed two identical files with different names in the same directory, the following was the result. Entering "1" would keep the first file, while "2" would preserve the second.

```
Set 1 of 1:
  1 [ ] /Users/mbp/Downloads/1/Real Breach.txt
  2 [ ] /Users/mbp/Downloads/1/Fake Breach.txt
( Preserve files [1 - 2, all, help] ):
```

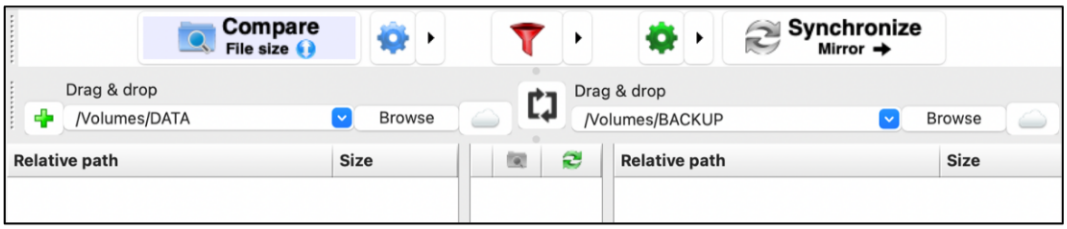
Backups

You have worked hard to collect, clean, and organize your data. However, storage devices crash all of the time without warning. I would be devastated if I lost my collection. Therefore, a good backup system should be in place. You should have already installed FreeFileSync, and we can use it to easily copy our data.

Until now, we have been copying our data to an external SSD labeled DATA. You should now insert your backup drive, and make sure it is labeled BACKUP. In the previous hardware chapter, I recommended a cheap 4 TB external USB drive with a small spinning disk inside. Since this is only used as a backup of our data, speed is not a priority.
Follow the previous tutorials to encrypt this new external backup drive, then conduct your first backup within FreeFileSync.

- Close any pop-up windows and click "Browse" in the left "Drag & drop" area.
- Select your root folder of your SSD in the left menu.
- Click the "Open" button and click "Browse" in the right "Drag & drop" area.
- Select your root folder of your backup drive in the right menu.
- Click the right arrow icon next to the green cog wheel near "Synchronize".
- Change the option to "Mirror".

The mirror option makes sure that the data on the SSD is always an exact replica of the content on your backup drive. If your drives match the labeling of mine, yours should look similar to the image below. You could save this configuration with the "Save as" icon, naming it "Data Backup".



Next, click "Compare" and allow the analysis. You may receive warnings that FreeFileSync is trying to access folders, but this is acceptable for this purpose. You may also receive a warning about an area which is inaccessible to the program. I click "Ignore All" when this happens. Once complete, you should see a summary of all files which will be synchronized. Clicking the "Synchronize" button begins the backup process, which can take some time on the first run.

The next time you need to back up your data, you would connect your drives; unlock the encryption by entering the passwords; open FreeFileSync; and select the "Compare" button again. This time, you should only be presented the files which have been modified since the last backup. Then, the synchronization process should be much faster. I do this weekly.

You now have ONE backup. To some, that is plenty. To me, that is risky. What if my house explodes with both drives inside? What if I accidently erase data from my DATA drive, and then synchronize that to my BACKUP drive? Most computer professionals will tell you that you need three copies of any important data, and I agree. This is why I keep one off-site backup in addition to the previous backup we just made. I have another external USB drive which has a replica of all data stored at another trusted home. When I visit monthly, I take my laptop and SSD in order to update the second backup drive off-site. It might not always be up-to-date, but the majority of my data collection is present in the case of catastrophe.

Putting It All Together

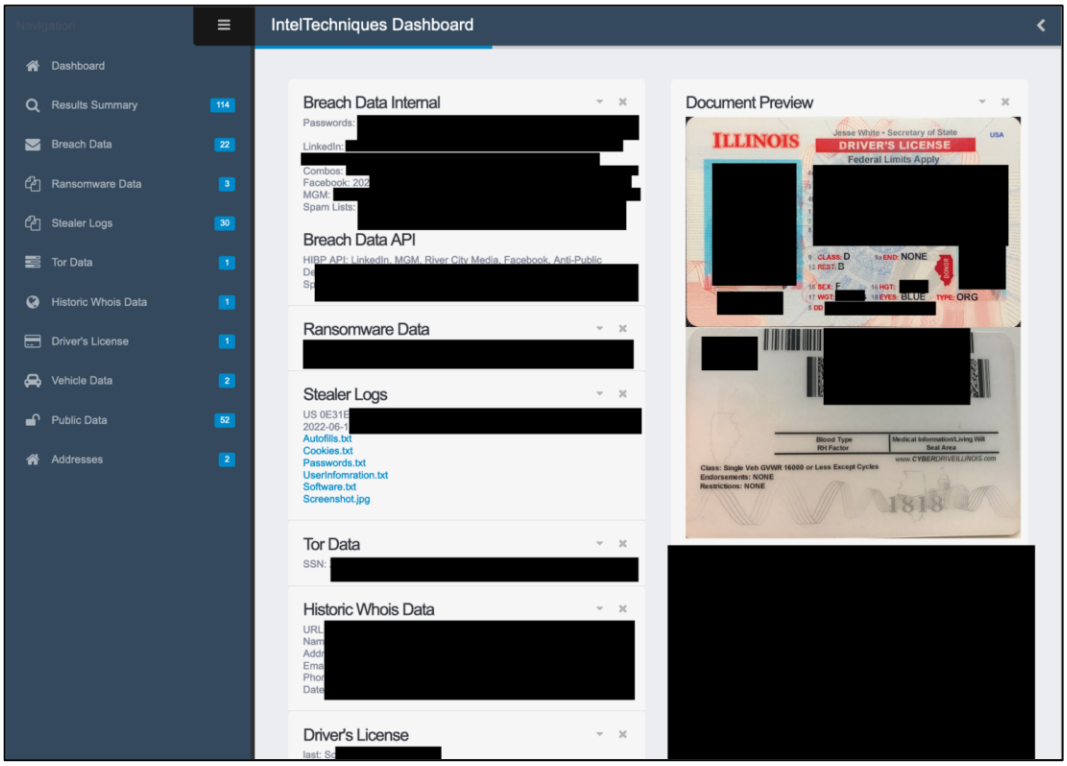
I am simplifying the explanation of these steps, but you have everything you need to conduct your own research into its capabilities. Possessing an efficient way to query your data will increase the likelihood that you will take advantage of these techniques. During my testing, I could use the script to query the entire COMB data within a few seconds; display file names from thousands of breaches immediately; or focus my queries within generic data types or specific files in less than a minute, all from an external drive. Consider the following typical usage.

- **Internal Investigations**: This one is obvious. This type of data can be crucial to investigations. It can immediately uncover real and alias information. Every day, breach data reveals the true person behind a burner account due to sloppy OPSEC.
- **Client Vetting**: We take our clients' privacy very seriously. We would never use a third-party system to properly verify the identity of a potential client. Instead, we use our own in-house system to make sure we know who we are dealing with. At least twice, a potential client's record in our system revealed suspicious activity related to criminal prosecution, identifying the reason they were asking about anonymous relocation services (we never assist in those situations).
- **Client Exposure**: We have many clients who keep us on a retainer. Part of that service is an initial analysis of exposed information, and a constant monitoring for any new details. Every week, we reach out to clients to let them know of a new breach, ransomware attack, or password log which may impact them or their company. Just a few months ago, I was able to alert a close friend that his daughter's full name, address, DOB, SSN, and banking details were being passed around a criminal marketplace focused on identity theft. We received the internal alert within two days of the initial exposure.

Breach data is nothing new to investigators. I have explained how we ingest typical text-based breach data since the 7th edition of *Open Source Intelligence Techniques*. The new world of daily ransomware dumps and stealer logs changes everything. We are collecting this data in masses never seen before. We often bring in over 2 terabytes of new stolen content weekly, which we parse down to an average of 50 GB of useful data for the week. Our current collection is over 40 TB. We focus mostly on text and private documents, and try to eliminate all public docs, company materials, and anything else which does not have an immediate impact on an individual.

The following figure displays a heavily redacted screen capture of my company's internal investigation portal, which was created specifically for our needs by a database expert. In this example, I searched only the email address of my target. Since this address appears within multiple breaches, it connects me to her real name (from the MGM and LinkedIn breaches). From there, the system cross-references that name to her driver's license. This license was scanned by a mortgage company, which was later hit with a ransomware attack, which resulted in all of their data being published on an onion site. Our system scanned the characters within the scanned image but also the

barcode, as explained in the previous text. Now that our system has enough true data about our target, it can cross-reference everything within our collection of breach data, including ransomware dumps, Tor content, and stealer logs. From there it determines her home address, which is cross-referenced with people search data, historic Whois data, vehicle data, and numerous public APIs. The result is an immediate view of all available public, private, and stolen data. Our team relies heavily on this portal, and each query takes approximately 5-10 seconds. It cross-references any data we have until all options have been exhausted, as encouraged throughout this book.



I want to close this chapter with some final warnings and reminders. Forgive me if this seems redundant, but we can all use another mention of the risks and concerns. Leaks, breaches, logs, and ransomware always consist of data which was either stolen or never meant to become publicly available. It is abused every day by professional and amateur criminals. Many readers will be disgusted by these chapters and upset that I share details on the acquisition of this content. When collected and used properly, I believe it can be beneficial to both investigators and those who monitor this activity from a defense perspective.

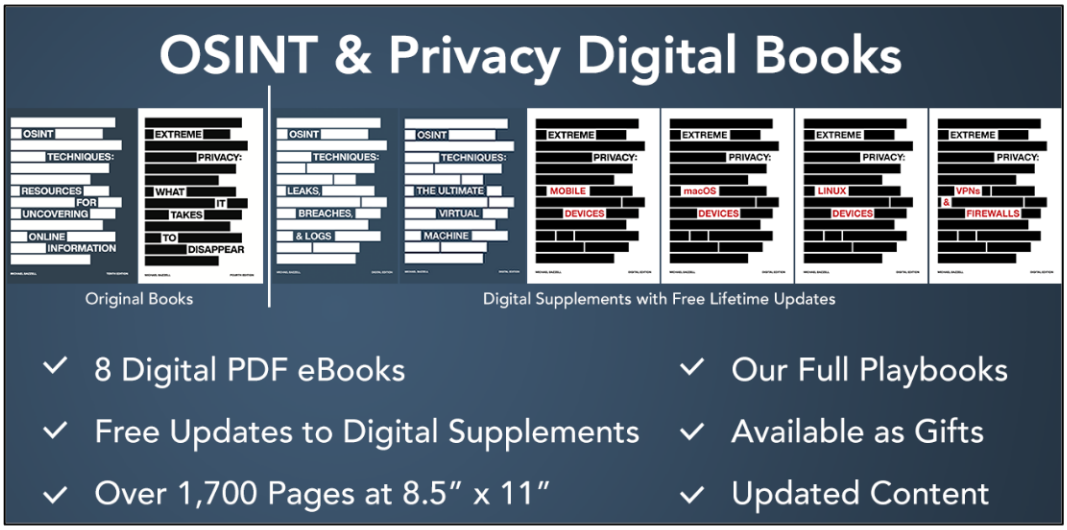
CONCLUSION

We are only beginning the journey into data collection. I hope I have sparked your interest for this type of investigation technique, and I challenge you to see what else you can find. If you are an early adopter of this guide, you should have all of the basics to get started. As time passes, I hope to continuously update these pages with new techniques, data sources, and other ideas. When this document should need updated, all modifications are completely free. If you purchased this PDF through my website, you will be notified via email when revisions can be downloaded. If you downloaded an unauthorized copy from a piracy website, please consider purchasing a legitimate copy. Your $20 purchase supports the research which goes into creating and updating these guides. Finally, if you find anything which needs updated or corrected, please email us at books@inteltechniques.com. My staff cannot respond to emails directly, but they will monitor them for any changes which we need to apply to the next version of this guide.

Thank you for the continued interest in OSINT.

~MB
IntelTechniques.com

Other Books by Michael Bazzell



OSINT Techniques, 10th Edition (2023): 36 chapters | 260,000 words | 550 pages | 8.5" x 11" | $30 - This textbook will serve as a reference guide for anyone who is responsible for the collection of online content. It is written in a hands-on style that encourages the reader to execute the tutorials while reading. The search techniques offered will inspire researchers to think outside the box when scouring the internet. Digital downloads include offline search tools, custom Linux scripts, and detailed report templates.

Extreme Privacy, 4th Edition (2022): 22 chapters | 320,000 words | 517 pages | 8.5" x 11" | $30 - This rewritten privacy manual is PROACTIVE. It is about starting over. It is the complete guide that I would give to any new client in an extreme situation. It leaves nothing out and provides explicit details of every step I take to make someone completely disappear, including legal documents and a chronological order of events. The information shared in this book is based on real experiences with my actual clients, and is unlike any content released in my other publications.

OSINT Techniques, Leaks, Breaches, & Logs (2024): 9 chapters | 57,000 words | 171 pages | 8.5" x 11" | $20 - This digital (PDF) supplement to ***OSINT Techniques, 10th Edition (2023)*** delivers a much more thorough guide about data Leaks, Breaches, & Logs. It provides our entire playbook which we use to locate, acquire, clean, store, and query various online data collections valuable to our investigations. All expired and outdated methods were replaced with new techniques, and brand-new topics were introduced throughout. We also explain all daily, weekly, and monthly tasks required to maintain your data collection. All updates are free and delivered digitally.

OSINT Techniques, The Ultimate Virtual Machine (2024): 14 chapters | 74,000 words | 231 pages | 8.5" x 11" | $20 - This digital (PDF) supplement to ***OSINT Techniques, 10th Edition (2023)*** delivers a much more thorough guide about Linux OSINT applications and Virtual Machines. It provides our entire playbook which we use to build, configure, clone, export, backup, and wipe out our investigative environments. All expired and outdated methods and applications were replaced with new functioning techniques, and brand-new programs were introduced throughout. For this release, we transition from Ubuntu to Debian

for a more private and stable environment, and also explain all tasks required to maintain your investigative machines. New downloadable scripts automate the entire installation process and all data search programs. All updates are free and delivered digitally.

Extreme Privacy, Mobile Devices (2024): 16 chapters | 73,000 words | 173 pages | 8.5" x 11" | $20 - This digital (PDF) supplement to ***Extreme Privacy, 4th Edition (2022)*** delivers a much more thorough guide about mobile devices. It provides our entire playbook which we use for our clients when we need to acquire new hardware, configure a custom operating system, execute proper DNS filtering, enable push services, install applications, obtain anonymous cellular service, establish VoIP connectivity, program redundant data eSIMs, provide secure communications, apply VPN strategies, and troubleshoot the things which will go wrong. We also explain all maintenance and best practices for a new private and secure device. All updates are free and delivered digitally.

Extreme Privacy, macOS Devices (2024): 10 chapters | 48,000 words | 131 pages | 8.5" x 11" | $20 - This digital (PDF) supplement to ***Extreme Privacy, 4th Edition (2022)*** delivers a much more thorough guide about macOS devices. It provides our entire playbook which we use for our clients when we need to sanitize previous Apple IDs; acquire new hardware; configure operating system settings; execute a proper firewall; install applications without Apple ID; configure browsers, VPNs, and DNS; establish VoIP connectivity; create virtual machines; and generate custom scripts for daily usage. We also explain all maintenance and best practices for a new private and secure macOS device. Purchase includes custom macOS scripts and an import file to replicate all firewall rules. All updates are free and delivered digitally.

Extreme Privacy, Linux Devices (2024): 10 chapters | 47,000 words | 124 pages | 8.5" x 11" | $20 - This digital (PDF) supplement to ***Extreme Privacy, 4th Edition (2022)*** delivers a much more thorough guide about Linux devices. It provides our entire playbook which we use for our clients when we need to acquire new hardware; configure operating system settings; execute proper DNS filtering; install applications securely; configure browsers and VPNs; establish VoIP connectivity; create virtual machines; and generate custom scripts for daily usage. We also explain all maintenance and best practices for a new private and secure Linux device. Purchase includes custom Linux scripts. All updates are free and delivered digitally.

Extreme Privacy, VPNs & Firewalls (2024): 9 chapters | 38,000 words | 100 pages | 8.5" x 11" | $20 - This digital (PDF) supplement to ***Extreme Privacy, 4th Edition (2022)*** delivers a much more thorough guide about VPNs and firewalls. It provides our entire playbook which we use for our clients when we need to acquire new hardware; configure firewall settings; execute proper DNS filtering; configure web browsers; and establish VPN connectivity. We also explain all maintenance and best practices for a new private and secure firewall device. Purchase includes custom firewall configuration files. All updates are free and delivered digitally.