



**Internews**

Local voices. Global change.

# **Field Guide to incident response for civil society and media**

Internews' Internet Freedom and Resilience team, 2023

<b>Acknowledgments</b>	6
<b>Chapter 1: Introduction</b>	7
Trust, knowledge, and experience	7
How to read this guide?	8
Future plans for this guide	8
<b>Chapter 2: Prerequisites and Required Supplies</b>	9
<b>Chapter 3: Reading threat research</b>	11
Sources	12
Contacting authors	13
<b>Chapter 4: the Linux command line</b>	14
Getting help on Linux commands	19
Who's the boss? Sudo	19
Editing files	20
<b>Chapter 5: Malware</b>	22
Computer programs, malware and anti-analysis	22
Different kinds of Malware	23
Persistence	25
Infection chains	25
Decoy	26
Command and Control	26
Antivirus	27
Vulnerabilities and exploits	28
<b>Chapter 6: Virtual Machines and REMnux</b>	30
Virtual Machines and snapshots	30
REMnux	31
Running REMnux for the first time	31
Handling malicious files	33
Turning off REMnux	34
REMnux on macOS M1 or M2	35
<b>Chapter 7: Threat Intelligence and VirusTotal</b>	36
VirusTotal	36
Important concepts for threat intelligence	37
Indicators of Compromise	37
IP addresses	37
Domain names, hostnames and DNS	38
Defanging	40
Hashes	40
Using VirusTotal	42
File checks and uploads	42

The Detection tab	43
The Details tab	44
The Relations tab	44
The Behavior and Community tabs	45
Other file types	45
Hostnames and domains	46
IP addresses	46
URLs	46
Threat hunting	46
Targeted and non-targeted threats	47
Sinkholes	48
Other tools	49
<b>Chapter 8: Android and Android malware</b>	50
Android versions and a note on Google	50
How likely is Android malware?	50
Android apps and permissions	51
A better approach: looking at app permissions	52
Location	52
Microphone	53
SMS	54
Contacts	54
Files	54
Accessibility	55
Notifications	55
Analyzing Android malware and advanced Android malware	55
<b>Chapter 9: Email forensics</b>	57
SMTP	57
POP3 and IMAP	59
Email headers	60
Obtaining email headers and body	61
Two kinds of From addresses	62
The sending IP address	63
The email body	64
Base64 and quoted-printable	64
Using emldump.py to view MIME parts	65
Embedded and external images	67
Forwarding emails for analysis	67
Authentication protocols: SPF, DKIM and DMARC	68
SPF	68
DKIM	68
DMARC	70
<b>Chapter 10: Analyzing email payloads</b>	71



Sandboxes	71
Manually analyzing attachments	76
Malicious Office documents	77
PDF attachments	81
Other kinds of attachments	83
Analyzing links	83
Vulnerable email clients	84
<b>Chapter 11: Website incident response</b>	<b>85</b>
Background, terminology and techniques	85
Websites and web servers	86
HTML	86
HTTP	86
HTTPS	88
Query strings and URL encoding	88
Headers	89
IP address	89
HTTP responses and status codes	90
Cookies	91
Dynamic content: PHP and JavaScript	91
Accessing a web server	93
The find command	95
Understanding the website's context	95
Web server logs	96
Access logs	96
Error logs	98
Logs and privacy	98
Redirects	98
Rewrites	99
.htaccess	99
DDoS attacks	100
Is it a DDoS attack, or is the site just very popular?	101
DDoS mitigation	101
Content management systems	102
General	102
WordPress	103
Joomla, Drupal and Magento	104
Server issues	105
Cleaning up an infected website	105
1. Disable the website	105
2. Fix vulnerability.	105
3. Restore from backup or remove malicious content	105
4. Re-enable website and verify it works	106



5. Harden the website	106
6. Keep checking	106
Hardening websites	106
Web application firewalls	106
Security Headers	107
Added logging	107
Backups	107
Static website	107
<b>Chapter 12: iOS incident response</b>	109
iOS versions	109
Apps and app permissions	110
The App Store	111
Third-party app stores in the future	111
Apple ID and iCloud	112
Mobile Device Management	112
Jailbreaking	113
Safety Check	113
Advanced spyware	113
Coordinating with Analysis Experts	114
Triaging	115
Trust	115
MVT	116
iTunes backups	117
iMazing and iVerify	118
<b>Appendix: Answers to questions</b>	119
Chapter 4	119
Chapter 6	119
Chapter 7	119
Chapter 8	121
Chapter 9	121
Chapter 10	122
Chapter 11	123
Chapter 12	123



## Acknowledgments

This guide has been written with the help of many others. This work would not have been possible without the help of seven Threat Labs: Conexo, CyberHUB-AM, Digital Security Lab Ukraine, Jordan Open Source Association, MariaLab, SHARE Foundation and SocialTIC. Since 2021, Internews has worked with these Threat Labs to ensure they had the appropriate skills and tools to analyze suspicious phishing and malware samples and share information with the community regarding attack trends, emerging threats, and countermeasures.

My regular discussions with them were vital in understanding this work and the target audience. My Internews colleagues Marc, Ashley, Khin, Andrew, Carlos, Łukasz and Neil provided great feedback and many suggestions. Gus was kind enough to read through several of the chapters and left valuable comments and suggestions. Many others were helpful, often in implicit ways, by sharing knowledge and experience in this fascinating field.

All mistakes, however, are my own.

Martijn Grooten

Last Updated October 2023

This content is available as a [CC-BY 4.0 International license](#).



# Chapter 1: Introduction

This guide consists of a series of interconnected lessons that will help you as a reader become familiar with and practice using tools and techniques required for incident response. The focus is on the digital threats facing civil society and media, particularly phishing, malware, and device compromise.

The original target audience of the guide was a group of seven 'Threat Labs,' part of an Internews project from 2021 to 2023. These are civil society organizations with experience in handling security incidents that support their communities by responding to digital threats and who are often a point of contact for the broader digital security and cybersecurity communities. Internews worked with these partners to ensure they had the appropriate analysis skills, tools and resources.

This edition of the guide is written for organizations or individuals already providing some level of digital security support to civil society and the media but seeking to build their incident response and analysis capacity further. In principle, the guide could be for anyone performing incident response in any context. The guide may also help existing Threat Labs onboard new staff members. However, three main aspects make this guide unique for this specific audience:

1. **The focus is on the kinds of incidents civil society and the media face.** You will not learn how to analyze digital threats that revolve around Microsoft Active Directory or that target a network through a vulnerable or weakly protected device at the edge of an organization's network: such set-ups are very rarely seen by civil society organizations, given the small size of most of them. In general, such threats are prevalent in cybersecurity and lead to incidents such as ransomware.
2. **There is an intentional acknowledgment of the trauma faced by the people facing these kinds of threats.** The goal of providing incident response isn't only to analyze the incident and 'clean up' affected systems but also to support those targeted. Once you start dealing with actual incidents, you'll realize that in surprisingly many cases, this involves comforting the target that an incident was not as bad as it seems and, in many cases, what someone believed was an incident was, in fact, normal behavior.
3. **This guide assumes explicitly the reader is doing incident response only as a small part of their work.** While some of these readers may aspire to learn cybersecurity skills far beyond what is included in this guide and maybe even aspire to a career in the field, what is included here is meant to be "just enough" for conducting aspects of this work in a more limited capacity.

## Trust, knowledge, and experience

Three things are essential when providing incident response for civil society and the media: Trust, knowledge, and experience.

**Trust** is important in any relationship, especially when working with at-risk groups and people. Building and maintaining trust is beyond the scope of this guide, but it will take time and effort and is rarely, if ever, based on technical skills.

These technical skills, or **knowledge**, are also critical. This guide teaches you many skills that should help you handle most incidents.

However, each incident is different. Therefore, the more incidents you've handled, the easier it will be to handle future incidents. You gain this **experience** by doing the work, though this guide also includes many exercises to "get your hands dirty" and gain extra experience in a safe environment. Another way to become

more experienced and knowledgeable is to read analyses of digital threats. A later section of this introduction will guide you on how to read other analyses.

## How to read this guide?

Internews strongly recommends that you consider this guide a self-study project, moving through at your own pace and in your own time. Each chapter includes several questions, the goal of which is not to test your knowledge but to guide you into thinking with an analyst's mind. Readers can find suggested answers to the questions in the appendix at the end of the guide.

Each chapter also includes several exercises meant to be a safe space to build your experience and practice various aspects of incident response. You are strongly encouraged to do the exercises. Some exercises are marked as optional, either because they could take too much time or because they require some device you may not have. If possible, you are encouraged to do these exercises too.

The guide's primary goal is to better prepare you as a practitioner to carry out incident response regarding digital threats targeting civil society and the media, particularly malware, phishing, and device hacks. A secondary goal is to help you analyze the threats you're handling. This analysis will allow you — and the community you support — to better understand these threats, which could lead to better digital security mitigations and stronger advocacy efforts informed by data.

One important thing to remember is that every digital threat is different, and there are rarely uniform steps you can routinely follow to analyze a particular threat.

When there are simple steps to analyze a particular threat, these are picked up by security software (such as antivirus) and sometimes operating systems themselves to neutralize them effectively. This, in turn, leads to threat actors changing their tools or tactics to bypass these protections.

Therefore, Internews recommends that aside from this guide, you also keep an eye on the threat landscape, for example, by reading threat intelligence analyses. Chapter 3 focuses on this.

## Future plans for this guide

Internews is seeking additional support to continue expanding this guide. With additional funding, we would like to write sections on Windows, macOS and networks. Further support would allow Internews to extend much of the existing content. If you would like to support Internews in these efforts, please reach out directly at [mgrooten@internews.eu](mailto:mgrooten@internews.eu).





## Chapter 2: Prerequisites and Required Supplies

To make this guide as accessible as possible, little **technical knowledge or experience is** required. You must be comfortable using a computer: running programs, installing new ones, and changing settings.

As for **hardware**, you will need at least a laptop or desktop computer. You can use the same computer that you use for work purposes, as you will perform analyses in virtual machines (more on this in Chapter 6).

The computer you use can run Windows, Linux or macOS. Although Chromebooks have many benefits, especially for this community, they are less suitable for the kind of analysis you will be doing. If you have yet to buy a computer and are considering macOS, make sure you have read Chapter 6 on REMnux, a tool you will be installing that does not work on macOS computers with an M1 or M2 chip. There are workarounds, though, and macOS is generally a very good operating system.

You will install VirtualBox on your computer to run virtual machines, and it is a good idea to look at the [requirements](#) for running that. The most essential requirement, besides “reasonably powerful” hardware<sup>1</sup> (a reasonably modern laptop will likely do), is disk space. You will want ideally 100GB or more and 4GB in RAM, though more of both is better if you have the option.

Aside from that, any other devices you have lying around, especially mobile phones, will be helpful, especially if you want to test things. The guide will not ask you to do anything risky on the devices, so using your everyday phone is fine.

Another requirement, at least as important as the hardware, is **understanding how civil society and media work** in the parts of the world where you will be providing incident support. You must understand what threats they face (including non-digital threats, such as arrests), how the rest of society perceives their work, common practices, etc.

This guide does not focus on understanding the local context. This guide is designed for individuals who already have this understanding and have established trust with the communities they seek to support.

Therefore, now is a good time to step back and ask yourself: [am I the right person](#) [wayback machine?](#) It is okay to decide you are not.

Patience is another requirement for incident response, especially for working with at-risk people and organizations.

Often, you will find investigations drag on for a long time due to slow responses. Those impacted by threats often juggle many priorities, impacting their ability to engage with you on the incident response and investigation. Sometimes, people stop responding while you are still working to investigate their incident. That can be frustrating. It is also the reality of doing this work. Try not to get demotivated, and look after your mental health, just like you would (and should) consider the mental health of those you support.

---

<sup>1</sup> The X86 mentioned by VirtualBox in the hardware requirements refers to the type of processor. Most modern desktop and laptop computers run X86 hardware, but macOS with M1 or M2 chips are the exception. Now you know why these were singled out in the previous paragraph.



The final requirement is an understanding of the **mental health** impact of and the **trauma** linked to digital threats (also referred to as their **psychosocial** impact). Digital threats can be very traumatizing, frequently having a lasting impact. Some are traumatized by past experiences, leading them to believe they are the target of additional or continued digital threats, even if that is no longer the case.

This is one of the most complex parts of providing incident response to at-risk people. It is challenging to balance taking someone's concerns seriously and letting them know there is no technical or other evidence that a particular threat has occurred. It is important to understand that this role goes beyond technical support, even if this is why someone initially called you to look at a case.

It is important to share your challenges with team members and other people within the community, leaving out confidential case details where needed. Sharing the workload can distribute the emotional and mental burden and ensure you have the support network you need to effectively and ethically continue this work.



## Chapter 3: Reading threat research

As mentioned in the Introduction, experience is crucial for any incident response work. In addition to learning by doing the work, you can also learn by reading cybersecurity articles and reports, of which there are plenty, published by security companies, government organizations, NGOs, and individual researchers.

Reading such articles can teach you how to analyze while also helping you recognize the characteristics of certain threats, which can speed up your analysis when you come across them in the future. Moreover, sometimes, a threat might be relevant to your community, and you can use the information from an article to give tailored security advice.

The number of such articles can be rather overwhelming, though, and their quality is mixed at best. This chapter provides tips on how to read them and where to find good ones. Though the focus is on written articles, most of this also applies to video analyses, for example, conference videos, of which thousands can be found on YouTube and other platforms.

Many of these articles are full of technical language and can thus be quite daunting to read. This is certainly true if you are new to the field, but it is still largely true for people with years of experience in cybersecurity. Do not let this discourage you, and keep the following in mind.

Many articles are more technical than they need to be. Authors with certain technical skills sometimes struggle to write for a more general audience and can find translating technical concepts into lay terms challenging. Cybersecurity articles are often full of code for that reason.

Additionally, many articles written by security companies are interested in making threats sound very bad as part of their branding and marketing efforts. It's rare for articles to include factually incorrect information, but claims may be implicitly or explicitly exaggerated. Unfortunately, such claims are often further spread by media – including security media – when they report on such threats. So, keeping possible exaggerated claims in mind and trying not to get overwhelmed by technical articles that are hard to understand, you are ready to start reading articles.

If you are reading these articles in a professional capacity, here are some suggested questions to help inform your review. Such questions are especially helpful if the article is full of technical content you don't yet have the skills to understand. Once you get familiar with security articles, you'll learn how to answer these questions quickly, often by skimming the article and focusing on the introduction and the conclusion.

**Can I reproduce the analysis?** If you read articles to improve your analysis skills, it can be helpful to see if you can reproduce the analysis. That's certainly not always the case, for example, if the analysis depends on artifacts that you don't have access to (think of malware samples that aren't publicly available or network traffic that can't be reproduced) or the tools required are beyond your skills set or require paid subscriptions. But in quite a few cases, you can follow along and use the analysis as an exercise for which you already have the answers.

**Is this directly relevant to my community?** Sometimes, your community is directly or indirectly mentioned in an article: a threat might target a country you work in or a civil society organization in that country. That's



probably quite rare, but that makes this threat particularly relevant and is a good reason to read closely – and maybe share the article further.

**Could the method described also be used against my community?** Perhaps the article is about malware targeting online banking users in Western Europe, which isn't all that relevant to you, but it used a certain kind of email attachment that an adversary could also use against your community. Understanding this threat can thus still be helpful. Threat actors around the world copy each other's tactics.

If none of these three questions have an affirmative answer and you're short in time, you may just want to stop reading the article!

**How does this threat start?** In particular, if it is malware, how do people get infected? Unfortunately, this is often left out of articles for the simple reason that this is often unknown. Much research starts with a malware sample found somewhere and then focuses on what it does. That can be interesting – though technical details of how it achieves things like stealing data from a device often are only relevant to other analysts – but it's far more interesting to learn how it got there in the first place. That is what is most actionable for the community.

Closely related to this question is: **how can this be prevented?** Often, there is a fairly simple way to do this, but it's not always clearly stated in the article.

**How can this be detected?** How can you find this particular threat on someone's device, account or network? Threats often leave certain artifacts, usually on the device, but sometimes in public. This is very helpful if you want to support a community that may be affected by the threat.

## Sources

Very few sites specialize in analyzing threats targeting civil society in particular, but you will want to follow [Citizen Lab](#). You'll also find the occasional technical analysis published by [Amnesty Tech](#), [Access Now](#), [Qurium](#) or [Equalit.ie](#).

Some technical blogs for general analysis are worth following. The '[diaries](#)' from the SANS Internet Storm Center are well worth reading for their many short analyses by several analysts, including Didier Stevens, whose tools we will feature in later chapters. SANS also publishes a short [daily podcast](#) with security news and advice.

The [blog](#) from malware analyst Tony Lambert is very useful because he takes you through the analysis process and usually uses tools available in REMnux to do so. The [blog](#) from security company Intezer includes many introductions to threat analysis topics by Nicole Fishbein. Website security company Sucuri has a [blog](#) with many good articles on website security.

[Malware Traffic Analysis](#) is a blog by Brad Duncan, who regularly posts network traffic captures from malware infections. And [Objective See](#) is a very well-written blog by Patrick Wardle focusing on macOS malware analysis in which he takes the reader through the analysis process.

Some companies with particularly good research blogs that sometimes feature threats relevant to civil society are [ESET](#), [Recorded Future](#), [Lookout](#) (which focuses on mobile threats) and Google's [Threat Analysis Group](#).

Other companies that regularly publish threat analysis articles are, in no particular order, [Mandiant](#), [Malwarebytes](#), [Sophos](#), [Kaspersky](#), [Volexity](#), [Avast](#), [Palo Alto Networks](#), [BlackBerry](#), [CrowdStrike](#), [Red](#)



[Canary](#), [Sekoia](#), [G Data](#), [Proofpoint](#), [Sentinel One](#), [Cisco Talos](#), [Trend Micro](#), [Check Point](#). This list is by no means exhaustive! You may also want to watch the alerts and advisories from [CISA](#), the United States government cybersecurity & infrastructure security agency.

You may wish to follow sites like [Bleeping Computer](#) and [The Record](#), both of which write news stories about threat research; you can always go to the original source to dive deeper into a particularly interesting or relevant analysis. The thrice-weekly newsletter from [Risky Business](#) (and the accompanying podcast) is another great resource for those who don't have time to follow the individual blogs.

News sites such as [Wired](#), [TechCrunch](#) and [Ars Technica](#) also cover cybersecurity and often include their own research.

## **Contacting authors**

If you have questions about a particular article or report, we encourage you to reach out to the author (through email if you can find it; otherwise, through social media; LinkedIn is ideal for it).

Not only does this likely result in an answer to your specific question, but it will also potentially expand your cybersecurity network. The engagement will benefit the authors, as they learn something about their audience and may use your questions to be clearer in future articles.



## Chapter 4: The Linux command line

If you imagine someone responding to a security incident, you may picture someone wearing a hoodie typing commands into a black terminal. That is a misconception — and not just the hoodie part: a lot of incident response does not involve the terminal.

There are many things you can do on the operating system itself or the Internet, and this guide favors these options, as they make analysis more accessible, if not to you, then to the people you support. It is always preferable to do things that the people you support can understand — and, in case you're providing support remotely, do themselves with your guidance.

That said, if you regularly handle incidents, you will encounter cases where you must use the command line. This guide at various places assumes you have some familiarity with it; however, this chapter is meant for those who don't. You may skip this chapter if you are already familiar with the command line.

There are many introductions to the Linux command line, both on the Internet and in books (such as [this one](#))<sup>2</sup>. This chapter offers a path for learning through trial and error (and a lot of Internet searches).

It is not meant as a complete introduction or even a list of the most used commands. It simply guides you through some exercises and questions so that you understand the basics and feel comfortable enough to use it and gradually learn more commands and tricks.

One more thing: The Unix shell is the formal way to refer to the Linux command line. There are several kinds of shells, such as **bash**, which is probably the most popular, and **zsh**, which macOS uses. They all work slightly differently, enough to notice once you become more familiar with one shell but not enough to bother with when you start. This guide refers to any terminal as 'Linux,' which isn't strictly correct either. This chapter was written using bash.

**Exercise 4.1.** Log into a shell to obtain the Linux command line. On a Linux or macOS machine this is as simple as opening the program called 'terminal' (or maybe 'xterm'). On Windows you can install Windows Subsystem for Linux using the [instructions](#) provided by Microsoft while Google [explains](#) how to set up Linux on a Chromebook.

Alternatively, you can wait until Chapter 6 (which will walk you through the installation of REMnux) and then return to this chapter. You won't need to use the command line until then.

**Exercise 4.2.** On the command line, enter the command `ls` (type it and click enter). You will see a list of files. Now enter the command `ls -a` and then the commands `ls --all`, `ls -al` and `ls -la`, one after the other.

---

<sup>2</sup> [The Linux Command Line](#) by William Shotts is as good as any, but I would also consider the beautifully handwritten [zines](#) by Julia Evans



The `ls` command lists all the files and directories in the current directory (some may refer to as a folder). The **option**<sup>3</sup> `-a` includes the hidden files, which in Linux have a name that starts with a dot.

You'll note that the command `ls --all` provides the same output. Indeed, `--all` is an alias for `-a`: if an option has more than one letter, it is usually preceded by two hyphens. Commonly, there are short and long versions of the same option.

The command `ls -al` shows the directory's contents in a list and provides some extra information. The command `ls -la` provides the very same output.

**Question 4.3.** What do you think the output of `ls -l` will be? And what will be that of `ls -a -l`? Run the commands to check your answers. (See the appendix for the answer.)

**Question 4.4.** In the output of `ls -al`, can you see which entries correspond to directories (folders)? (See the appendix for the answer.)

**Question 4.5.** You'll notice the entry's modification date in the listed output (and the creation time if the file was created recently). Right before that, there is a number. For files, can you guess what this corresponds to? (See the appendix for the answer.)

So, the `ls` command shows you the contents of the current directory. But what is this directory?

**Exercise 4.6.** Run the command `pwd` to show the current directory. You'll notice that directories use forward slashes on Linux instead of backslashes in Windows. There are no drives with letters such as C:, D:, etc., on Windows.

If you just logged into a shell, you will notice that the current directory is your home directory, which likely looks like `/home/name`. Here `name` is the username of the current user, typically a word consisting of only lowercase letters.

**Exercise 4.7.** Now go one directory up (to the **parent directory** as it is called) by running `cd ..` and confirm this worked by entering `pwd`. If you're curious about the contents of this directory, you can use `ls` again, with or without arguments.

You saw here two dots `..` being used. They are a shortcut for the parent directory, regardless of the current directory. Similarly, a single dot `.` is a shortcut for the current directory and a tilde `~` is a shortcut for the current user's home directory. Both are surprisingly often helpful.

Now, go one directory up again and repeat that a few times. You can confirm with `pwd` that you cannot go further than a directory `/`.

The directory `/` is the **root directory**. You can see the files and directories in Linux as a tree, with this being a main branch. All files and directories — even those on external devices, such as a USB drive plugged into the computer — are in a subdirectory of a subdirectory, etc. of the root directory.

---

<sup>3</sup> The difference between **options** and **arguments** may be a bit confusing at first. An option starts with one or more hyphens and an argument does not. As a general rule, options control the behavior of a command while arguments control its output.



You can go back to your home directory by running `cd home` and then `cd name` with `name` being the current user or by running `cd home/name` at once. You can also run `cd` without **arguments** as a quicker way to the home directory.

**Exercise 4.8.** Run the command `du` and then `du /` and look at the output. As you may be able to guess from the output the `du` command shows you the total size in bytes of its contents (including those of subdirectories) for each subdirectory.

This command can be helpful if you're dealing with a full drive and want to look where the biggest files (or maybe many smaller files) are located. Several things aren't ideal, though.

First, the output isn't sorted, so you'll have to scroll a lot and manually look for the biggest numbers. It would be a lot easier if you could sort the data. Luckily, you can!

**Exercise 4.9.** Run the command `du / | sort` and look at the output. A vertical bar in Linux is called a **pipe** and is used to 'pipe' the output of one command into another.

This is very helpful, except the output doesn't look right. Fix that by running `du / | sort -n` and confirm this gives the output you need.

You can pipe multiple commands after each other!

**Exercise 4.10.** Run the command `du / | sort -n | tail` and notice you'll see the last ten lines corresponding to the ten 'biggest' directories. If you want to see a different number of lines, for example, twenty, you run `tail` with the argument `-n 20`.

**Question 4.11.** Why is the root directory `/` always the 'biggest'? (See the appendix for the answer.)

**Question 4.12.** Can you guess the command to see the first lines of an output? Check if you have guessed correctly. (See the appendix for the answer.)

In the previous exercises, you probably noticed a lot of errors on the screen saying 'du: cannot read directory.' That is because some directories require root permissions to access, and a regular user doesn't have this permission. Later in this chapter, we'll explain how to view those directories, but there is also a way to suppress these errors — or errors in general.

**Exercise 4.13.** Run the command `du / 2> /dev/null | sort -n | tail` and confirm you'll get the same output as before but without the errors.

In Linux, `2>` directs the output of the errors to a file rather than showing them on the screen, and `/dev/null` is like a black hole: any data written to it is immediately discarded.

**Question 4.14.** Why did you have to run the command like that and not like `du / | sort -n | tail 2> /dev/null`? (See the appendix for the answer.)

Sometimes, you want to inspect the data more carefully. One option is to write the data to a file.





**Exercise 4.15.** Run the command `du / 2> /dev/null | sort -n > bigdata.txt` and confirm (do you remember how?) that a file `bigdata.txt` now exists in the current directory.

Now that the data exists in a file, you can search through it. There's a useful command for that, but what was it called again? I can only remember that it started with `gr-`.

**Exercise 4.16.** Type the letters `gr` (don't press Enter) and then the Tab key twice. You'll see a list of all the commands starting with `gr`.

The one we're looking for is `grep`. This allows you to search a text file for files matching a specific condition. Let's look for any line that contains the string<sup>4</sup> 'html.'

**Exercise 4.17.** Type `grep html big` and, without pressing Enter, press Tab. You'll notice how probably<sup>5</sup> the command completes itself to `grep html bigdata.txt` and press enter and run that command. Confirm the output is as expected.

You have just incidentally learned a very useful command line trick called **tab completion**. This is less about commands you may have forgotten and more about making it easier to type sometimes long commands and filenames. Using the command line regularly will make you very grateful for tab completion.

But the main goal was to search within the file; you've just learned how to do that. That is, you've learned the very basics. You will find that `grep` is a very powerful command with many possible options.

**Question 4.18.** Let's say you want to look for all directories containing the string '.html,' with a preceding dot<sup>6</sup>. Run the command as before and notice that the output isn't what you would have expected. Do a quick Internet search on how to get the desired output. Do you understand the reason? (See the appendix for the answer.)

Linux commands not working exactly as you expect is very common for beginners and advanced users alike. A search engine is the most important tool in such cases.

**Question 4.19.** Can you think of how to get the same output without using a file but with piping instead? (See the appendix for the answer.)

But what if we want to look inside the file itself? That's when the `less` command is useful.

**Exercise 4.20.** Run `less bigdata.txt` to look inside the file. You can 'walk' through the file using the arrow keys, Page Up, Page Down, Home and End. You can also search using the forward slash `/`. Make yourself comfortable with the environment for a bit, then exit by pressing 'q.'

<sup>4</sup> A 'string' is any piece of text: a word, a URL, some gibberish etc.

<sup>5</sup> The one exception is where there is more than one file in the directory whose name starts with `big`. In that case, you'll have to press Tab twice to see all the files.

<sup>6</sup> This is a pretty hypothetical example. The file only contains directories, not individual files.



If you guessed it is possible to ‘pipe’ the output of another command to less, then you are correct! Don’t hesitate to try this!

Unsurprisingly, very common procedures such as moving, copying and removing files are also possible.

**Exercise 4.21.** Run `mv bigdata.txt BigData.txt`, then `cp BitData.txt BigData2.txt` and finally `rm BigData2.txt`. At each step, try to predict what the command will do and confirm you were right by using `ls`.

You are probably asked to confirm removing the file in the last command. That’s a good thing, as it gives you an extra chance to prevent accidentally removing a file. Likewise, `mv` and `cp` will request confirmation before overwriting an existing file.

Natively, the commands don’t ask for confirmation, but on *most* Linux systems, they are ‘aliased’ so that the option `-i` is invoked automatically. For example, `rm` is an alias for `rm -i`. You can confirm this by running `alias rm`.

The use of *most* in the previous paragraph is important. Those aliases aren’t set on some Linux systems, and files are removed or overwritten without confirmation. Be mindful of that, especially if you are using an environment you’re not familiar with!

**Exercise 4.22.** Run `mv BigData.txt /tmp` to move the file to the `/tmp` directory.

Here, you see how to move (or copy) files to a specific directory. You also learned about the **temporary directory**. It is a directory that is present in all Linux systems and is regularly emptied. It is ideal for temporarily storing files you don’t need to keep long-term.

For example, when we initially created the file `bigdata.txt`, it would have been better to store it in `/tmp` to avoid it being left behind in the home directory. Thankfully, the previous two exercises ensure nothing is left that won’t automatically be flushed.

There is one more important Linux command worth mentioning, which is `find`.<sup>7</sup> This command lets you find files and directories with specific conditions.

**Exercise 4.23.** Run the command `find /` and look at the output. This shows you all the files and directories on the current system (and remember that this includes external devices connected — ‘mounted’ in Linux speak — to it).

The `find` command is helpful during investigations on Linux systems, for example, web servers, where you often want to look for recently changed files.

**Exercise 4.24.** Run the command `find / -mtime -30 -type f -size +1G 2> /dev/null` to show all files (not directories) on the system that were last changed less than 30 days ago and that are at least 1 GB in size.

<sup>7</sup> Those familiar with the Windows command prompt may know a command with that name. Confusingly, that is roughly the equivalent of Linux’s `grep`.



Those reading carefully will have noticed something strange: with `ls`, we had an option `-al` which was short for `-a -l`, but here the option `-mtime` isn't short for `-m -t -i -m -e`, nor is the option preceded by a double hyphen. Unfortunately, some commands are an exception to the rule you learned previously.

**Exercise 4.25.** Run the command `touch test`, then confirm (using `ls`) that a file `test` was created and remove this file (using `rm`). The `touch` command is occasionally helpful (including later in this guide) to ensure a file is present on the system: if it exists, nothing happens; if it doesn't, it will be created as an empty file.

## Getting help on Linux commands

The most effective way to learn about Linux commands, especially if you have a specific use case in mind, is to search the Internet. However, there are two 'native' options too.

The first is to run the command with the `--help` option (for some commands, `-h` works too). This often shows you all the possible ways the command can be invoked, though the amount of detail varies greatly. As the output may be long, you can pipe it into `less`.

The second possibility is to consult the internal man page for the command by running `man command`. Those 'man pages' can be very long and tedious but can also be quite helpful, especially if you are looking for the specifics of a certain option.

**Exercise 4.26.** Use the `man` function to change the `find` command in Exercise 4.24 to show all files of any size whose name ends in `.html` that have been created in the past week

## Who's the boss? Sudo

In some of the previous exercises, you saw some errors because you didn't have permission to access certain files and directories. Linux has a complex system of users and permissions, at the top of which there is the `root` user, who can do almost everything.

If you don't have permission to do something, you can log in as `root` and proceed<sup>8</sup>, but that's rarely, if ever, a good idea: there's just too much that can go wrong. The way to run commands with root privileges is to use the special command `sudo`.

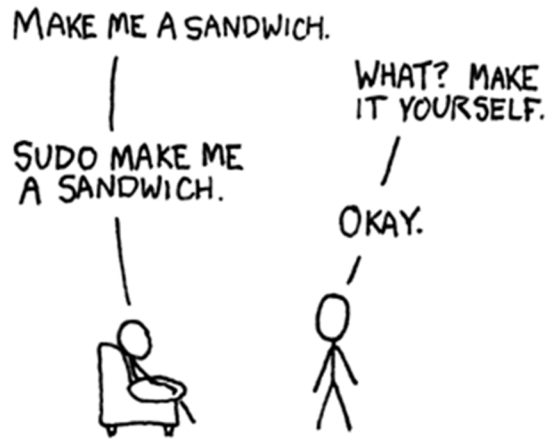
You just add `sudo` before your command (for example, `sudo du /`), and you are prompted for your password (not that of the `root` user!). The command is run as if your user had `root` access.

However, you may receive an error saying you're "not in the sudoers file." Only specific users can use `sudo`, which is especially important on a system with many different users. If you run into this issue, speak to the administrator of the machine you're on, or if you are that yourself, do an Internet search on how to add a user to the sudoers file on your specific system. You don't want to make a mistake in this area, and the advice may vary depending on your particular Linux system.

---

<sup>8</sup> In case you're curious: `su` is the command to do so. It's best avoided, but if you want to login as another non-`root` user, `su name` is the way to go.





Source: <https://xkcd.com/149/>

**Exercise 4.27.** Run the command `du / > /dev/null`. Here the output is sent to `/dev/null` and you only see the errors. Then run the command again with `sudo` to confirm it works. As mentioned above you may be told you aren't in the sudoers file but for example on REMnux it should work out of the box.

## Editing files

Linux relies heavily on text files. In Windows, you'd use *Notepad* to open and edit them; on a Linux system with a graphical user interface, there are many similar editors. However, sometimes all you have is the command line, and there isn't the possibility of editors with fancy menus you can click on with the mouse.

There are, however, plenty of options for editors that work within the command line. As a beginner, `nano` is probably the way to go. You edit a file by typing `nano` followed by the filename. You can start typing text right away, and at the bottom of the screen, you will find some helpful commands, such as `Ctrl-O` for save and `Ctrl-X` for exit.

```
GNU nano 3.2          nanotest.txt          Modified
Hello!
This is what the nano editor looks like.
Do you notice the commands at the bottom of the screen? Here ^ means Ctrl, while M- means Alt.
So you can save the file using Ctrl-O and undo the last thing you typed with Alt-U.

^G Get Help      ^O Write Out    ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos    M-U Undo      M-A Mark Text
^X Exit          ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line  M-E Redo      M-G Copy Text
```



**Exercise 4.28.** Use `nano` to create a file `/tmp/nanotest.txt` with some text. Save it and exit `nano`.

For an occasional Linux user, `nano` will suffice. If you find yourself regularly using text editors on Linux, you may want to familiarize yourself with `vim` or<sup>9</sup> `emacs`, two more advanced editors that come with a somewhat steep learning curve.

---

<sup>9</sup> The use of 'or' is deliberate here: there is a big almost religious debate among Linux users about which one is the best of the two.



## Chapter 5: Malware

This chapter introduces you to the concept of **malware**, which is crucial to understand in incident response. However, it is equally important to understand not all incidents involve malware. For example, online harassment often doesn't, nor does phishing for credentials.

Malware is short for **malicious software**. It is common to think of malware as a program that does something bad — for example, copy sensitive files to a remote server or give someone remote access to the device — but not all malware manifests as a program. Some malware is just a process that is running in the background. Malware that doesn't manifest as a program on disk is referred to as fileless. Keep in mind that looking for malware isn't just about looking for malicious files.

Some malware only operates within the context of another program. An example would be malicious browser extensions, which operate within the context of a browser and can, in the worst of cases, access everything a browser has access to, which could include emails, social media accounts and collaboration software.

Sometimes, you may see references to **viruses**, **trojans** and **bots**. They all have specific meanings — a virus 'infects' an existing program by making a small modification to it, a trojan is installed in the mistaken belief it is a legitimate program, and a bot is, together with many other similar kinds of malware, part of a botnet — but they are increasingly used interchangeably. It's not worth being overly concerned with the difference.

The same is true for malware considered a **worm**, which would be any kind of malware that automatically spreads itself. In any case, worms are a lot less common these days than they were in the past.

### Computer programs, malware and anti-analysis

Malware is a kind of computer code that is usually but not always a program. You don't need to be able to program for this guide, but a very general understanding of how a program works is probably helpful. If you've never seen a computer program before, don't worry! Think of it as a series of instructions, like the ones you enter on the command line, with some special 'things' that help with the flow.

As an example, here is a very short Python program that uses these three things:

```
people = [ 'Ali' , 'Brittney' , 'Carolina' ]

def hello(person):
    print(f'Hello, {person}!')

for p in people:
    hello(p)
```

First, the name ('variable' in programming speak) `people` is given to a list of three human names, then a function `hello` is defined (hence the `def` keyword) that takes an input and prints a 'Hello' message that includes that input (the `print` function is built into Python) and finally a loop (a for loop) runs through all the `people` and runs the `hello` function on each of them.



Can you guess the output of the program?<sup>10</sup>

That was a ‘good’ program. Not a particularly useful one, but certainly not a bad one. If you want to understand what malware looks like, imagine the program doesn’t run through a list of people but through a list of files stored in a particular directory. Instead of printing, it uploads the files to a remote server.

It goes beyond the scope of this chapter to explain how all this is done in Python, but it suffices to say it’s not particularly complicated. It also wouldn’t be difficult for a security program to detect this behavior, which is why a lot of malware uses code obfuscation and anti-analysis techniques.

To understand what this may look like, let’s pretend for a moment that the original Python program from a few paragraphs ago was actually something harmful and that the author wanted to hide this functionality.

Let’s look at another Python program, which uses obfuscation:

```
import base64

yubivetyenv = [ 'QWxp' , 'QnJpdHRuZXk=' , 'Q2Fyb2xpbmE=' ]

def hsdgljknjmkklkj (ngslbhglhml) :
    print (f' {base64.b64decode ("SGVsbG8=") .decode () } ,
{base64.b64decode (ngslbhglhml) .decode () } ! ' )

for ygtnwgnlnn in yubivetyenv :
    hsdgljknjmkklkj (ygtnwgnlnn)
```

Unlike the previous program, you won’t be able to see what it does, just from looking at the code. Even an experienced programmer can’t predict its output without spending some time analyzing it. As it happens, the output is exactly the same as the previous program. All that has changed is that a library of functions related to base64 encoding and decoding<sup>11</sup> was imported, then all the strings (the people’s names and the word ‘Hello’) are base64 encoded in the program and decoded in the function, and the function and variable names have been given nonsensical names. This is actually a pretty simple kind of obfuscation.

More than confusing the analyst, obfuscation is used to stop anti-malware programs or threat analysts from detecting the malware. For example, the code might refer to a URL or domain from which a next-stage payload is downloaded. Security products will scan the program for blocklisted URLs and domains and will block any code that tries to connect to these domains. So, URLs and domains are almost always heavily obfuscated in malicious code.

**Code obfuscation**, which you have just seen, is an example of an **anti-analysis** technique, a general term for techniques that make it harder to manually and automatically analyze and detect malware. Because of code obfuscation in particular and anti-analysis in general, malware analysis is a lengthy process. In almost all cases, you can find out enough about the malware by using sandboxes (more on which in Chapter 10), malware repositories (like VirusTotal or Malware Bazaar; more on the former in Chapter 7) and other people’s public analysis (which you read about in Chapter 3).

---

<sup>10</sup> If you want to run this program, save it in a file `test.py` on a Linux terminal — again, it’s you may have to wait until in the next chapter and you have set up REMnux — and then run it by invoking `python3 test.py`

<sup>11</sup> Base64 is a very basic kind of encoding. Certainly not encryption, but it makes normal words look like gibberish. You’ll learn more about base64 in Chapter 9.



## Different kinds of Malware

There are various kinds of malware with different functionality, some of which are more harmful than others.

For most users, especially at-risk users, the worst kind of malware gives an adversary full or partial access to an infected device. This kind of malware is referred to as **RAT** (for Remote Access Trojan<sup>12</sup>), **backdoor** or **spyware**. The difference between these three isn't always very clear, and different threat analysts may use different descriptions for the same malware. RAT is a term more commonly used for malware where the adversary can control the operating system as if they were sitting in front of it, like using TeamViewer, while spyware typically gathers information as and when requested by the adversary.

A particular kind of spyware that is installed manually on an unlocked device, in particular a phone, is referred to as **stalkerware**. Stalkerware is typically used in abusive relationships to spy on (ex-)partners but is also sometimes used to target civil society and may be installed following the seizure of a device.

Related to this category is information stealing malware, **infostealer** for short, which is malware that steals specific information, such as passwords and browser cookies. This kind of malware often serves its goal by running only once, so there is less need for persistence.

A **keylogger** is malware that captures keys entered on the keyboard, often with contextual information, and sends these to a server controlled by the adversary. It is commonly used to steal passwords.

**Ransomware** is malware that encrypts files on a single device or, more typically, a network of devices and demands a ransom to decrypt them. Ransomware has become a serious problem for organizations around the world in the past decade, but it is less an issue for civil society organizations, partly because their low funds make them uninteresting targets for financially motivated actors and because ransomware actors usually target networks such as Active Directory<sup>13</sup>, which civil society organizations less commonly use.

However, civil society organizations may still be affected by ransomware if a provider they use is hit by ransomware. In that case, an extra concern is that ransomware actors often threaten to publish data from the affected organization if they don't pay the ransom.

A **wiper** is malware that wipes the content of a directory or drive on one or multiple computers. Wipers aren't very common but are sometimes used for political purposes as a digital equivalent of arson. The Stuxnet malware, which in 2010 was found targeting a nuclear enrichment facility in Iran in what is believed to be a US operation, is a famous example of a wiper, as are various wipers deployed by Russian actors in Ukraine.<sup>14</sup>

**Adware** is software that displays an unreasonable number of ads. It falls within a grey area between malware and legitimate software but often decreases the security of a device and leads to the device 'feeling' less clean.

---

<sup>12</sup> Or Remote Access Tool, referring to the fact that some RATs, such as the widely used Remcos RAT, are sold legitimately to offer remote support and then repurposed for malicious purposes.

<sup>13</sup> Active Directory is a widely used system by Microsoft to handle user authentication, file sharing and other activities on a network. If you're mostly dealing with small groups and individual users, there's a good chance you won't come across it in your work. However, if you find yourself regularly reading malware analyses, you may want to become familiar with its basics — as often, Wikipedia is your friend here

<sup>14</sup> For those interested in Stuxnet, Kim Zetter's 2014 book *Countdown to Zero Day* is a recommended read. For those interested in wipers and other malware deployed in Ukraine, Andy Greenberg's 2019 book *Sandworm* is worth reading.





Finally, there are various kinds of malware that merely use the infected device and its resources, for example, to send spam, mine for cryptocurrencies, click on ads in the background or use the machine as a **proxy**<sup>15</sup>. While these don't typically access sensitive data and may be considered less dangerous, they still provide access to an infected device that could eventually be used for something more nefarious. Moreover, they, together with adware, show that something happened on the device or network that leads to an infection, and this something would ideally be fixed.

## Persistence

A lot of malware intends to hide on the system and keep running even after the device is restarted, something that is referred to as **persistence**. You will have noticed — or maybe you have explicitly set — some programs always opening when you start up a laptop. That is an example of persistence, albeit a legitimate one. Not all malware needs to achieve persistence (it was previously mentioned an infostealer typically doesn't), but for any kind of backdoor, the actor operating typically wants to maintain long-term access, and persistence is thus necessary.

Later, we will look into how malware maintains persistence on various operating systems. From an analyst's point of view, an advantage of persistence is that it results in some artifact being present on the device, which you can search for.

Very occasionally, especially for advanced and highly targeted malware, its authors aim for the opposite of persistence, where the malware, after completing its task, attempts to remove all of its traces from the device.

## Infection chains

Sometimes, malware is downloaded directly into the device. An example of this is malware masquerading as a legitimate program that tricks a user into downloading it onto their device. However, it is increasingly common, especially for Windows, for malware to take multiple phases to make it onto the device.

For example, an email may contain a link to a Word document hosted online. When opened on a Windows computer, the user is tricked into enabling macros, which run a PowerShell script on the machine. (You don't need to understand what these mean at this point; you'll learn more about them in Chapter 10.) This PowerShell script downloads a non-persistent piece of malware, which the actors behind the campaign use to install the final payload: the 'thing' that does the real job.

From a threat actor's point of view, there are two advantages to this approach.

The first is that it is easier to evade detection this way. The downloading and installing of a program is an ideal opportunity for an antivirus program to scan a file for it being malicious, and while this isn't 100% accurate, a lot of malware would be stopped this way or would lead to the user being presented with a warning.

The second reason is that it offers threat actors a great level of flexibility. They may only decide to run the final payload if certain conditions are met; for example, the user is in a specific country, or the environment is recognized as not a research environment. (The running of the final payload being dependent on the location is one anti-analysis technique deployed by malware.) A lot of malware operations, especially those operated by cybercriminals, sell the **'initial access'** provided by an earlier payload to other groups who will

---

<sup>15</sup> In that scenario, someone else's Internet



install the final payload, depending on what is deemed most profitable for the particular device. This is similar to how online advertising slots are sold in real time, depending on the user's previous activities.

It is good to keep this in mind when analyzing malware, in particular on Windows. Just because your analysis, for example, using a Sandbox, shows a cryptocurrency miner was installed doesn't mean the same would have happened to the real target if they had run the malware. It may even be a decoy to confuse analysts.

## Decoy

**Decoy** is a technique used to confuse analysts and users alike. It is most commonly used by performing an 'expected' activity during the installation of the malware. For example, a malicious Word document would also contain real content, perhaps even the kind of content that would make a technically less experienced user believe the enabling of macros is justified.

However, as in the previous section, a decoy process is sometimes used as an anti-analysis technique where less harmful malware than the intended one is installed if an analyst's environment is recognized. An experienced analyst will become suspicious if the analysis is interrupted fully but may be less likely to be suspicious if a real (albeit less serious) malware is executed.

## Command and Control

Most malware needs a way to connect to its actors, either to send data from an infected device or to receive commands to execute certain tasks — and often both. Typically, this is done by making a connection to a **command and control** server, often shortened to C&C or C2.

A lot of malware contains a hardcoded list of IP addresses and/or domain names (such as 1.2.3.4 or example.com; more on these in chapter 7) of command and control servers, often with an ability to dynamically update the list for extra resilience. Sometimes, the list of C&C servers is obtained from a public site; examples that have been used in the past are the comments to a particular YouTube video or posts from a particular Twitter account.

Some malware includes an algorithm to generate a new daily list of C&C domains. This is called a **domain generating algorithm**, or DGA for short. DGAs allow actors to register the domains when they need them. However, once a DGA is discovered by a security researcher, they too can register a domain to 'sinkhole' it. Sinkholes will be covered in chapter 7.

Occasionally, malware uses Tor onion services for C&C communication, which are a lot harder to take down. If you're not familiar with Tor, now is a good time to read up on it; its [official website](#) does a good job of explaining how Tor works. There is no need to study the protocol in detail, but make sure you're familiar with the concept of onion routing, as well as with onion services and .onion domains.

Different malware uses different protocols to connect to a C&C server. Many use common protocols such as TLS or SMTP, while other malware families use custom protocols.

One particular note should be taken of malware that uses DNS for C&C communication, where tiny bits of information are encoded in the requests and responses on a domain managed by the threat actors (this is a special case of a technique referred to as **DNS tunneling**). Some more advanced malware uses this to communicate from infected devices not directly connected to the Internet that still have access to DNS.

Because most devices don't have ports open to the public internet, a malware actor can't send requests directly to devices under their control. Rather, an infected device will connect to the C&C server very often,

ready for the actor to respond with specific tasks. This makes most malware rather noisy, which can be used by security products to detect it on the network, either because of connections to known C&C servers or because a custom C&C protocol is detected. The Internews team hopes to draft a future chapter that will deal with performing forensics on the network.

In limited cases, malware may use non-network-based methods for C&C exfiltration, for example, by copying data to USB devices on the network and then somehow ensuring the actors eventually read these. Researchers have also shown very fancy techniques of exfiltrating data from air-gapped networks, for example, by using a printer to send light signals to a nearby drone. These methods are rarely deployed against civil society and are not something you need to concern yourself about in your work.

## Antivirus

Almost as soon as the first malware was discovered — in the 1980s in the form of computer viruses — **antivirus** programs were written to block and detect malware as well as to clean up infected devices. Antivirus products continue to exist today, though they are often marketed under names such as anti-malware or endpoint security.

At its core, an antivirus product looks for files to see if they are malicious by checking them against a database of known malware and by performing heuristics on them, which is a fancy way of saying that it checks if a file 'looks like malware.' However, on operating systems where it is able to run with high privileges, it is also able to look for malicious behavior and could thus block file-less malware.

Antivirus isn't perfect, especially against more advanced malware, and especially given that malware regularly attempts to disable an antivirus product or add itself to an internal list of allowed programs. It would be wrong to consider an antivirus product — or a security product in general — a perfect solution against malware, no matter what the vendor's marketing department wants you to believe.

It is also true that operating systems have become a lot more hardened and contain many natural defenses against malware. It isn't as easy to infect a device as it was in the mid-2000s when simply connecting a Windows computer to the Internet would lead to it being infected in about 20 minutes, barely enough time to install updates and turn on antivirus.

At the same time, it would also be wrong to dismiss antivirus as useless. It will detect many of the more mundane threats, which, even in high-risk environments, will be the majority of malware seen. It is particularly good at detecting malware hidden in cracked software or other programs that present themselves as free versions of otherwise paid-for software. You will probably know how common such software is in your part of the world, but it is fair to say they are common even in many high-risk civil society environments.

An antivirus product already present on the device can also help you during investigations, for it will provide logs that you can use during an investigation.

If certain antivirus products are commonly used by the groups you support, it will be helpful for you to become familiar with how they work and how you can check their logs for potentially malicious activity on the device. If you can, it could help to have a contact within the company — look for a researcher, not a salesperson! — who can help you during investigations.



## Vulnerabilities and exploits

The term **vulnerability** is closely linked to malware. It refers to a weakness in a computer system that allows someone or something to perform an unauthorized action. Vulnerabilities are usually caused by human mistakes and are surprisingly hard to avoid.

For example, imagine a system that lets you log in with a username and password, but a mistake allows a user to enter an empty password and still have access to the system. While such basic vulnerabilities exist, most vulnerabilities are more complex. For example, entering a very specific long string in an input field causes certain commands to be executed on the system.

There are many different kinds of vulnerabilities. For example, **remote code execution** allows an adversary to execute remote code on a system, such as a web server or a mail server. This gives an adversary the same access as someone directly connected to the server, and they could use this access to read or modify things or to connect to other systems on the same network.

A **privilege escalation** vulnerability gives someone more privileges than they should have. For example, if there is a way to grant an ordinary user on a Linux system to run commands as the root user without having special privileges.

You can probably guess what an **arbitrary file upload** is from its name. If you can do this on a web server that runs PHP (more on which in Chapter 11), you can probably take over the web server.

A method to make practical use of a vulnerability is called an **exploit**. Sometimes, an exploit naturally follows from a vulnerability — for example, in the empty password example above — but sometimes, it's very complicated, and there are vulnerabilities for which no exploit has been found.

Not all vulnerabilities are as serious as others. Assessing the seriousness of vulnerabilities is complicated, but if you want to have a go, try to understand a) what conditions need to be met for the vulnerability to be exploited and b) what extra things does exploiting a vulnerability allow one to do.

For example, a remote code execution vulnerability in a web server that only requires one to be able to connect to that web server is really serious. However, a vulnerability in a protocol that means one has to tap network traffic for at least a week to be able to steal a password isn't particularly serious in almost all cases: few adversaries have the power to tap traffic for a week, and for those that do, there are almost always easier ways to steal passwords.

Most vulnerabilities are given a number in the form *CVE-year-number*, where *year* is the four-digit year in the Christian calendar, and *number* is a number of four or more digits. For example, CVE-2017-0199 was a vulnerability found in Microsoft Office in 2017.

A fix for a vulnerability is called a **patch**. Security updates for software typically include patches for vulnerabilities. Patching a vulnerability isn't always easy and, in rare cases, not possible at all.

A vulnerability and an exploit discovered before the entity responsible for the software or hardware is aware of it is called a **zero-day vulnerability** or **zero-day exploit**. For users of the software or hardware, this means the vulnerability can be used against them even if they have fully patched.

There is some debate as to whether the time between awareness and the release of a patch still counts as zero-day: the exploit can still be used against a fully patched system, but the awareness means that it is, at least in theory, possible to apply some **mitigation** techniques.

Sometimes, mitigation techniques exist that could prevent future vulnerabilities. One example very relevant to civil society is Apple's Lockdown Mode on iOS, which hardens the device by removing some features that



have been exploited in the past by iOS malware, such as Pegasus. In this case, mitigation comes with a usability cost.

Some malware exploits vulnerabilities. For example, the aforementioned CVE-2017-0199 is used by malicious Office documents to execute code on the system if the Word version hasn't been patched. Note that the patch for this vulnerability was issued in 2017, but more than six years later, many users still run unpatched versions of Word.

It is also important to note that a lot of malware doesn't exploit vulnerabilities at all other than the human nature to click through security warnings.



## Chapter 6: Virtual Machines and REMnux

In this chapter, you will install **REMnux**, a Linux-based toolkit for malware analysis. REMnux comes with many tools pre-installed, making it easier to complete some of the tasks we will learn to do. Because REMnux is based on Ubuntu, a common Linux distribution, there are other tools that are not part of REMnux that we can easily install.

We will install REMnux as a **virtual machine** (often shortened to VM). You can think of a virtual machine as a ‘computer inside a computer.’

### Virtual Machines and snapshots

Within your physical computer (called the ‘**host**’), a program called a ‘**hypervisor**’ can start one or more virtual machines (referred to as ‘**guests**’). Each of these will allow you to do pretty much anything you can do if you had installed them as a separate machine, giving you a number of machines to work with without having to have a physical computer for each of them. For example, you can set up a Windows host machine and run Linux in a VM, giving you almost all the benefits of running both operating systems but on a single computer. You can also set up a Linux host machine and run Windows inside a VM!

There are many kinds of hypervisors. There are several options to run virtual machines on your laptop or desktop computer, such as VMware, Hyper-V for Windows<sup>16</sup>, Parallels for macOS and **VirtualBox**. We will be using VirtualBox, which is free to install and runs on many operating systems.

One of the most useful aspects of virtual machines is the ability to take ‘**snapshots**.’ A snapshot stores the current state of the operating system of the VM, including what is actively running when the machine is on. You can then revert back to a previous snapshot if something went wrong. It is good practice to take a snapshot before making any big changes to allow you to roll back if needed.

A snapshot is different from a backup. A backup stores all data, but only that. A snapshot stores all the information of the operating system, including what programs are running and what windows are open<sup>17</sup>. However, it does so in an efficient way and only stores whatever is needed to revert to that state. You don’t have to understand the details of this latter comment; just keep in mind that snapshots are often a lot smaller than backups.

Still, snapshots can take up a lot of space on your computer. If you regularly take snapshots, make sure you delete snapshots you don’t need anymore.

Another big advantage of virtual machines, especially when using snapshots, is that you can recover when something goes wrong. For example, if you want to make some configuration changes to a virtual machine that you worry may mess up things, you can take a snapshot first. If the configuration changes do indeed mess up things, you can simply revert to the snapshot!

---

<sup>16</sup> For technical reasons, Hyper-V is less suitable for the purpose of analyzing malware.

<sup>17</sup> That is, if you took a snapshot of a running machine. You can also take a snapshot of a machine that has been turned off.



This makes virtual machines ideal for dealing with files that may be malware. You can store them in a virtual machine and run various tools to analyze them. You could even, in theory, run them and just revert to a previously taken snapshot afterward. You will not be doing that in this chapter, and in a later chapter when you learn about sandboxes, you will discover why it's actually quite complicated to do so.

Analyzing files without running them is called **static analysis** (as opposed to **dynamic analysis**, where you run the files). Static analysis means you don't have to worry about isolating the virtual machine from your host and from the Internet.

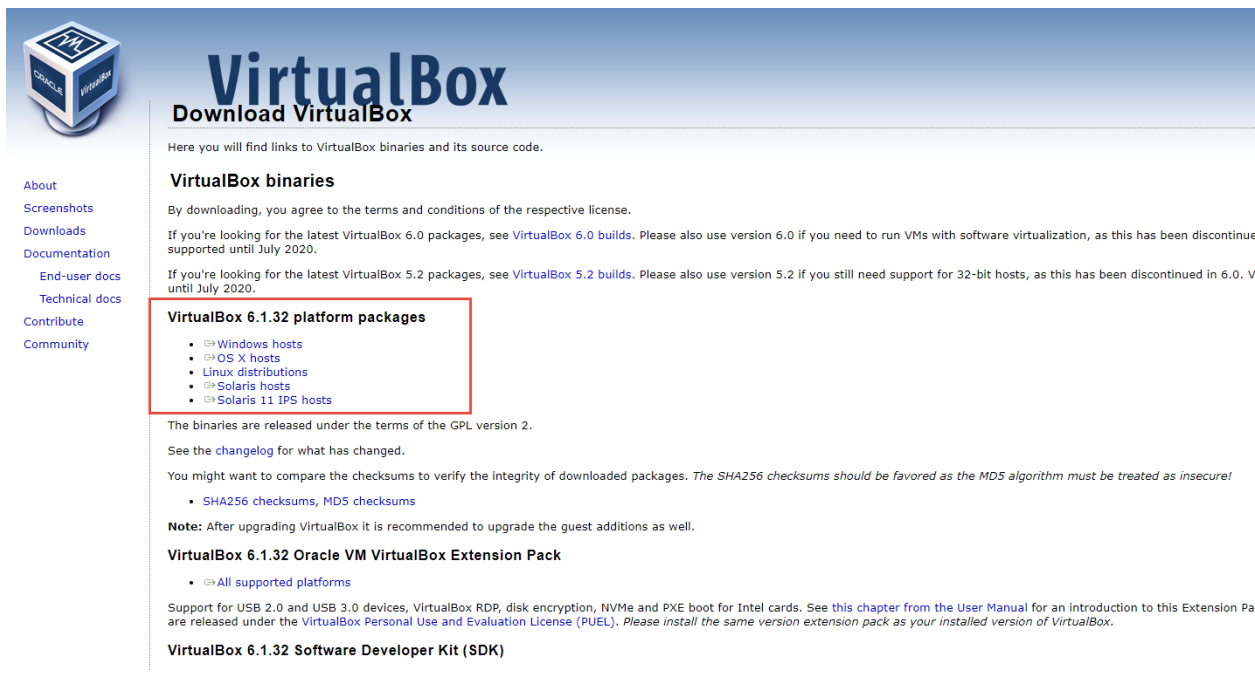
## REMnux

Most malware is written for operating systems other than Linux, so if you perform your analysis on Linux, you can't cause harm by accidentally running it. Linux also doesn't run an antivirus by default — such as Microsoft Defender on Windows — so you don't have to worry about this flagging any of your files. Finally, a lot of malware analysis tools are written for Linux.

REMnux is just a version of Linux (a 'Linux distribution') that has a lot of these tools pre-installed, making it ideal for analysis. While installing programs on Linux is pretty straightforward in most cases, you will occasionally run into dependency issues, where your program requires another program to be installed or a specific setting to be changed. By using REMnux, you won't have to worry about installing the individual tools and at the same time, you will have a fully functioning Linux machine.

If you are using a macOS computer with a M1 or M2 chip, you can skip to the section at the end of this chapter. For technical reasons, you cannot install REMnux on such a processor, not even in a virtual machine and not even in a special hypervisor like Parallels. The rest of this section assumes you are running X86 architecture. If you are running Windows or Linux, you almost certainly are.

To install REMnux, we first download VirtualBox from its [official website](#). Alternatively, if you are using Linux as a host, you may be able to install it directly through your distribution's package manager, which is easier and more secure, if only because you're less likely to make mistakes.



**VirtualBox**  
Download VirtualBox

Here you will find links to VirtualBox binaries and its source code.

**VirtualBox binaries**

By downloading, you agree to the terms and conditions of the respective license.

If you're looking for the latest VirtualBox 6.0 packages, see [VirtualBox 6.0 builds](#). Please also use version 6.0 if you need to run VMs with software virtualization, as this has been discontinued until July 2020.

If you're looking for the latest VirtualBox 5.2 packages, see [VirtualBox 5.2 builds](#). Please also use version 5.2 if you still need support for 32-bit hosts, as this has been discontinued in 6.0. V until July 2020.

**VirtualBox 6.1.32 platform packages**

- [Windows hosts](#)
- [OS X hosts](#)
- [Linux distributions](#)
- [Solaris hosts](#)
- [Solaris 11 IPS hosts](#)

The binaries are released under the terms of the GPL version 2.

See the [changelog](#) for what has changed.

You might want to compare the checksums to verify the integrity of downloaded packages. *The SHA256 checksums should be favored as the MD5 algorithm must be treated as insecure!*

- [SHA256 checksums, MD5 checksums](#)

**Note:** After upgrading VirtualBox it is recommended to upgrade the guest additions as well.

**VirtualBox 6.1.32 Oracle VM VirtualBox Extension Pack**

- [All supported platforms](#)

Support for USB 2.0 and USB 3.0 devices, VirtualBox RDP, disk encryption, NVMe and PXE boot for Intel cards. See [this chapter from the User Manual](#) for an introduction to this Extension Pack are released under the [VirtualBox Personal Use and Evaluation License \(PUEL\)](#). Please install the same version extension pack as your installed version of VirtualBox.

**VirtualBox 6.1.32 Software Developer Kit (SDK)**



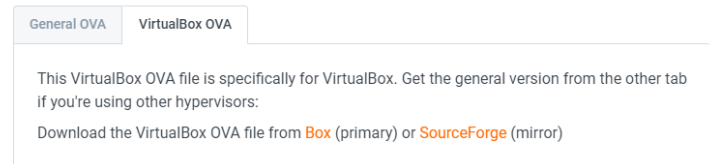
During installation, you can just follow the instructions. It is important to check if you have enough free disk space: ideally, you need at least 30GB to install and use REMnux, though you can probably get away with a bit less.

After you have installed VirtualBox, you can download REMnux on your host computer from [its official site](#). You should download it as the .ova file designed for VirtualBox.

## Step 1: Download the Virtual Appliance File

The REMnux virtual appliance approximately 5 GB. It comes as an industry-standard OVA file, which you can import into your virtualization software. It's based on Ununtu 20.04 (Focal).

Decide which OVA file to download. Unless you're using Oracle VM VirtualBox, get the general OVA file. If you're using VirtualBox, get the VirtualBox version. Download your preferred OVA file:



General OVA    VirtualBox OVA

This VirtualBox OVA file is specifically for VirtualBox. Get the general version from the other tab if you're using other hypervisors:

Download the VirtualBox OVA file from [Box](#) (primary) or [SourceForge](#) (mirror)

Some browsers (e.g., Brave) change the extension of the OVA file after downloading it, possibly giving it the incorrect .ovf extension. If that happens, rename the file so it has the .ova extension before proceeding.

Once you download it, double-clicking the downloaded .ova file should install it, but you can also follow the instructions provided by REMnux on its website if that does not work. You can leave the settings as suggested during the installation process, but make sure you choose (at least) two CPUs.

Note that REMnux suggests you confirm the hash value. If you're not comfortable doing this, don't worry; you can skip this. It is just a simple way to confirm you downloaded the correct file without any issues. It could also potentially detect a rogue version of REMnux, but the likelihood of this existing is pretty minimal, and the protection this offers is far from perfect.

## Running REMnux for the first time

In VirtualBox, start the guest operating system to launch REMnux. You can log in with the username 'remnux' and password 'malware.' Obviously, this is hardly a secure password. That's not a major concern because to get to your REMnux instance, an adversary would need to get access to your host machine, and if they have that, a strong password won't offer any protection. REMnux isn't designed to guard secrets against a compromised host!

Once you have logged in, open a terminal by clicking on 'Activities' and then on the terminal icon. In the terminal, type:

```
remnux upgrade
```

This will install the latest versions of all the tools. You will also want to run:

```
remnux update
```

This will ensure the underlying operating system is up to date.



You should run this regularly to download updates, and it is recommended that you at least run the first command before starting a major investigation.

If you've not used REMnux before, spend some time getting familiar with the REMnux environment. For example, open the Firefox web browser and confirm you can open web pages, as well as the file explorer, to see how you can view files in your home directory.

**Exercise 6.1.** You are welcome to skip this exercise if you've worked with virtual machines before. If not, it is a simple way to familiarize yourself with the concept of snapshots. We'll leave it to you to figure out how each step is performed; just note that a snapshot is something you take in the hypervisor, not from the guest operating system.

1. Take a snapshot of your REMnux instance.
2. Create a test file somewhere (see Exercise 4.25). Make sure you can see the file in the file browser.
3. Revert to the previously taken snapshot. You will probably be asked if you want to save the current state; you can say 'no.'
4. Confirm in the file browser that the file you created isn't there anymore.
5. Delete the snapshot.

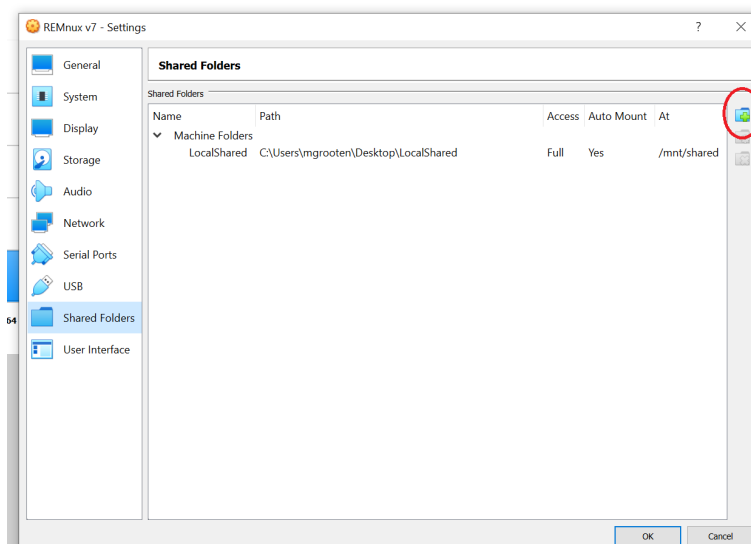
## Handling malicious files

Before analyzing a file on REMnux, you first need to make sure the file is downloaded on or copied to the virtual machine.

One way to do this is to use the Firefox installed on REMnux to log into your email or cloud service where the file is present and download it from there. This should work, but some caution should be applied.

While Ubuntu, on which REMnux is based, is generally a secure operating system, REMnux is designed for analysis, not for handling sensitive content. It is, therefore, a good practice to avoid connecting REMnux directly to your organization's email or cloud accounts. Instead, you could set up a separate Google account for this purpose and then connect to Gmail and/or Google Cloud from REMnux.

An alternative method is to create a shared folder that will be accessible both by the host and by the REMnux guest. You can do this easily in VirtualBox. First, on the host operating system, create a new folder somewhere, for example, on your Desktop, and give it a clear name. Also, make sure the guest operating system (REMnux) is running.



Then, in VirtualBox, open the settings for your REMnux instance and choose 'Shared Folders.' Click on the folder icon with a plus in the top right corner. In the folder path, browse to the folder on your host system you just created. Check 'Auto-mount' and choose a 'mount point' on REMnux. If you're not familiar with file paths in Linux, you can choose `/home/remnux/Shared` to create a folder 'Shared' in your home directory.

Click 'OK' and then 'OK' again. The shared folder on your host now corresponds to the shared folder on REMnux.

**Exercise 6.2.** Follow the procedure above to create a shared folder. In the end, confirm everything worked by copying any file from your host machine to the shared folder there and then check in REMnux if it is there.

It is considered good practice to store files we're analyzing in a password-protected zip file with the password 'infected' (all lowercase) to avoid them being flagged by antivirus software running on the host. This is also how files are shared among researchers.

We'll discuss workflows for handling potentially malicious files in a later chapter. For now, the following workflow is recommended when using the shared folder to analyze files.

1. Make sure the file on the host is in a zip file with the password 'infected.' You will want the file to have been in this format before you even downloaded it on your host machine, especially if it is running Windows or macOS.
2. Move the zip file to the shared folder.
3. Open REMnux and move the file from the shared folder to another folder that isn't shared with the host.
4. Unzip the file and perform the analysis. Read on to learn more about how to complete analysis!

**Question 6.3.** Can you explain the reason for step 3? (See the appendix for the answer.)

## Turning off REMnux

By running a virtual machine, you are essentially running two machines at once, which impacts power usage and, on a laptop, battery life. It is thus good practice to turn off REMnux when you aren't using it.

In VirtualBox, if you close the window of a virtual machine, you are given a choice to save the machine state, send the shutdown signal, or power off the machine. You will want to choose 'save machine state' or send the shutdown signal.

If you save the machine state, you can resume your work more easily. When you turn on the virtual machine again, it returns to the state you left it in, with the same programs running.

By contrast, shutting the VM down is a cleaner way to start a new investigation. The virtual machine is turned off and will need to be rebooted. Do whichever works best for your purposes.

Powering off the virtual machine is the equivalent of pulling the plug, leading to immediate shutdown, which isn't ideal.



**Exercise 6.4.** [This wayback machine](#) is an analysis someone did that uses tools that are available in REMnux. Create a new folder on REMnux, open a terminal and browse the directory. Then execute the commands from the exercise, at least until the third instance of running oledump.py. You do not have to understand what the blog says (this is really not the goal!); just confirm the commands work for you! You shouldn't need to install any new programs for this.

## REMNux on macOS M1 or M2

As mentioned before, REMnux cannot be installed on macOS with an M1 or M2 chip. That's because of a hardware issue: REMnux is made for X86 processes, and these Macs run ARM. Maybe one day, the REMnux team will release a REMnux distribution for ARM, but for now, if you are running such a Mac, REMnux is not a compatible tool.

This is unfortunate, as macOS is otherwise ideal for the kind of threat research this guide explores.

However, there are a number of alternative options. One is to install a Linux-based distribution that uses the ARM64 architecture using a hypervisor (like VirtualBox as used in the walk-through, which is free and open-source, as is UTM, or Parallels, which requires a subscription). Debian and Kali Linux, for example, provide ARM64 installation images. You would then need to install individual tools from REMnux on this virtual machine as and when you need them. This adds the extra complication, though, of needing to install many new programs in your new Linux virtual machine.

A second option is installing the tools on macOS itself. This isn't recommended for beginners and has the added disadvantage of analyzing the malware on your main machine. This is not recommended unless you really know what you are doing.

A third option is to install REMnux in the cloud somewhere. This isn't free, but it will probably cost you less than US\$10 per month and might be a good option for some. It has the added benefit that you can access your REMnux machine from anywhere and that you can share access with others, even those in a different physical location. If you are going for this option, read the REMnux installation instructions carefully, particularly the [section](#) regarding cloud environments.



## Chapter 7: Threat Intelligence and VirusTotal

This chapter focuses on threat intelligence and using VirusTotal.

**Threat intelligence** (or Cyber Threat Intelligence, often shortened to CTI) helps you understand digital attacks and their context, such as who or what is behind them and what links there are between different attacks.

For example, if you are supporting a journalist who received a phishing email, you probably don't just want to understand whether it was indeed a phishing email but also understand what kind of phishing email it was. Was it a run-of-the-mill phishing email sent to thousands or millions of indiscriminate targets? Or was it sent to a specific person and maybe a select few others by an actor who is targeting them? The support you provide may depend on the answers to these questions.

If a threat involves a malicious file or app ('malware'), you may also want to understand roughly what it does. This can help you mitigate the threat in case the malware is run, as well as understand how to defend against it in the future.

Threat intelligence is a huge field, and there is a lot more to learn about it, certainly more than could fit in this guide. If you are interested in diving deeper, threat intelligence analyst Katie Nickels has written a two-part self-study plan ([part 1](#) [wayback machine](#); [part 2](#) [wayback machine](#)).

### VirusTotal

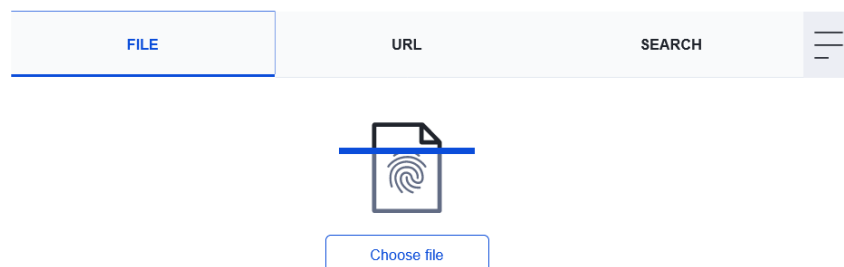
**VirusTotal** is a popular online service that can be used for doing basic and sometimes more advanced threat intelligence. VirusTotal was founded in 2004 in Spain and was acquired by Google in 2012. At the time of writing (November 2022), the front end isn't particularly integrated with other Google services, though. For example, accounts on the site aren't directly linked to a Google account.

VirusTotal's original function was to serve as a malware repository where files could be checked against many antivirus products, and that is still probably its best-known feature. Because people and organizations have uploaded millions of pieces of malware to it over the years, VirusTotal is also likely the largest repository of malware in the world, which makes it a great place to look for links between different kinds of malware.

What makes this even more useful is that VirusTotal doesn't focus only on malware samples but also explores the different elements the samples interact with. VirusTotal also scans and aggregates IP addresses, domain names, and URLs and finds links between them. So you can upload an attachment you find in an email, learn that it is malware, learn that it is linked to a domain that at some time pointed to an IP address that another malware file connected to when run, and then conclude that the original attachment you uploaded is related to this known malware.



Analyse suspicious files, domains, IPs and URLs to detect malware and other breaches, automatically share them with the security community.



Before you open VirusTotal, there are two important things to note. First, *don't just upload* files to VirusTotal without understanding what you are doing. And secondly, making connections like this isn't an exact science, and it's easy to draw incorrect conclusions. For example, IP addresses are frequently re-assigned to new organizations, so what was a malicious IP address last week might be benign this week. Beware of drawing quick conclusions, especially if these conclusions have consequences.

## Important concepts for threat intelligence

This section introduces some important concepts that you'll need to know for the rest of the chapter and future chapters.

This chapter offers a brief overview of these concepts; however, we encourage you to use your favorite search engine to explore each topic in more depth. Search engines will be perhaps one of the most important tools in your threat intelligence work. No matter how advanced your skills are, you will always find the need to look things up, often surprisingly basic things. Being able and feeling comfortable doing so is an important skill for any threat intelligence analyst.

## Indicators of Compromise

The term **Indicator of Compromise** (or IoC for short) refers to any artifact or piece of evidence found while researching a compromised device or network. This usually refers to IP addresses or domains the malware or compromised system connects to or any malicious files the attacker leaves on your system. Sometimes, it also refers to email addresses, Bitcoin addresses, common phrases used by the attacker, or more esoteric artifacts.

If you want to publish about an incident you've handled or a threat you have analyzed, it is good practice to include a separate summary of IoCs you've found. It helps others check their own systems and networks, for example, by checking system or server logs for evidence of a similar attack.

Below is a deeper dive into some IoCs you may encounter.

## IP addresses

**IP addresses** are computer-readable addresses made of numbers (and sometimes letters) for every device connected to the internet. Unlike your street address or phone number, they are often quickly re-assigned by the network so that it can accommodate more devices: for example, your home wifi may reissue the same IP address to your laptop, your phone, or your friend's devices depending on availability. Domain names—the basic building block of the human-readable addresses you may use to visit websites—also “resolve” or are connected to a certain range of IP addresses.

When we refer to IP addresses, we usually mean **IPv4** (IP version 4 addresses). There are also **IPv6** addresses, but in this guide, an IP address is an IPv4 address unless otherwise specified. There are no IP addresses other than versions 4 and 6.

You do not have to understand all the details of how IP addresses work; however, it is important that you can do at least the following:



- Recognize what IPv4 and IPv6 addresses look like.
- Understand what a private IP(v4) address such as 192.168.1[.]2 is<sup>18</sup>
- Understand what Tor is and how Tor hides someone's IP address
- Understand what a VPN is and how it hides someone's IP address

One way that you may find an IP address making a suspicious connection would be finding in the log of your website that IP address requested a URL that appeared to exploit a vulnerability<sup>19</sup>. The IP address you see could be the public IP address of the malicious actor. However, many malicious actors use some kind of proxy, such as Tor or a VPN, to hide their IP address. So, in this case, the IP address may not be relevant for your research. However, "the malicious actor was using Tor to connect" may also be a useful piece of data to keep track of or share.

```
44.203.11.113 - - [15/Nov/2022:10:32:23 +0000] "GET /feed/
87.249.108.114 - - [15/Nov/2022:10:34:44 +0000] "HEAD /feed/
92.247.181.17 - - [15/Nov/2022:10:37:29 +0000] "GET /feed/
54.211.20.179 - - [15/Nov/2022:10:38:05 +0000] "GET /tag/vi
Chrome/80.0.3987.122 Safari/537.36"
3.81.136.228 - - [15/Nov/2022:10:38:56 +0000] "GET /categor
hrome/80.0.3987.122 Safari/537.36"
195.145.170.187 - - [15/Nov/2022:10:39:09 +0000] "GET /cate
85.119.83.156 - - [15/Nov/2022:10:40:04 +0000] "GET /feeds/
```

*IP addresses in a web server log*

If you want to study the IP address, the first question you will need to answer is whether it is:

- fully controlled by the actor running the phishing page (their own site, and not a public hosting site, like for example imgur or YouTube). In this case, you can be reasonably confident that other content hosted there is linked to the same actor.
- shared with other content. Many IP addresses host multiple unrelated services, such as a web server serving multiple websites or a mail server serving multiple domains. These services aren't necessarily related to each other. Also, note that sometimes legitimate content shares the same IP address with multiple different kinds of malicious content. Without any other kind of evidence (for example, phishing pages that look very similar), in this case, you cannot confidently link content hosted on the same IP address to the same actor.
- managed by a legitimate entity but somehow compromised. For example, the malicious actor may have hacked the server. In this case, you need to understand the hack and when it happened before being able to draw any conclusions based on the IP address.

It is often not easy to find out which of these three situations you're in, but a service like VirusTotal can help you with that.

## Domain names, hostnames and DNS

A **domain name** is the part of a human-readable web address that is purchased from a domain registry—mostly the top-level domain (.com, .org, .co.uk, .in, .br, etc.) and whatever immediately precedes that dot (google, amazon, internews, etc). A **hostname** refers to that address as well as any subdomains the

<sup>18</sup> Ignore the square brackets for now; they are explained in the section 'defanging' later in this chapter

<sup>19</sup> This is extremely common and most such requests are done fully automatically and aren't targeted. It doesn't mean the server has the particular vulnerability or even that it runs that particular software!



domain owner may set up within their domain. So `internews[.]org` or `google.co[.]uk` are domain names as well as hostnames, but `www.internews[.]org` and `mail.google.co[.]uk` are only hostnames. The terms domain name and hostname are often used interchangeably. You do not have to understand all the details of how domain names, hostnames and **DNS** work; however, it is important that you can do at least the following:

- distinguish between country-code top-level domains and other top-level domains
- know what a subdomain is
- know how to use the `dig` command on Linux (for example your REMnux instance!) to perform DNS lookups ([this](#) is a good introduction)
- know what A, AAAA and MX records are
- know how to find a domain's registration date using the `'whois'` command
- understand what a domain name registrar is

When you encounter a hostname while analyzing a threat, it is usually because some content is hosted there, or because a connection is made to the hostname. If you want to study the domain name or hostname further, determine whether you are in one of the following situations:

- The domain is legitimate and not used for any malicious purposes. This is, for example, the case for `internews[.]org`. Malware may still connect to legitimate domains: sometimes for an explicit reason (it might, for example, use a site like `whatismyipaddress[.]com` to determine the device's public IP address), sometimes as a decoy.
- The domain is legitimate but also used for malicious purposes. For example, `drive.google[.]com` is clearly a legitimate site, but quite a lot of malware is hosted there.
- The domain is legitimate but has been compromised and used for malicious purposes. The compromise could have happened at the server the domain is pointing to, but also at the DNS level. In the latter case, using a site like VirusTotal can help you check DNS history.
- The domain was registered and set up for malicious purposes.

When registering a domain, malicious actors don't say they are going to use it for phishing or malware. So, if you want to understand whether a domain has been set up for malicious purposes, you will have to do some investigation.

Domains registered for malicious purposes:

- are usually registered relatively recently
- often look very random (e.g. `vniopquiopvqr[.]com`) or resemble legitimate ones (e.g. `internews-official[.]org`)
- more often use unusual top-level domains such as `.top` or `.surf`
- often don't have an MX record set up or use the default one from the registrar

DNS lookups for a domain aren't visible to the entity who runs the domain<sup>20</sup>, but active probing of a web server or mail server by connecting to that domain is. Keep this in mind if you don't want your investigation to be visible, and use a VPN or Tor to hide your identity!

---

<sup>20</sup> They may notice a lookup if it is for a very specific subdomain, but this is an edge case.





## Defanging

When dealing with potentially malicious domain names or IP addresses, you don't want someone to accidentally click on them (note that quite often, they are automatically turned into names). You also don't want some security software scanning the context in which the artifacts are shared (for example, an email security product) to detect them as malicious and raise an alert.

Therefore, it is considered good practice to **defang** domains and IP addresses by placing square brackets around all the dots or at least the last one. In practice, the latter is almost always good enough, which is why this has been done in this guide. So instead of `internews.org`, we write `internews[.]org` and instead of `127.0.0.1`, we write `127.0.0[.]1`.

The artifacts in the previous paragraph and many of the ones in this guide aren't malicious and could, therefore, leave out the square brackets. However, it is a common practice to just defang them all.

Many tools and services where you can enter domains and IP addresses, including VirusTotal, will automatically remove square brackets, so you don't have to do that before searching for something.

**Question 7.1.** Why don't you need to defang file names? (See the appendix for the answer.)

## Hashes

A **cryptographic hash function** is a mathematical algorithm that takes data of arbitrary size (for example, a password or the contents of a file) and turns it into a fixed number of bytes: the '**hash value**' or just '**hash**.' When you deal with malware samples, hashes are very convenient for many reasons.

The first is safety. You don't want to share malware in a way that it could accidentally run and cause damage. Sharing a hash of a sample is safer.

The second is size. A hash is small, so you can share it in an email, Signal message or on social media without having to attach the original file.

A third reason is that a hash serves as the sample's "fingerprint," making it easy to find other instances of it in a collection of malware samples. This can be very useful. Sometimes, you don't want to make more information about the sample public than it already is. When you share that you saw a malicious email attachment with a particular hash, people can check their own collection of attachments or repositories of malware samples like VirusTotal and see if they saw the same file.

When a malicious actor sends out multiple malware files in a campaign, it is fairly easy for them to modify each in a small way so that the hashes are completely different. However, in practice, they don't do that often, so hashes continue to be a good way to help people identify that they are seeing the same malware files (and thus the same adversaries) in a campaign of attacks.

It is important to note that the filename isn't part of the hashed data, so if you rename a file, the hash won't change. However, if you change one byte in a file, it changes.

Cryptographic hashes have the following properties:





- the hash can be computed quickly, even for very large inputs
- given a hash value, it is in practice impossible to calculate the original data
- given some data, which allows you to compute its hash value, it is in practice impossible to find other data with the same hash value
- the tiniest of changes to the data gives a completely different hash value

As a note, there are some other kinds of hashes that play a role in threat intelligence that don't share this latter property. These are called fuzzy hashes. You may, for example, notice SSDEEP on VirusTotal. This is a 'fuzzy hash'.

In practice, there are three relevant hash functions: md5, sha1 and sha256. The former two are older and slightly imperfect, though that is unlikely to affect your work. However, when you can choose a hash function, it is recommended to choose sha256.

A sha256 hash consists of 32 bytes (or 256 bits; hence the name) and is written as 64 characters using the digits 0 to 9 and letters a to f. Because of the above properties, the particular characters of a hash don't matter though: in your research it is just a string of characters.

**Question 7.2.** You can't just 'reverse' the hash to find the original data, given the second property above. However, if you find a hash and have no idea what it is, you can always use a search engine and maybe get lucky. What is the original data that resulted in the following hashes?

md5:d41d8cd98f00b204e9800998ecf8427e

sha1:da39a3ee5e6b4b0d3255bfe95601890afd80709

sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855

(The colon notation is common to indicate what type of hash something is, but as you see, the three hashes also have different lengths.)

**Exercise 7.3.** On your REMnux instance, create a text file, enter the text 'hello world' (without quotes or a line break), and save the file by calling it `test.txt`.

1 Then use the `sha256sum` command to calculate the sha256 hash of the file. Take note of the hash value.

Use `md5sum` and `sha1sum` to also calculate the md5 and sha1 hashes.

2 Now rename the file to `newtest.txt` by using the `mv` command. Calculate the sha256 hash of the new file. Confirm it is the same.

3 Now open the new file for editing, and turn the 'h' into a 'H' (lowercase to capital). Save it and calculate the new sha256 hash. Confirm it is different and unrelated to the original hash.

4 Go to [Malware Bazaar](#) (make sure you do this in a browser within REMnux), find its malware database and click on any file. Download the sample and unpack it (remember how it is common practice to put malware in a zip file with the password 'infected').

Unpacking a zip file normally works with the `unzip` command, but you may get an error. If you do, install the `7z` command by running



```
sudo apt-get install p7zip-full
```

and then unpack the file by running

```
7z x [file]
```

Then calculate the sha256 hash of the malware sample.

If you do this right, you'll notice the hash is the same as the file name. This is pretty common among malware databases!

## Using VirusTotal

With all this background information, let us look at VirusTotal, where you can compare domains, IP addresses, and URLs from your files with those uploaded by threat analysts around the world.

Before using VirusTotal, you have to understand how the site works. Its main functionality is free and doesn't even require registration. Creating a free account would give you some additional options, but we will not use them in this guide.

VirusTotal also offers paid accounts, which make it easier to search the huge collection of files for certain properties, such as a name, and to download them. Here is why this matters: assume your adversaries, especially the more powerful ones, have paid accounts and that they use them to look for files relevant to their targets. So be careful about uploading files to VirusTotal: *only upload files you are comfortable making public.*

## File checks and uploads

There are two reasons why you may want to upload a file. The first is to check its context and to learn more about it: is it malicious, what domains does it connect to when run, etc?

The second is to share it with the security industry. Most security companies are VirusTotal customers, and uploading a file is a simple way to share the file with all of them. It could help them detect the specific attack you are facing and stop it from affecting others.

It is worth noting here that most security companies see millions of malware samples a day, and most scanning – and the adding of detection – happens fully automatically. That's often fine, but if you want them to pay close attention to the file for some reason, make sure you also reach out directly!

When checking a file on VirusTotal, perform the following four steps:

1. Calculate the sha256 hash of the file as described above. Check the hash on VirusTotal. If it exists, the file has already been uploaded.
2. If it is not, decide whether you want to upload the file. Even if the file is malicious, it may have been designed specifically for the target and thus contain some personal information. In the case of a targeted file, uploading it to VirusTotal shows that it was opened and being analyzed. It is also okay to decide not to upload the file because you're just not sure!



- If you do want to upload it, consider whether the name of the file is something you are willing to share. If not, or if you're not sure, rename the file to `[sha256].[extension]`: keep the original extension (.exe, .docx etc), but rename the part before the dot to the sha256 hash. That's good practice (we saw that with Malware Bazaar above) and doesn't give anything away about the file.
- Upload the file using VirusTotal's web interface. Make sure you only upload the file itself and not a whole folder or a zip file containing it!

## The Detection tab

In the following sections, we will use the sample `971c5b5396ee37827635badea90d26d395b08d17cbe9e8027dc87b120f8bc0a2`<sup>21</sup>, which is a malicious .exe file used in targeted attacks by an Iran-linked threat actor called APT42. It was referred to in a [report](#) [wayback machine](#) by security company Mandiant (now owned by Google).

VirusTotal shows you a header with several 'tabs' underneath. The header includes the sha256 hash, the file size, and when it was last analyzed.

Underneath, the first tab is called detection. It shows the scanning results from dozens of antivirus engines. The word engine is used here deliberately: this part of the product usually only looks at the file itself rather than what it does.

Note that low detection rates are pretty common, especially for new files, and the number of detections should also be seen as an indicator of how known the file is among the security community, not how well a particular product would have stopped the threat in a real situation.<sup>22</sup>

The screenshot shows the VirusTotal detection tab for a file. At the top, there is a circular progress indicator for the 'Community Score' showing 54 out of 170. A red banner indicates that 54 security vendors and 2 sandboxes have flagged the file as malicious. The file's metadata includes the SHA256 hash, size (2.79 MB), and scan date (2022-11-08 16:13:00 UTC). Below this, there are tabs for 'DETECTION', 'DETAILS', 'RELATIONS', 'BEHAVIOR', and 'COMMUNITY'. The 'DETECTION' tab is active, showing a 'Security Vendors' Analysis table.

Vendor	Detection	Engine	Detection
Ad-Aware	Dropped:Trojan.Agent.FZTK	AhnLab-V3	Downloader/Win32.Paph.C1552293
Alibaba	Trojan:Win32/APosT.2f81a6ec	ALYac	Trojan.APosT.gen
Antiy-AVL	Trojan.Generic.ASMalwS.3E79	Arcabit	Trojan.Agent.FZTK
Avast	Win32:Malware-gen	AVG	Win32:Malware-gen
Avira (no cloud)	TR/Dropper.hwvbb	BitDefender	Dropped:Trojan.Agent.FZTK

False positives – harmless or legitimate files incorrectly detected as malware – do happen but are rare. In practice, anything more than a few detections suggests that a file is malicious.

A tip: if the file was last scanned some time ago, you can click on the curved arrow in the top left corner to have the file scanned again. Detection typically improves over time, and this gives you a more accurate picture.

<sup>21</sup> An example of referring to a file by its hash!

<sup>22</sup> This comes down to VirusTotal only using a stripped down engine of the file, not the full product. Compare this to a security guard who wouldn't have recognized a thief based on their appearance, but might still have stopped them from actually stealing something.

Finally, there are two things about the engines. Some are actually the same (for example, AVG was acquired by Avast some time ago; the products have long been the exact same), and they're only listed separately because the brands still exist. The detection names can usually be ignored: they are often generic, and when they're not, they are typically wrong.

Finally, note that if you are logged into VirusTotal, you'll see some detection rules listed above the antivirus results. These are some other detection rules, and unlike antivirus detection names, they can be helpful in understanding what kind of malware this is.

## The Details tab

The Details tab shows more information on a file, such as various hashes: sha256, md5, etc. Our file is a Windows executable (you'll see the file type listed). For other file types, the information here is different; for example, for Android files, it shows the permissions requested, which can be very helpful.

The Details tab tells you when the hash was first submitted to VirusTotal, which is a good indicator. It also tells you when the file was created and when it was first seen in the wild; these aren't particularly accurate, so don't pay too much attention to them.

You'll also notice different names under which the files were submitted. Sometimes, this helps you understand how the file was used, as seen in the following exercise.

**Question 7.4.** Someone you support receives a malicious file named `tax_information.docx`, and you suspect an adversary is targeting them specifically with a campaign made to look like it's from the local tax office. On VirusTotal, you will find the same file has also been submitted as `package_receipt.docx` and `birthday_gift.docx`. How could this information help you? (See the appendix for the answer.)

## The Relations tab

The Relations tab shows relations between the file you uploaded and other IoCs in VirusTotal: URLs, domains and IP addresses contacted when the file was run in a sandbox (more on which in a later chapter) as well as files that were downloaded from the Internet or created directly when the file executed.

You can click on these objects to learn more about them and possibly find links to other threats; this is called pivoting. We'll discuss this further later in the guide.

**Exercise 7.5.** The domain `update-driversonline[.]bid` stands out among those connected to this file (remember, we're still talking about `971c5b5396ee37827635badea90d26d395b08d17cbe9e8027dc87b120f8bc0a2`). There are many reasons to suspect this is malicious and some reasons to think it is not. Which ones can you think of?

The Relations tab could also contain other contexts, such as URLs from which this file was downloaded or archives (like a zip) that contain it. All of this can be helpful information to understand the threat!



## The Behavior and Community tabs

The Behavior tab discusses what the file does. For a Windows executable, this can be really helpful. However, for a beginning analyst, the information here can be quite confusing. Without any further understanding of what files do (for which you could, for example, use a sandbox, which will be covered in Chapter 10), be cautious of drawing conclusions purely based on this tab.

The Community tab holds comments about the file. Many comments are automatically generated and show more sandbox results. Some comments are added by humans; these are especially helpful. If you want, you can create a free account on VirusTotal and provide some context on the file and maybe even a suggestion to contact you if someone knows more. In the latter case, do keep in mind that this is publicly available, including by adversaries, so don't add anything that can be linked to you as a person if you consider that not safe to do.

## Other file types

VirusTotal can handle many different file types. The tabs and their content can be somewhat different depending on the file type.

Check the VirusTotal tabs for the following three examples and spend a few minutes on each to familiarize yourself:

A malicious Word document:

2382d4957569aed12896aa8ca2cc9d2698217e53c9ab5d52799e4ea0920aa9b9

An Android package (APK) file:

86acaac2a95d0b7ebf60e56bca3ce400ef2f9080dbc463d6b408314c265cb523

A macOS executable: 483b2f45a06516439b1dbfedda52f135a4ccdeafd91192e64250305644e5ff48

**Question 7.6.** In the examples above, look around at the tabs and try to answer the following questions:

1. Does the Word document contact any suspicious domains? If not, does it make any suspicious connections in other ways?
2. Does the Android file request permission to listen to the microphone?
3. Without using a search engine other than VirusTotal, can you find a report about the macOS executable?

(See the appendix for the answer.)

## Hostnames and domains

Now search VirusTotal for the domain `update-driversonline[.]bid` that we saw in our original sample. You can also go to that sample on VirusTotal and click on that link in the relations tab.

You'll see that VirusTotal also has plenty of information on domain names. The Detection tab will look familiar. Detecting domains is less of a hard science than detecting files, though, and there isn't always a clear distinction between 'good' and 'bad' domains (remember the four kinds of domains we discussed in this



chapter, in the section on Important concepts for threat intelligence), but here too: the more vendors detect it as malicious, the more likely it is.

The Details tab has more information on the domain, including whois information, which tells you when it was registered and, occasionally, by whom. The Community and Relations tabs will look familiar; the latter is particularly useful. You can use it to pivot and also find subdomains.

**Question 7.7.** The domain was contacted by three files, including our original sample. Do you think the three files are related? Why do you think that? (See the appendix for the answer.)

## IP addresses

VirusTotal also contains information on IP addresses. The Relations tab here is very helpful: it shows that domains have had their A record pointing to an IP address. This can be really helpful when trying to find other domains linked to a particular domain.

**Question 7.8.** Use the VirusTotal entry for `update-driversonline[.]bid` and look at related IP addresses. Can you find other domains you think were used in the same campaign? (See the appendix for the answer.)

## URLs

Finally, VirusTotal also has pages on URLs, but they rarely add much to what the corresponding domain or hostname pages show. Because URLs are often unique, adding them to VirusTotal can give an adversary information about you analyzing it. Therefore, it's best to just look for the hostname.

Checking a hostname or a domain name is almost always 'fine': VirusTotal actively scans domain names, so you don't have to worry about 'uploading' a new domain. The only case in which you may want to be a bit careful is if you have a very specific hostname, such as `[long string].domain`, which may be unique to this particular target. That is rare, though.

## Threat hunting

Threat hunting is a proactive activity where you are looking for threats rather than analyzing existing ones. Sometimes, threat hunting involves looking for threats targeting a specific organization or community without any specific focus, for example, by looking for newly registered domains that use your organization's name, which could mean the threat actor is registering them for use in a phishing campaign.

More often, threat hunting starts with an existing threat, such as a malicious file, and then looks for related artifacts (other files, domains etc.) to get a better picture of the threat and the actor behind it.

VirusTotal is a great tool for threat hunting, and in the previous section, we did that a little bit by looking for related files and domains. VirusTotal's paid accounts are really great for this, as they open up many more



possibilities for threat hunting. You can pivot to investigate more indicators you find and also use 'YARA rules', a method to search files in a large collection that is particularly helpful when looking for related malware.

It may surprise you how much you can achieve with free online tools without knowing how to reverse engineer malware<sup>23</sup> or how to write YARA rules. There are two important caveats, though.

The first is to be wary of bold conclusions. It's tempting to get very excited<sup>24</sup> when you link some malware to a known actor, especially when it is an advanced actor linked to some government. Sometimes, these links are indeed there. At many other times, they aren't. For example, both actors may have shared third-party infrastructure, or there has been a deliberate attempt to misguide researchers. Another possibility is that domains used by both threats ended up pointing to the same sinkhole (more on sinkholes shortly).

The second caveat is don't forget the goal of your investigation. You don't have to look for similar threats to each threat you investigate. If the person reporting the event didn't open the malicious attachment or the attachment wasn't targeting them or their organization specifically, it's okay to close the investigation and focus on other things.

## Targeted and non-targeted threats

Digital threats such as malware and phishing have traditionally been divided into targeted and non-targeted threats. In the former scenario, the threat is aimed at an individual user or organization or maybe a very small number of them. In the latter scenario, any user could have received the threat, and the actors behind it just hope for some of those receiving it to 'fall for it.'

The challenge is that in practice, many non-targeted threats can still appear somewhat targeted. First, it is very common for a non-targeted campaign only to target a particular country or region. This allows the actor behind it to write messages in the local language and add local context that makes it more believable.

Secondly, threat actors often use data from many compromised email accounts when sending malicious emails. This way, they can make an email look like it was a reply to something you had previously sent or like it came from one of your contacts. This would make an email seem very targeted, even when this isn't the case: the actors are able to automatically send a large number of emails, each of which appears quite personal to the recipient.

Finally, threat actors, especially those engaged in cybercrime, have found a hybrid between targeted and non-targeted threats, where a non-targeted threat results in a compromised account, often within a large organization. Access to this account is then sold to another actor, who uses the access in a more targeted way, for example, by deploying ransomware throughout the network. Even though the latter threat was very targeted, the original threat wasn't!

The vast majority of digital threats aren't targeted, and this is true even for threats against those who also face very targeted threats.

---

<sup>23</sup> Reverse engineering is the process of learning what a piece of software, often malware, does from the compiled bytecode

<sup>24</sup> It is okay to get excited about these things, even if the threats are serious and affect real people in often serious ways! Just don't let your desire to help people and groups be driven by what threats excite you most. The vast majority of threats are very mundane.





**Question 7.9.** An email with a malicious attachment addresses the recipient, an employee of a civil society organization, by their first and last names and is made to appear from another employee at the organization. Can you give a reason why this may be an automated, non-targeted threat? (See the appendix for the answer.)

## Sinkholes

Most malware is controlled by a server or set of servers, which are controlled by the threat actor. This server or set of servers is called the command and control (often referred to as C&C or C2). Most malware requires an internet connection so that it can take orders from its C&C. Malware tells your device to connect to one or more domains that point to the C&C server's IP address. Pointing to a domain instead of an IP address allows the actors to switch to a different server if the original one were to become unavailable to them.

To stop a C&C operation and the malware it is running, a law enforcement agency or a security company may show a domain registrar evidence that the domain was used for malicious purposes and convince the registrar to let them take over the domain. They may then point the domain to a server they control (often referred to as “**sinkholing**” the domain). This means that every time the malware tries to connect to its C&C server, it connects to the sinkhole domain, giving its operators a good insight into the threat. Sometimes, the law enforcement agency or company can then send commands to neutralize the malware if they develop enough understanding of how the C&C works.

Some malware uses a domain-generating algorithm (or DGA) to generate new domains, often on a daily basis. This sneaky move makes it harder for an analyst to identify that multiple malware samples are pointing at the same C&C server because they can't search for identifying domains as easily. The malware actors register these new domains as long as their campaign runs, and don't have to worry too much about domain takeovers or a domain being blocked by security products. In many cases, the DGA is 'cracked,' allowing researchers to predict what domains are going to be used, proactively register them, and point them to a sinkhole.

When investigating a malicious domain, it is thus important to keep in mind that a domain you see may have been sinkholed. There isn't a public list of sinkhole IP addresses – if there were, malware authors would simply make the malware not connect to any IP address in that list – and it's not always obvious when an IP address is a sinkhole. Many domains from seemingly unrelated campaigns pointing to the IP address are often a big giveaway that it is a sinkhole. In case of doubt, enter the IP address in a search engine, check the 'community' tab on VirusTotal, or ask around!

**Question 7.10.** Using a sinkhole to send commands to neutralize malware infections is somewhat controversial. Can you think of reasons why this is the case? (See the appendix for the answer.)

## Other tools

VirusTotal is a great tool for threat analysis and hunting. It isn't the only one, though. Perhaps the most useful tools of all are search engines. Whether you prefer Google or one of the many great alternatives, you may be surprised at how much you can often find by searching for a domain name, IP address, file hash or other artifact.





The same is true for social media, in particular Twitter, though it should be noted that at time of writing (early December 2022), many security researchers have left Twitter and moved to Mastodon. The sharing of IOCs was pretty common on Twitter and might become so on Mastodon. A big advantage of finding an IOC on social media is that you can respond to the person posting it and ask questions or add your own comments.



## Chapter 8:

### Android and Android malware

Android is an operating system for mobile devices. It is one of the two major OS options, with Apple's iOS being the other and its main competitor<sup>25</sup>. If you want to know more about Android and its relation to the Android Open Source project, the [Wikipedia article](#) is helpful.

As a note, the exercises in this chapter require that you have access to an Android phone. Any phone will do, including your own personal device. You will not be asked to make any modifications to the phone. If you do not have access to such a phone and still want to do the exercises, it is possible to install a virtual machine version of Android in VirtualBox, though that can be a bit fiddly.

#### Android versions and a note on Google

As of the time of writing, this content was up to date. However, Google, which develops Android, continuously updates the operating system. In general – but not always! – these updates improve privacy and security as well as your ability to control the security and privacy settings. Keep this in mind when reading this text, especially if you read it quite a while after it was last updated. To learn about the latest updates, you can consult the [official Android website](#) and its [developer guide](#).

Also, keep in mind that most of Android's external appearance varies a lot between phones, even when phones use the same Android version. Sometimes, the way things work 'under the hood' varies too. It is possible that things discussed in this chapter are slightly different on the phone you're using, though they have been tested on various phones.

Another thing to keep in mind is that this chapter focuses on malicious apps and was written with the assumption that Google itself isn't in the user's threat model. If you're not backing up data to Google's cloud unencrypted, Google can't read Signal or WhatsApp messages. However, Google can see what apps you have installed on your phone and could be forced to hand this data over to authorities.

#### How likely is Android malware?

Android malware is real, and it's definitely something any civil society organization should include in their threat model. This chapter helps you analyze Android devices for potentially malicious apps.

However, before you start, it may be a good idea to consider how likely Android malware is. Android malware, though more common than iOS malware, is relatively uncommon. The most common Android malware engages in advertising fraud and has relatively little impact on the device.<sup>26</sup>

This is due to how Android apps work. Getting a user to install a malicious app is not easy – especially when the attacker wants a particular individual to install it. If the attacker isn't exploiting vulnerabilities in the Android operating system (which is rare and may require a lot of technical sophistication), installed apps are restricted in what they can do. So before concluding there must be malware on an Android device, it is worth considering

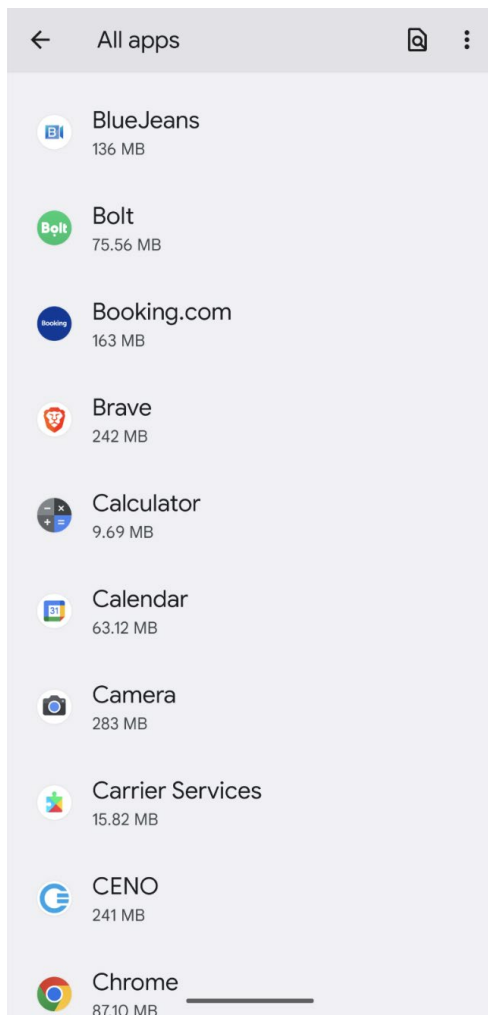
---

<sup>25</sup> They don't run on the same devices though: an iPhone can't run Android and a phone made for Android can't run iOS

<sup>26</sup> Clicking ads in the background, or showing a large number of ads to the user



other options (for example, an account compromised through a stolen password), in particular, if the action committed by the suspected actor doesn't seem sophisticated enough to be a malware campaign.



**Question 8.1.** A journalist receives an anonymous email saying the sender knows she was in a particular bar last night. This was indeed the case, and she finds the email, understandably, very disturbing. With just this information, how likely do you consider this was malware on her phone that shared her location? (See the appendix for the answer.)

## Android apps and permissions

Programs on Android are typically referred to as 'apps.' As an Android user, there are apps you'll know because you use them regularly and they have icons on your home screen, but there are also apps you are less familiar with, such as apps that were installed by the phone manufacturer (Samsung, HTC, Huawei etc.) or systems apps that provide certain functionality but are otherwise hidden from the user.

For privacy and security, the Android operating system uses permissions to limit the resources that apps can use. For example, an app cannot access the contacts stored on the phone unless it has been granted permission to do so.

With the possible exception of some very advanced malware (more on which in a later chapter), this is also true for malicious apps, whether they are installed from Google Play or '**sideloaded**' onto the phone.<sup>27</sup>

It is possible to see the permissions granted to an app by going through the settings and finding the individual app in the app list. This

can help you determine if an app is, for example, tracking your location or listening in on your conversations: it will need your permission to do that.

A natural way to look for malicious apps on an Android phone would thus be to look through all the installed apps – the settings on the phone allow you to do that – and then see if any of them have unusual permissions. There are some downsides to this approach:

- A phone easily has 100+ apps installed. It is very time-consuming to go through them all.
- Malicious apps often masquerade as something totally innocent (such as a calculator app), and you might miss them during a check of the apps.
- There are many apps installed on an Android phone the owner has never heard of, usually because they came with the operating system. People sometimes get anxious when seeing these apps ("My phone has a car app. I don't even have a car! It must be spyware.")

<sup>27</sup> Sideloading is the act of installing apps outside the official app store, in this case Google Play. It is possible on Android, but requires the user to click through some warnings. It thus requires some social engineering, or for an adversary to have short-time access to the unlocked device.



A very different way to approach this is to connect the phone to a PC and scan the phone with special software designed to detect malware. This can be helpful in some cases, but it also has downsides:

- It requires the analyst to have physical access to the phone, while most support is remote.
- It puts the bar for analysis fairly high in terms of tools, knowledge and experience. Whereas most issues can be detected even by the phone user if they are guided through it, these advanced tools are less accessible for everyday users.
- Not everyone, and certainly not every high-risk individual, will trust someone else connecting their phone to someone else's computer.

## A better approach: looking at app permissions

There is, however, a better approach, and that is to use the Permissions Manager in the Settings of an Android phone. This feature shows most<sup>28</sup> of the permissions apps can request and, per permission, which apps have been granted these permissions. In some cases, it also shows some granularity for the permission.

For example, if you are worried about the possibility of a malicious app on the phone listening in on the microphone, you can see which apps have been granted this permission and check if they are legitimate apps. Note that, unfortunately, even legitimate apps sometimes ask for too many permissions. Just because a flashlight app requests your location doesn't automatically mean it's malware – though you still probably don't want it to have this permission.

The way to go about this is to look for the Permissions Manager in the settings and then check the individual permissions.

To know whether an app's request for permission is a concern, consider the way you usually use the app. If you use Location, Microphone, SMS, Contacts, and Files with that app, the permissions may be reasonable. When you don't use those permissions with an app, the use of those parts of your device might be a sign something is wrong, so look for unexpected permission requests.

The goal here is not to fully understand what the apps are doing but to limit your investigation to a small number (sometimes zero!) of unknown apps that you may want to investigate further. As a bonus, you will also discover which legitimate apps require permissions you or the person whose phone you are investigating feels uncomfortable with; you may be able to turn these permissions off.

Sometimes, it is not possible to change the permissions of a pre-installed app (the official Android website has some [information](#)<sup>wayback machine</sup> on that). Because many people are less familiar with such pre-installed apps, they may suspect an app like this to be doing something malicious. If this is a phone you aren't familiar with, you may have to do some research into these apps to understand what they do and to comfort the user (and maybe also yourself).

## Location

Apps that have been granted this permission can access the phone's GPS location. Obvious examples include maps apps and ride-hailing apps: they need the location to function well.

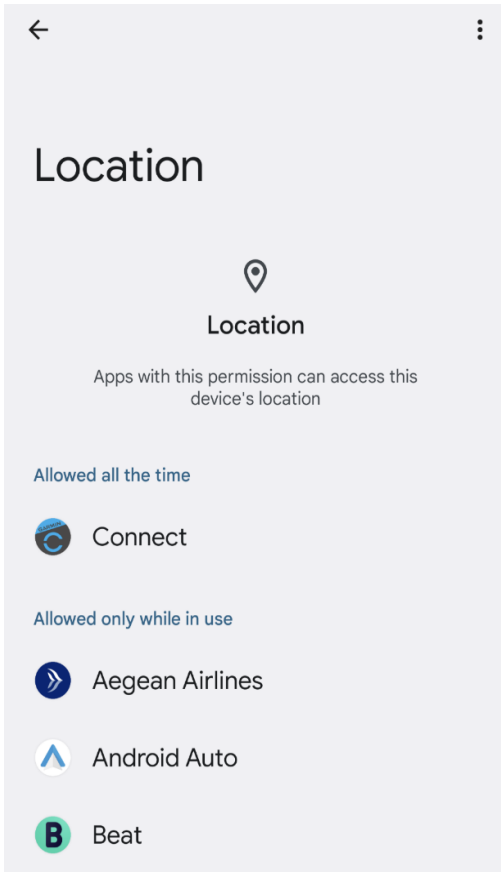
---

<sup>28</sup> Some permissions are very obscure and not quite relevant in this context. Two others are discussed later in the chapter.



An app can be granted access to the location all the time or only when the app is being used. It may also have to ask every time it requires the location. Apps that do things secretly, such as spyware, are more likely going to need permission all the time. Be particularly wary of apps having been granted those kinds of permissions!

Whether the companies behind these apps can and should be trusted with your data is beyond the scope of this chapter. However, regardless of whether you trust the companies, there are several ways an adversary may obtain access to the location data obtained by the apps.



The most obvious ones are having access to the phone and thus the app itself, or to a corresponding online account, and having the ability to view the location history; Google, for example, provides this for Google Maps. An adversary may also make legal requests for this data through a subpoena.

If you thus want to assess whether someone's location data is safe, you should look into how easy it is for them (and thus for anyone successfully impersonating them) to access location history and how the company responds to data requests; many larger companies publish transparency reports.

Regardless of that, for most of these apps, it should be fine to set the permission to be only used when the app is in use or to ask permission every time. This means the app doesn't get access to location data all the time.

Finally, it is worth noting that using the GPS is heavy on the battery: someone whose location is being tracked all the time through GPS will almost certainly notice this because their battery runs low quickly. However, a battery may run low quickly for many reasons, so this is not conclusive proof of malware on a device.

Finally, note that GPS is very precise. It can show a location as accurate as 30 cm (about 1ft). In some cases, someone is worried about leakage of their coarse location, for example, in which city or country they are. This information can leak in many explicit ways (e.g., through the IP address) and sometimes also in implicit ones (through information shared publicly). It is beyond the scope of this chapter to show how this can be prevented, but it is very challenging!

## Microphone

Apps that have been granted this permission can access your microphone and thus record anything you say, as well as ambient sounds.

There are many legitimate reasons for apps to use this, for example, apps that allow you to send voice messages, apps that record videos or those that can be accessed through voice commands.



Like the location permission, access to the microphone can be granted all the time, only when the app is running, or by making the app ask every single time. There aren't many good reasons for an app to have this permission all the time. An app that has been granted this permission is thus a big red flag.

## SMS

An app that has been granted permission to read SMS messages can spy on these. Depending on whether the user actively uses SMS to communicate, this may be a concern when it comes to spyware.

However, SMS is also used for multi-factor authentication, so an app with access to SMS can quietly read codes sent this way<sup>29</sup>. Related to this, some apps use SMS as a way to confirm the device's phone number and request this permission to avoid the user having to enter a code manually. This permission can thus be used maliciously by an app that wants to impersonate the user.

The phone's default SMS app naturally has this permission, and so have some other apps that can be used to handle SMS, such as Signal.

Because SMS arrives in the background, apps with access to SMS will need this all the time. There is no option to only grant this permission while the app is running or to ask for this permission every time.

## Contacts

Apps with this permission can access the contacts on a phone. It is used by the phone app itself, as well; for example, by messaging apps such as Signal and WhatsApp that identify users based on their phone numbers. Contacts typically include phone numbers, but depending on how the phone is being used, it may also include email contacts. For many high-risk users, their list of contacts is very sensitive information.

As with SMS, this permission is either allowed permanently or denied.

## Files

This permission gives apps access to files on the device. A file manager or a security product are examples of legitimate apps requiring this permission. The Files permission has changed a lot in recent Android versions and has become more granular. If you want to look for a malicious app, at least initially, it is best to assume that if the app has permission to access the files, it can access any file stored there.

Note that this does not give apps access to chats from Signal, WhatsApp, etc.

This permission is also either allowed permanently or denied.

**Exercise 8.2.** For each of the five listed permissions, check on an Android phone which apps have been granted this permission.

**Exercise 8.3.** The Permission Manager on Android lists more than these five permissions. Look through the other ones and think whether they could be used by malware.

<sup>29</sup> This is one of several reasons why using SMS for multi-factor authentication is less secure than other options, such as an authenticator app or a hardware token



There are two special kinds of permissions that are listed separately as they are not in the Permissions Manager: accessibility and notifications. They are very relevant, though, because they are often abused by malicious apps. How these settings can be accessed varies between phones: if you can't find it easily in the Privacy menu in the Settings, use the search function in the Settings.

## Accessibility

Accessibility makes apps on a phone more accessible for people with certain disabilities, for example, by helping them integrate with screen readers. An app registered as an accessibility service is able to control the phone to a certain extent: it can click buttons, register when a button is clicked, fill in text forms, or register what has been filled in forms.

Several legitimate apps that don't provide services for disabled people still register as an accessibility service. One example is password managers, which require this feature to fill in stored passwords into apps.

You can find which apps are registered as an accessibility service by looking for 'Accessibility' in the phone's settings and then looking for installed services.

Note that as of Android 13, released in August 2023, only apps installed through Google Play will be allowed to request accessibility permissions.

## Notifications

Notifications are the pop-ups you receive on the phone when you get a new message or when there is some other kind of alert. Which notifications you receive depends on the settings of the individual app.

Apps can request access to notifications, which means they can see all notifications and the app that sent them. An app that syncs a phone to a smartwatch and displays notifications is an example of a legitimate app that requires access to notifications.

Malicious software can and sometimes does use this to spy on the user's activities, even if notifications only show incoming messages, not outgoing ones.

**Exercise 8.4.** As before, check on an Android phone to see which apps have been granted notification and accessibility permissions.

## Analyzing Android malware and advanced Android malware

In the previous chapter, you learned how VirusTotal can teach you quite a lot about malware by pivoting. For example, you may find that a piece of Android malware connects to a certain domain, to which another piece of malware also connects, and this malware has been analyzed. That suggests that the file you're analyzing might be a different version of that malware, though, as always, there are many caveats.

There is also [apklab](#), a service by security company Avast. It is like VirusTotal but more tailored towards Android and can be really helpful in analyzing Android malware. The site requires an account, which is free but needs to be approved by its maintainers. We recommend requesting an account.



You can also run the file in a sandbox (more on this in Chapter 10). This is often enough to understand what a particular file does.

For a thorough understanding of analyzing Android malware, there is an excellent [course](#) from Google researcher Maddie Stone.

It is also important to note that there is some advanced spyware that hides so deeply on phones that it doesn't run as an app. Doing forensics on phones looking for that kind of spyware is really complicated, especially for Android phones. You will likely need to work with experts, such as those at Citizen Lab or Amnesty Tech, until you build your own skills.

We hope to expand further on analyzing Android malware in future additions to this guide.





## Chapter 9: Email forensics

The importance of email for any organization or individual is clear, as are, probably, the threats that exist against email, such as account hijacking, or threats where email is the carrying vector, such as phishing.

Content on email spans across two chapters in this guide: this chapter focuses on how to analyze an email that has been received and to determine whether it is legitimate. The next chapter looks at email attachments, which may be malicious, and how to analyze them.

This chapter helps you determine whether an email was sent by the person or entity it claims to come from. You'll learn how email works and how to analyze the various parts of an email. The exercises help you familiarize yourself with various tools that you can use for email forensics.

As this guide helps you detect spoofed emails or emails sent from maliciously set-up accounts, it is important to note that once it has been determined that an email wasn't spoofed, it is often<sup>30</sup> not possible for the recipient to determine whether it was sent from a compromised account. The only way to make that distinction is to contact the supposed sender through another channel, such as a messaging app, and confirm that the email was indeed sent. Always keep this in mind when performing email forensics!

Email goes back to the early 1980s, when the Internet was a very different place and security wasn't much of a concern. As a consequence, email in its most basic form doesn't support encryption, authentication or integrity, and the latter means you can't simply trust an email to come from who it claims to come from.

However, it is good to be mindful of the fact that spam filters<sup>31</sup> of various kinds, mostly running on mail servers, together with protocols such as SPF, DKIM and DMARC, serve as informal enforcers of the 'rules.' So, while in email forensics, you will sometimes deal with fake artifacts of email, the commonly repeated myth that you can simply forge any part of an email and have it delivered isn't true.

### SMTP

**SMTP** (the Simple Mail Transfer Protocol) is the protocol used to send email from one machine (often a mail server) to another.

The following example comes from [Wikipedia](#) and shows a short email being sent from `bob@example.org`<sup>32</sup> to `alice@example.com` with `theboss@example.com` in `cc`<sup>33</sup>:

---

<sup>30</sup> In some cases, there may be circumstantial evidence showing that an account was compromised, for example traces of a mail client that the account owner never uses. This is not generally the case though.

<sup>31</sup> Often called 'email security product' these days. Its purpose is to keep spam and other malicious emails out.

<sup>32</sup> `example.org` and `example.com` are domains to be used for this purpose, which is why I don't need to defang it

<sup>33</sup> `Cc` is short for 'carbon copy' and shows who is copied in the email. There is no difference in how an email is handled for recipients of the email and those `cc'd`, but some mail clients make a small distinction in how emails are displayed. `Bcc` is short for 'blind carbon copy'. If someone is `bcc'd` in an email, the other recipients won't be able to see their email address



```
R: 220 smtp.example.com ESMTP Postfix34
S: HELO relay.example.org
R: 250 Hello relay.example.org, I am glad to meet you
S: MAIL FROM:bob@example.org
R: 250 Ok
S: RCPT TO:alice@example.com
R: 250 Ok
S: RCPT TO:theboss@example.com
R: 250 Ok
S: DATA
R: 354 End data with <CR><LF>.<CR><LF>
S: From: "Bob Example" bob@example.org
S: To: "Alice Example" alice@example.com
S: Cc: theboss@example.com
S: Date: Tue, 15 Jan 2008 16:02:43 -0500
S: Subject: Test message
S:
S: Hello Alice.
S: This is a test message with 5 header fields and 4 lines in the message body.
S: Your friend,
S: Bob
S: .
R: 250 Ok: queued as 12345
S: QUIT
R: 221 Bye
```

You do not need to remember the details of SMTP for doing email forensics, and if you've already forgotten it, that's fine. It is included because seeing it at least once is helpful for a general understanding of email. However, if you find it too intimidating for now, just skip it and move to the section on headers, which is more hands-on!

The sending mail server (S) makes a TCP connection to the receiving one (R) on port 22, and when the recipient accepts the connection, it says so by a 220 return code<sup>35</sup> and also shares some details, such as the server name and the software it is running (Postfix in this case).

The receiving mail server then sends the **HELO**<sup>36</sup> message in which it shares its hostname. The recipient responded with the return code 250, indicating it was okay. If this sender was blocklisted based on the hostname or the sending IP address, it might respond with a different code and close the connection.

---

<sup>34</sup> It may be a bit counterintuitive that the conversation starts with the recipient (Alice) talking, but this response is to a TCP connection being started by the sender (Bob)

<sup>35</sup> You don't need to know return codes, but in case you're curious: they are always three digits and the first digit is the most significant one. 2xx means all is good; 3xx means we're not done yet; 4xx indicates a temporary issue and tells the sender to try later; and 5xx means a permanent issue, such as the email being blocked as spam. Quite often though, a spam filter will accept a message and either quietly discard it or put it in a spam folder. The 4xx code is sometimes used to see if the connection is retried, which a legitimate sender will but spammers often won't. The types of return codes are somewhat similar to HTTP return codes, which you may be familiar with and in any case will learn about in Chapter 11.

<sup>36</sup> Short for 'hello': in the early days of email, commands were always four characters



Then the sender sends the **MAIL FROM** address. This is the email address of the sender. However, it is not always the same as the From: email address displayed in a mail client<sup>37</sup>. More on this in a bit.

Again, the receiving mail server accepts the message, after which the sender sends the email addresses of the recipients in **RCPT TO** messages, one at a time. In this case, there are multiple recipients on the same domain, so the email is only sent once. Another 250 return code shows that, again, this was accepted, but here, a mail server may respond differently to say a recipient doesn't exist.

Then, the sender sends the **DATA** command, and the 354 response code indicates that the receiving mail server is ready for the email to be sent.

The email is then sent, one line at a time, until a single dot indicates the end of the email<sup>38</sup>. The email is accepted (note the 250 return code), and the receiving mail server helpfully shares the internal ID given to the email. Sometimes, a spam filter may detect spam and respond with a different code. The QUIT command is sent to say there are no more emails to be sent.

You are unlikely to look at SMTP transactions during your forensics work, but if you did, you'd probably notice two main differences with the example above.

The first is that instead of HELO, you'd almost certainly see **EHLO**, which is an extended version of HELO, allowing for some extra commands.

The second difference is that one of these commands, **STARTTLS**, is then used to encrypt the remainder of the connection, preventing the email from being read as it travels between one mail server and another. STARTTLS isn't perfect and does not have end-to-end encryption, but it prevents governments and other powerful entities from reading emails by just tapping the network connection, as some have done in the past.

**Question 9.1.** Why doesn't encrypting an SMTP connection offer end-to-end encryption? If you don't know what end-to-end encryption is, don't worry; just look it up! (See the appendix for the answer.)

## POP3 and IMAP

**POP3** (the third version of the Post Office Protocol) and **IMAP** (the Internet Message Access Protocol) are two protocols used by a mail client that runs on a device (such as Outlook, Thunderbird or a mobile email client) to retrieve emails from a mail server. You may come across these protocols but are unlikely to need to understand them to perform email forensics.

IMAP is the more modern of the two and allows for a mail server and one or more mail clients to remain synchronized. For example, you can use Gmail in a browser, but if you turn on IMAP, you can also read emails in Thunderbird on your desktop. If you read an email in the browser, it will show up as read in Thunderbird and vice versa; the same applies to moving emails to different folders. IMAP doesn't handle sending emails; for this, Thunderbird does use SMTP.

<sup>37</sup> The mail client is the program in which emails are read and written. It could run inside a browser (think Gmail or Protonmail), on a desktop (Outlook, Thunderbird) or on a mobile device.

<sup>38</sup> Should an email for some reason contain a single dot on a line, the sender would send two dots. If there were two dots on a line, it would send three, etc.



If there is a concern about an unauthorized third party having access to emails, it is important to check if POP3 and/or IMAP are enabled on the email account. This is typically set on a web interface.

**Question 9.2.** Why can't a mail client use SMTP to retrieve email from a mail server? (See the appendix for the answer.)

## Email headers

An email is split into two parts: the headers and the body. The **headers** are everything up to the first empty line of the message text; the body will be discussed later.

Each header consists of a key followed by a colon and the value of the header. The value of a header may be split over multiple lines, in which case subsequent lines are slightly indented. Don't worry if this is confusing; it will become clear when you look at some headers, which you will do in a bit!

Headers are added by the mail client in which the email is written, and then every server the mail passes through adds at least a Received header and possibly more headers.

Some headers may occur more than once in an email, while other headers occur only once. Different emails have different headers, but there are many that occur in (almost) all emails. Here are some common headers:

- **From** contains the email address of the recipient, often but not always together with a displayable name. It may look something like this:

```
From: Bob <bob@example.org>
```

But other forms are possible too.

- **To** (and **Cc**, **Bcc**) contains the recipient(s) of the email. Again, this may be a simple email address, but it often also includes a displayable name.
- **Subject** contains the subject of the email that is displayed in the mail.
- A **Received** header is added by each mail server that handles the email. Received headers are always added to the top of the email, so you can follow the email path by looking at the Received headers from bottom to top. Emails can easily have five Received headers and often have many more than that: most emails are handled by multiple servers at both the sending and the receiving side, and sometimes there are also servers in between, such as that of a third-party spam filter.

Unfortunately, there is little uniformity in the structure of Received headers. They show the IP address of the mail server and the time the email was handled by this server. Often, they include more information, such as the recipient's email address, the EHLO domain and the reverse DNS of the sending IP address, the internal ID given to the email (which can be helpful if you're trying to trace an email in server logs) and details on the security of the connection.

- **Return-Path** is present in most emails and includes the MAIL FROM address from the SMTP transaction. More on this below.
- **Reply-To** is an optional header to indicate where replies will go, in case this isn't the From: address (which is the default).



- **Content-Type** tells you what type of content the body contains, such as plain text, HTML or multiple parts. In the latter case, the divider between the parts is also shared. More on this later.
- **Content-Transfer-Encoding** is sometimes present to denote the encoding of the body, used in particular to encode non-ASCII characters<sup>39</sup>. Common encodings include base64 and quoted-printable, more on which later.

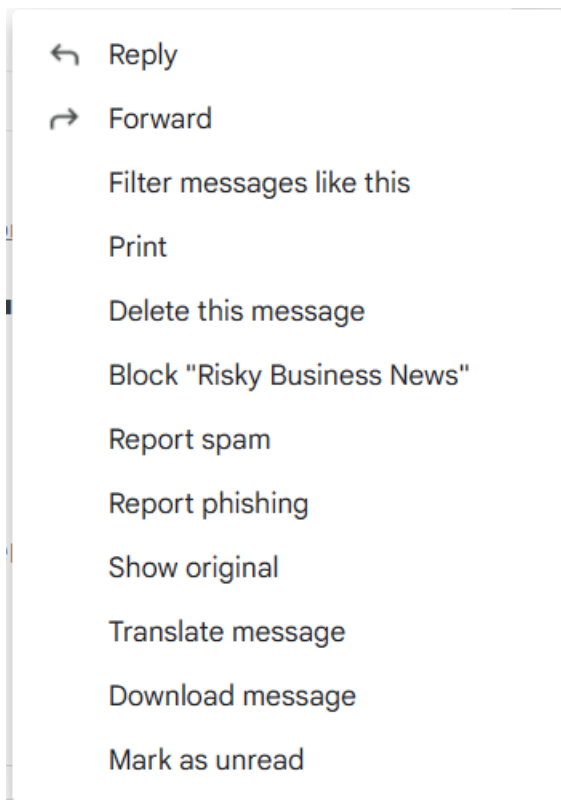
Sometimes, the content of a header is encoded, often because it contains non-ASCII characters. It will look, for example, like this

```
Subject: =?utf-8?Q?hello?=
```

In this example, the header is UTF-8 encoded. There are [online tools](#) to decode such headers (the linked one is just an example; should it not work, you can easily find others).

## Obtaining email headers and body

Most email clients allow you to see the full email headers or the full headers and body (also referred to as the 'raw email' or 'original email'). More on the body later.



In **Gmail**, including email for any Google Workspace<sup>40</sup> account, you can access the full email by clicking on the three vertical dots on the top right corner of an email and then selecting 'Show original.' This will open a new tab where you can see the headers and body – though not always the full body (attachments are sometimes left out).

In the new tab, you can download the full email through the 'Download' button. If you're only interested in the headers and the main part of the body, copying this to the clipboard works, too.

In the top part of the tab, you also see the most important headers, as well as authentication results for SPF, DKIM and DMARC. More on what this means later.

There are two things to note, especially if you help someone obtain the headers remotely. First, if emails are grouped by thread, the user will need to click the three dots on the specific email in question.

Secondly, here and elsewhere, opening headers in a new tab might be blocked by the browser, and you'll need to allow pop-

ups from this domain to be opened. A warning will likely be displayed in the address bar, but that might not be immediately obvious if viewing the headers doesn't appear to work.

<sup>39</sup> Think of these as characters with accents or using scripts or alphabets other than Latin

<sup>40</sup> Google Workspace is Google's corporate offering, used by many civil society organizations for their organizations email. Google regularly changes product names so in the future, this name may have changed.



In **Protonmail**, if you click under the three horizontal dots underneath an email, you will find the option to 'View headers.' This gives you the headers as well as some PGP information relevant to Protonmail that you can ignore. It will not show you the raw body.

In **Yahoo Mail** in the browser, you click on the three horizontal dots above the email. Then, 'view raw message' opens the full message (headers and body) in a new tab.

In **Outlook** on the desktop, you need to double-click on the email so that it opens in a new window. Then you choose 'File' in the menu, and at the bottom, you'll find 'Properties.' If you click that, a new window opens from where you can find the headers under 'Internet headers.'

In **Outlook Web Access**, the online version of Outlook, you can find the headers by clicking on the three horizontal dots next to the email, then clicking 'View' and then 'View message details.' Again, this only gives you the headers.

In **Apple Mail**, you can get the full email (headers and body) by opening an email and then going to 'View' in the menu, from which you click on 'Message' and 'Raw source.' (If you're only interested in the headers, 'All headers' gives you those.)

In **Thunderbird**, you 'View' in the menu and then clicking 'Message Source' gives you the full raw email.

Google keeps a [page](#) [wayback machine](#) with information on viewing headers in a few more mail clients.

Unfortunately, mobile mail clients typically don't allow you to view the headers. This certainly limits the ability to remotely support someone who only has access to a mobile phone.

**Exercise 9.3.** In all the email accounts you have access to, open an email and view the email headers and/or body. Look for the types of headers mentioned above.

*Tip:* if you regularly support people with certain types of email accounts, make sure you have an account on that service (assuming it is free) and that you are a bit comfortable with its interface. For example, you may not use Google, but having a Gmail account just to learn how it works is likely crucial, given how popular it is.

## Two kinds of From addresses

As mentioned before, the "MAIL FROM" email address isn't always the same as the email address displayed in the mail client, which is the "From:" address from the headers. You can think of MAIL FROM as the name displayed on the back or left side of an envelope of a physical letter (it is sometimes called the 'envelope From') and the From: address as the name at the bottom of the letter inside. (Neither "MAIL FROM" nor "From:" are official ways to refer to the two kinds of addresses. Other resources may use different names.)

In many cases, MAIL FROM and From: match, but this isn't always the case. For example, if your organization uses a third-party company like Mailchimp or SendGrid to send its weekly newsletter, the MAIL FROM address will likely be a long address belonging to that company and often unique for each recipient, but the From: address will be one belonging to your own organization. This way replies to the email will be received by your organization, but in case the email is undeliverable, bounces will be received by the company handling the newsletter.



From a forensics perspective, a difference between the two senders isn't necessarily a red flag that requires further investigation. However, in case the two values differ, it is worth looking into the MAIL FROM to see who the domain belongs to. Sometimes malicious emails are sent from compromised mail servers with a real MAIL FROM address (for example, to ensure SPF passes; more on that later) but a forged From: address. The recipient of the email would only see the latter and thus be tricked.

**Exercise 9.4.** Look among your recent emails for a legitimate email where From: and MAIL FROM (i.e., the Return-Path header) are different (hint: look for newsletters).

## The sending IP address

Sending IP addresses can help you determine whether an email was forged. It is important to know that a mail server (and thus a spam filter running on it) cannot be certain about pretty much any part of the email. However, one thing a mail server can be sure about is the IP address of the sender, which is the one making the SMTP connection. This is why senders' IP addresses play a vital role in spam filtering. Mail services lookup sending IP addresses on a blacklist. If the IP address is on the list, the email service may block the message or combine it with a scan of the body to determine whether the message is spam.

As you typically don't have access to the SMTP logs, the easiest way to find this IP address is in the Received headers. Go through them from the top and find the first received header from an external server not in your organization. That will include the IP address. It will likely be an IPv4 address; occasionally, IPv6 addresses may occur. Most blocklists focus on IPv4, so if you find an IPv4 address, it will likely be easier to find more information.

A very helpful site is <https://multirbl.valli.org/>, which allows you to look up an IP address in a large number of blocklists. You don't need to know the details of most of these lists, but this site can give you a quick idea of whether an IP address is a known spammer: the more block results, the more likely it is.

Several things are worth noting in understanding blocklists. The first is that blocklists get updated very quickly. If a server is sending spam or otherwise malicious emails because it is compromised, its IP address is often blocklisted but also quickly removed once the issue has been solved. If you're looking up an IP address long after the email was sent, its results won't be too accurate. If you do the lookup very soon after the email was sent, it may be worth checking again after a few hours, though for larger spam campaigns, blocklists often add the IP addresses within minutes.

Secondly, most blocklists are updated automatically based on spam seen by various sensors. Consequently, if the sender sends a very small number of malicious emails, the campaign may stay 'under the radar,' and the IP address might not end up on blocklists.

Relatedly, if the emails are sent from accounts at large providers, such as Google or Apple, the IP addresses are so widely used for sending legitimate emails that they almost certainly won't end up on blocklists. These providers are considered 'too big to block.'

Finally, the blocklists focus on emails sent over the public Internet from one mail server to another. By policy, many ISPs don't allow their home customers to run their own mail servers<sup>41</sup>. Some blocklists, such as

---

<sup>41</sup> This policy goes back to the first decade of the century, when Windows PCs often got infected with spam-sending malware. These days, if someone wants to send email from a home account, they either use webmail, or their mail client uses a variant of SMTP that submits the email to a mail server they have an account on.





Spamhaus PBL<sup>42</sup>, list such IP addresses proactively, regardless of whether there is a history of sending spam. Other lists, such as lists by SORBS or Abusix, proactively list dynamic IP addresses, which are also not supposed to send emails. If you look up your home IP address on these lists, you might find it listed for these reasons, not because of any past malicious activity performed from the IP address.

**Exercise 9.5.** Look for a recent legitimate email you received, and check the sending IP address against the blocklists. Make sure you find the correct Received line.

**Exercise 9.6.** Now, do the same for a recent spam email you received.

## The email body

The **body** of the email is the main part of the email: everything after the headers. This includes the text of the email that your mail client displays, as well as any attachments or embedded images. The body of an email is in plain text by default. You may also see a Content-Type or Content-Transfer-Encoding email header, which tells your client that the body of the email is formatted in HTML or has attachments or embedded images and should be displayed differently.

When performing email forensics, you often don't need to look at the body. In fact, as you saw earlier, obtaining the raw body isn't easy in some mail clients. Still, a basic understanding of what an email body looks like can be helpful.

If the body of the email's Content-Type is multipart (whether it is multipart/mixed, multipart/alternative or something else), it will also define a string that serves as a boundary between the parts. The body is then split into multiple parts, each of which is separated by the boundary on a line on its own. The final body part is followed by the boundary on a single line, followed by two hyphens on the same line.

Each email part (more formally, a MIME part) looks a bit like an email on its own, with a small header and a body. The Content-Type and Content-Transfer-Encodings are probably present for each part. It is possible for a part to itself contain multiple parts!

The text of most emails these days is written in HTML, the same markup language web pages are written in. However, the HTML used in email is more limited in functionality.

It is common for emails to contain both HTML and plain text in different body parts. The plain-text part of the message will be displayed by mail clients that only handle text.

## Base64 and quoted-printable

Two common email encodings for pieces of text that include non-ASCII characters are quoted-printable and base64.

In **quoted-printable**, only non-ASCII characters are encoded. This means that if a text contains mostly ASCII characters, you can probably still read it (as in Exercise 9.7 below). You don't need to understand how it works, but if you're interested, you can check the [Wikipedia](#) page.

---

<sup>42</sup> And also Spamhaus ZEN, which combines all the lists of Spamhaus, probably the best-known blocklist provider.





In **base64** encoding, the whole text is encoded using 65 printable characters (the alphanumeric ones, together with +, / and =). As a consequence, you won't be able to read any base64-encoded text until you decode it (which, thankfully, is very easy). Again, you don't need to understand how base64 works, but [Wikipedia](#) is a good resource if you want to learn more. (It will also explain why there are 65, rather than 64, characters used.)

To decode some quoted-printable text, you can use one of many online decoders (again, the [linked site](#) is one example). You can also save the text in a file, say `qp.txt`, and then run the following command as a single line in a terminal window in REMnux (or any other Linux system) to save the decoded text to a file `out.txt`:

```
cat qp.txt | perl -MMIME::QuotedPrint -0777 -nle 'print decode_qp($_)' > out.txt
```

You can then open `out.txt` in a text viewer of your choice. If you use `out > out.txt` command, the output will be displayed in the command line, but this may not render all non-ASCII characters well.

In case you're curious what that command does (remember, copying and pasting commands you don't understand is something you should avoid as much as possible!), it first outputs the content of the `qp.txt` file, then uses this output as an input for a tiny Perl program that uses the `MIME::QuotedPrint` module to decode the text and then saves its output in the file `out.txt`. Refer to Chapter 4 for an understanding of what the vertical line `|` and the greater than sign `>` do.

For base64 decoding, there are also many [online decoders](#) (once again, this is one example). You can also use the built-in Linux command `base64`. If you have base64-encoded text stored in a file `b64.txt`, the following command saves the decoded text to a file `out.txt`:

```
cat /tmp/b64.txt | base64 -d > /tmp/out.txt
```

**Question 9.7.** Use the command line to decode the following text:

```
Libert=C3=A9  
=C3=89galit=C3=A9  
Fraternit=C3=A9
```

(See the appendix for the answer.)

**Question 9.8.** Use the command line to decode the following text:

```
VGhpcyB0ZXh0IGNvbnRhaW5zIG9ubHkgQVNDUkY2hhcmFjdGVycy4gVGhlcmUgd2FzIG5vIHJl  
YXNvbiB0byBlc2UgYmFzZTY0IHRvIGVuY29kZSBpdC4gSWYgeW91IHNLZSB0aGlzIGluIHByYWN0  
aWNlLlCBpdCBpcyBhbHdheXMgc3VzcGljaW91czogc29tZXRpbWVzIHRleHQgaXMgYmFzZTY0IGVu  
Y29kZWQgdG8gZXZhZGUgc3BhbSBmaWx0ZXJzIG9yIHNL1Y3VyaXR5IHByb2R1Y3RzLiBIb3dlbmVz  
LCBiZW5hdXNlIGJhc2U2NCBlbmNvZGluZyBpc24ndCB1bmNyeXB0aW9uLlCBpdCBpc24ndCB0eYXJk  
IGZvciBzdWN0eHNjYW5zIHRvIGxvbnRzIG9ubHkgQVNDUkY2hhcmFjdGVycy4gVGhlcmlhIA==
```

(See the appendix for the answer.)

**Question 9.9.** For potentially sensitive text, why would you prefer to use the command line instead of a web interface? (See the appendix for the answer.)



## Using emldump.py to view MIME parts

If you have access to a raw email (so the headers and the body together), you may want to extract certain parts of it, for example, attachments, for further analysis. Given that attachments are often base64-encoded, you can open the file in a text editor, copy the encoded attachment to a new file and then use the above method to decode it, store the output to a file and analyze it. There, however, is a much simpler way in REMnux, using a tool called [emldump.py](#).

To use this Python tool, assume you have the raw email saved in a file `email.eml`<sup>43</sup>. Then run:

```
emldump.py email.eml
```

This will show you the various parts of the email, their size, their MIME type and, if available, their name.

An example output will be:

```
1: M          multipart/mixed
2:   79126 text/html
3:   99985 application/pdf (firstattachment.pdf)
4:   99822 application/pdf (secondattachment.pdf)
```

Now if you want to see the first attachment, which is number 3 in the list, and store it in a file `file.pdf`, you run:

```
emldump.py email.eml -s 3 -d > file.pdf
```

And that's it: the `-s 3` means the third part will be selected and `-d` means the part is 'dumped.'

Of course, to save an attachment, you can also use the email client. But this means a potentially malicious file is (temporarily) stored on the device and may be blocked by antivirus!

**Exercise 9.10.** In an email client that allows you to save raw emails, such as Gmail, find a non-malicious (!) email with at least one PDF attachment. Save the attachment and move it to your REMnux installation. First, view the email in a text editor. Do you recognize the MIME boundary defined in the header and the various MIME parts in the body? Do you see the encoding, as well as the header, in a MIME part that gives the part a name?

Then, use `emldump.py` to extract the PDF attachment. Open the PDF in Firefox on REMnux to confirm that extracting worked!

Don't forget to delete the email file and the PDF file afterward to practice this good habit.

## Embedded and external images

Many emails these days contain images. Sometimes, these images are embedded in the email itself: they are a MIME part, and if you study the HTML source of the main body, you'll find this MIME part being referenced in a `<img>` tag.

---

<sup>43</sup> You can use any file extension, including no extension at all, but `.eml` is commonly used for raw emails



More commonly, images are downloaded from an external website. This makes the emails smaller and has the advantage for the sender that they can trace when an email is opened and from which IP address: they see when the images are being downloaded. This may be a privacy concern, especially for high-risk individuals.

For such people, it is recommended to set the email client not to download images by default. [This website](#) [wayback machine](#) is one of many that explains how to do this in various email clients.

As an analyst, it is important to keep this in mind when analyzing a potentially malicious email that refers to external images. If you query the URL in some way, this may be viewed by the sender as opening the email. Depending on the context, this may or may not be a concern.

## Forwarding emails for analysis

It is common that you will want to analyze an email received by someone who is at a different location and thus would like the email to be forwarded to you.

Forwarding is common in emails but isn't natively built into the email protocol. When someone forwards an email in a mail client, a new message is created within the body of the original message, including the attachments, together with some basic information from the original email to indicate that it was forwarded. The original headers won't be present, and forwarding emails thus isn't very helpful if you want to perform forensics on them.

Thankfully, several mail clients allow one to forward an email as an attachment, which is a simple way to have someone send you the full email.

In Gmail, one should select the email in the folder view (for example, the inbox or the spam folder), then click on the three vertical dots that appear above the list of emails and select 'Forward as attachment.' In Outlook, in the menu view, it is an option under the Home menu near the end. In Apple Mail, it is possible to open an email and then select 'Email,' 'Forward as' and 'Attachment' in the menu.

If that does not work, but the user can view the raw email, it is recommended they do not save the email to the disk, as an antivirus product might block that and disabling antivirus is a bad idea. Instead, the user should copy the full email and paste it into a service like [Cryptpad](#)<sup>44</sup> and then share it with you.

If that doesn't work, for example, because they only have a mobile device, forwarding emails the normal way can still be helpful to analyze the body for malicious links and/or attachments, but you might not be able to detect spoofing.

If you regularly support people who require your assistance analyzing links and/or attachments, it is best to set up a dedicated email address for them to forward emails to (any free webmail will do) rather than use your regular email account.

A dedicated email address means it won't be a big issue should the account be blocked, which could happen if you handle a lot of malicious emails: you can just create a new one. It also means you can access the account from an analysis machine, such as REMnux, without having to worry about the security of the account. Be sure to delete emails that turn out to have been legitimate, as they may contain sensitive information.

---

<sup>44</sup> Or another end-to-end encrypted service where one can paste text



## Authentication protocols: SPF, DKIM and DMARC

As mentioned before, there is nothing built into email itself that prevents spoofing or forgery. To somewhat fix that shortcoming, the protocols SPF and DKIM were developed, later to be joined by DMARC.

### SPF

In **SPF**, a domain owner uses DNS to state which IP addresses are allowed to send email with this domain as the MAIL FROM address (SPF does not say anything about the From: address!). You don't need to understand how SPF records work (but [this wayback machine](#) is a good place to start if you're interested), but for each IP address, there are four options:

- Pass: this IP address is allowed to send emails from this domain
- None: no statement is made about this IP address being allowed to send emails
- Softfail: this IP address probably isn't allowed to send emails
- Fail: this IP address isn't allowed to send emails

Because email is complicated it is surprisingly hard to be certain which IP addresses will send email for a domain, so in practice most organizations don't set any IP address to fail SPF. Instead, they state which IP addresses are those of their mail servers and then apply 'softfail' or sometimes 'none' to anything else.

Most email providers add the SPF status in the headers of emails they receive. You can also check the SPF status of a domain and an IP address in various online tools such as [this one](#). Note that SPF records may change over time, and results may be less accurate for older emails!

### DKIM

**DKIM** is a protocol that cryptographically links an email to a domain name and a keyword called 'selector.'

That might seem confusing. To understand the context, remember that email doesn't natively come with authentication or integrity. If an email seems to come from facebook.com, there is no way to be sure it really did, or if it did, it wasn't modified along the way.

What DKIM does is add a digital signature to the body and some of the headers – including the most important ones like Subject, From and To – that link it to the domain. A DKIM signature by facebook.com proves that the email was handled by a server belonging to Facebook and that the body and the relevant headers haven't been modified since.

The selector is a keyword that allows an organization to have different signing keys for different functions. For example, it could use one key to sign PR emails and another one to sign business emails. The selector is rarely relevant for you.

DKIM helps ensure that legitimate emails are being delivered correctly, as now a receiving mail server will know that the email came from Facebook, which it likely considers a reputable sender.

A DKIM signature is added to the header of an email using the DKIM-Signature header. Here is an example of such a header:



```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
d=internews.onmicrosoft.com; s=selector2-internews-onmicrosoft-com;
h=From:Date:Subject:Message-ID:Content-Type:MIME-Version:X-MS-Exchange-
SenderADCheck; bh=eImdSwruOz8MASinrUKJ8cYwpQfdjnKuri+wbkULvI4=;
b=NMCGIhuMHiXiQqYmL6OfT6lTM/8czHgFcjbzgh7ifnVvKVjefS82PBjAtJdt/u0m4SDKgkz
IQ6HHp8ObU+dJprmc5iOchXjIAAiJpmz+n9eXcfi8vCQTC8zk5htQ498CUXFIaquE3YvOpq5m
xktdcHzUPGKJDj13aCoxh8RQfiA=
```

You'll recognize many different parts. The most relevant ones are `d=`, which is the signing domain; `s=`, which is the selector; `h=`, which contains the signed headers; and `b=` and `bh=`, which contain the signature. Note that it is possible for an email to have multiple DKIM signatures; multiple servers could have added one.

There are several important things to note about DKIM. The first is that digital signatures are very sensitive, and DKIM easily breaks if a mail server makes a tiny modification to an email, for example, adding an extra line to the body. The DKIM specs, therefore, prescribe that a broken signature (one that 'fails' in DKIM language) be considered the same as the absence of a signature.

If you're trying to determine if an email is legitimate and find that it fails DKIM, don't pay too much attention to that. If the email was forged, there are almost always other properties, such as the sending IP address, that don't match what is expected from this sender. On the other hand, if the DKIM signature is valid, you can be certain<sup>45</sup> that it was sent by a server belonging to that organization.

Secondly, DKIM keys are rotated regularly and trying to verify the signature of very old emails might lead to a failure, even if the email would have passed DKIM originally.

Thirdly, DKIM is sometimes used to verify the authenticity of emails in a data leak. While this is indeed a possibility, the protocol's authors warn that it was not designed for this purpose.

Finally, the mere presence of a DKIM-Signature header does not mean that the signature is valid! Verifying the signature manually is very fiddly and generally recommended against; remember that accidentally adding an extra line to the body makes a valid signature fail. Thankfully, most mail servers that verify DKIM add the results in an Authentication-Results header, which also contains information about SPF and DMARC. Some mail clients, such as Gmail, when choosing 'Show original,' also show you the authentication results directly:

SPF:	PASS with IP 40.107.92.127 <a href="#">Learn more</a>
DKIM:	'PASS' with domain internews.onmicrosoft.com <a href="#">Learn more</a>
DMARC:	'PASS' <a href="#">Learn more</a>

**Question 9.11.** The above DKIM signature is for an email sent by Internews. Internews uses Microsoft 365 for its email. Can you explain why the signing domain here is one belonging to Microsoft (which owns onmicrosoft.com) and not Internews? (See the appendix for the answer.)

<sup>45</sup> DKIM uses public key encryption, with the public key being published through DNS and the private key kept secret on the signing server. In theory, it is possible for the signing key to get stolen and used to sign forced emails. In practice, this is extremely rare and not something to consider.



**Question 9.12.** The DKIM protocol doesn't have an option to just sign all the headers. Why is that? (See the appendix for the answer.)

**Exercise 9.13.** In a mail client of your choice, look for an email sent from another provider. Is it DKIM-signed? Can you find the results in the Authentication-Results header?

## DMARC

**DMARC** was a later addition that solved a problem email providers were dealing with: how to handle an email that claims to come from, say, `facebook[.]com` that doesn't have a (valid) DKIM signature, or that fails SPF? Should they just discard it as fake or assume that it may still be legitimate? DMARC lets Facebook, in this case, set guidance on what to do with such emails.

DMARC has a mail server look up the DMARC record (think of this as the DMARC settings) of the domain in the From: address. Remember that this is sometimes different from the MAIL FROM address, which SPF uses. As with SPF and DKIM, this lookup is performed against specific DNS records. If the email passes either SPF or DKIM, DMARC passes; otherwise, DMARC fails.

The DMARC record then recommends what the mail server does with emails that fail DMARC<sup>46</sup> and whether (and how) to send feedback to the domain in the From: address. For organizations that send a lot of emails and that are often targets of phishing campaigns, such as PayPal or Facebook, this feedback gives them insight into such campaigns and also helps them quickly debug possible problems with their legitimate emails.

In performing email forensics, however, DMARC is a lot less relevant than SPF or DKIM.

One nice thing about all three protocols is that they create a catch-22 situation for spammers and other malicious actors: either they use them, making their emails more clearly linked to them. Or they don't, in which case the emails stand out in a world where not using SPF, DKIM and DMARC is increasingly rare.

This is why there has been a trend in recent years for malicious emails to be sent from compromised email accounts or servers or from accounts on 'too big to block' email providers such as Gmail and Yahoo.

**Question 9.14.** In the DMARC example above, the email was sent from `facebook[.]com`. In practice, at least at the time of writing this guide, Facebook uses the domain `facebookmail[.]com` for sending emails, like account security notifications.

There is no formal way to link `facebookmail[.]com` to Facebook. What steps would you take to decide whether such a domain is linked to a particular organization, like Facebook, in this case? If you have access to such emails from Facebook, feel free to look at them. (Note that there is no single right answer to this question, but see the appendix for some suggested answers.)

---

<sup>46</sup> Of course, if the mail server considers the content of the email spam or malicious, it will still block it regardless of the DMARC policy. But if it doesn't notice anything bad but the DMARC policy says to block it, it will likely do so.



## Chapter 10: Analyzing email payloads

Chapter 9 focuses on analyzing the email itself, while this chapter will teach you some basics of analyzing malicious attachments and links (also called “payloads”) in emails. The first part covers sandboxes, a convenient way to analyze potentially malicious files and links, and the second part is an introduction to analyzing the files manually – though, as the chapter will explain, this is something you will want to avoid if you can.

Because this chapter deals with malware, make sure you have thoroughly reviewed Chapter 4.

### Sandboxes

A **sandbox** is a program that provides a safe environment for a computer program to run in. One particular kind of sandbox is a **malware sandbox**, in which malware can be executed in a contained way, and its actions can be analyzed.

The sandbox this chapter recommends will let you upload a file to run or connect to a URL to load in a browser. After some time for analysis, the sandbox will provide you with a report of the activities on the machine, often highlighting what, if any, of the activities are indicative of malicious behavior.

Because malware often uses multiple stages, obfuscation, and other anti-analysis techniques, using a well-run sandbox is often a lot faster than performing a manual analysis. It also makes it easier to get a high-level understanding of how malware works.

You could, in theory, set up your own sandbox environment by using virtual machines: you create a virtual machine (or VM), typically running a recent version of Windows, and take a snapshot, then run the malware on that VM. After some time, you look at what happened and then revert to the snapshot.

However, there are many downsides to this approach:

- While a VM would help you avoid any damage done by changes to the operating system, you still run it on your own network. Malware may find ways to spread from one machine to another. It may also connect to the Internet, and if these connections are detected, your IP address may end up on a blocklist. Some services won't let you access them from a blocklisted IP address.
- You'll have to manually check all the changes made to the VM and decide whether they indicate malicious behavior. You'll also have to somehow capture the network traffic and see what connections are being made.
- Anti-analysis techniques often look for virtual environments, in which case the malware won't run if it detects one. Some examples of what malware might be looking for in a VM are CPUs that are typical of virtual machines, a background image that is the default Windows one, an unreasonably small hard drive, or very few files present on the hard drive aside from the ones present after installation.
- Some malware makes a connection to its control server, which checks if it has already run from this particular IP address. If it has, it will not run again.

Some of these issues can be relatively easily addressed. Others are more complicated – and the list above is certainly not exhaustive.





Thankfully, sandboxes exist that are already set up to mitigate these challenges. A very popular one is [Cuckoo](#), which is open-source. If you want to experiment with running a sandbox, Cuckoo is a good place to start. It was originally written by people with connections to the internet freedom community, so it's designed with civil society in mind.

However, it is good to know the original Cuckoo, the last version of which was released in 2018, isn't under active development<sup>47</sup> and thus is less likely to be able to handle current threats properly. A group of people at CERT-EE, the Estonian national CERT, is rewriting Cuckoo for Python 3; you can follow their progress on [GitHub](#), and if you feel adventurous, install this version of Cuckoo, but expect having to do quite a bit of troubleshooting to get it to work.

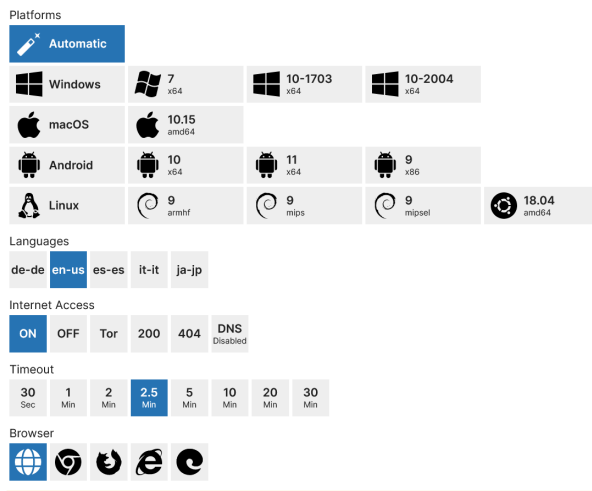
A much easier option is to use cloud-based sandboxes. Several ones are available, such as [ANY.RUN](#), [Hybrid Analysis](#), [Joe Sandbox](#), [Triage](#) and even an online version of [Cuckoo](#). All of them have free versions that allow you to upload malware and URLs, though some do require registration<sup>48</sup>. However, a warning: as with VirusTotal, using one of these sandboxes essentially makes the analysis publicly available. If it seems likely the malware has been targeted specifically to you, publicly signaling that you are analyzing it could let the attacker know you are investigating them.

Each sandbox has its own unique characteristics, but they also share many features. What follows is a description of Triage, but you are encouraged to try out other sandboxes too and, if there is one you particularly like, become familiar with how it works.

The following screenshots are taken from a malicious document [found](#) (sha256: 8404d3dc32b0555bc3b076d7fc080d2a341508b4a2c84805a1d5ffc0057e2b39) on Malware Bazaar, where you can download it for free if you want. The sandbox analysis is [available](#) in the public version of Triage.

In Triage, use 'Submit' in the top left corner of the page to upload a file. Very helpfully, it allows you to upload password-protected files (the default password it tries is 'infected' – remember this is the industry standard to use!), so you don't have to worry about keeping a potentially malicious file on your computer before submitting it.

You can also submit URLs, either to download malware directly from a URL and then analyze it or to analyze a URL in a browser.



<sup>47</sup> One issue is that Cuckoo was originally written for Python 2, a deprecated version of Python

<sup>48</sup> At time of writing, Hybrid Analysis and Cuckoo can be used without registration, Any.run and Triage require free registration and Joe Sandbox requires registration and account approval



On the next page, you can choose various settings for the sandbox. The most important is the operating system. As you can see, you can choose from various versions of Windows, macOS, Android or Linux. If you don't choose anything here, you can have Triage choose one or more operating systems for you, and that usually works.

You can also choose the language settings of the machine. The default is en-us: US English. Sometimes, malware that targets a particular region or group of people only runs on machines with a particular language. Then, you can choose whether to keep Internet access on (the default) or off or to connect via Tor. A lot of malware makes a connection to an adversary-controlled server, and if this is a possible concern for you – maybe you don't want to tip off the adversary that you're analyzing it – you can try to run the sandbox with the Internet connection disabled or set it so that web requests emulate 200 or 404 return codes. Some malware makes a quick check for the presence of some websites to see if it is run in an environment without an Internet connection; this would fake such a connection. Some malware may still not run, however.

You can also set a timeout: the execution will stop after no activity for some time (2.5 minutes, by default). A common anti-analysis technique is for malware to wait for some time before performing any malicious activity, so in some cases, you may want to increase the timeout.

Finally, you can change the browser the sandbox uses to open websites. This can very occasionally be useful in case the potential threat you're analyzing only runs in a particular browser.

You can now start analyzing the file (or URL), which opens it in the sandbox.

One nice thing about Triage is that you can both see what is happening on the sandbox's virtual operating system – or operating systems: it can run multiple analyses in parallel! – and also interact with that operating system. Sometimes, this might be helpful, for example, if malware watches your system and records when a button is clicked. In most cases, it suffices to have the sandbox run in the background, though.

When the sandbox has completed, you can view the analysis. This is what you're most interested in.

The screenshot displays the 'General' tab of a file analysis interface. On the left, a list of file properties is shown, each with a copy icon:

- Target:** 8404d3dc32b0555bc3b076d7fc080d2a341508b4a2c84805a1d5ffc0057e2b39.zip
- Size:** 11KB
- Sample:** 230224-rw6daahqdg
- MD5:** 49f24a25546445332ba596d8e8480f5c
- SHA1:** e1983637e782bf7a6fc4a51e91ab63d7f56fa90a
- SHA256:** d17fa9cb0fd40bfc3f70ab4f8d738757069d2813bd91098bd2a862c12f833e58
- SHA512:** 93cb4f855a7bcb13124fcb54eff1b1fd9814fa13dc7f32a9f6e17643bed9b79e4a7eb91c603b8bc40620592b...
- SSDEEP:** 192:jjzOoCxiAk0lr2OPgRiDvAfmuv8c5gWdQmOdVk+qMikrm9vOyY+KH0++YFPAjzOoCuAkLr2O/7Z08c+Nd...

On the right, a 'Score' section shows a score of 10/10. Below this, a horizontal bar chart displays tags for the malware: 'agenttesla' (highlighted in red), 'collection', 'keylogger', 'spyware', 'stealer', and 'trojan'.

The 'Overview' page is where you will most likely want to look first. This gives you a summary of the sandbox analysis. In this case, we see that Triage gives our sample a score of 10 out of 10 and tags it with the name 'agenttesla'. Agent Tesla is common malware used to steal information, such as credentials, from an infected machine. You'll also note tags such as 'stealer', 'keylogger' and 'spyware' that make it even clearer that this is bad.



In most cases, this will be good enough: you know that the file you wanted to analyze is Agent Tesla. There are enough analyses of Agent Tesla available on the Internet to give you an idea of what it does. Because Agent Tesla is fairly common, Triage also knows how to extract configuration information from the malware and displays that in the 'Malware Config' section.

**Question 10.1.** Based on the configuration extracted from the malware, how does this Agent Tesla variant extract information from an infected machine? (See the appendix for the answer.)

The 'Targets' section again shows you that this is Agent Tesla, but it also shows its malicious activities. For example, a blocklisted process makes a network request, and an 'MZ/PE' file<sup>49</sup> is downloaded. Remember that the file you are analyzing is an Office document. It is highly suspicious that it is trying to download files!

This section also shows other things the malware tries to do, such as accessing FTP configuration files and data from email clients and web browsers. That activity is common for information-stealing malware. Even if you were trying to analyze malware the sandbox did not recognize, such activity is highly suspicious.

There is also a section for the 'MITRE ATT&CK Matrix'. ATT&CK is a "framework" that threat analysts use as a common, easier-to-understand language for describing and documenting tactics and techniques used by adversaries. In this case, this tab shows you techniques (such as collecting emails or modifying the registry) used by this malware. ATT&CK was designed with enterprises in mind and focuses on threats and not on individual malware samples, but it won't hurt to have a look at the techniques recorded in this section.

The 'Replay Monitor' section lets you replay what the malware did to the operating system, which is really helpful to understand what happened visually. Finally, if the malware tried to download any new files from the Internet, you can view them in the 'Download' section and possibly download them yourself for additional analysis. Beware that these files could be malicious, so handle them with care!

Aside from the 'Overview' page, you will also want to have a look at the tabs in the sandbox corresponding to the file's behavior. In this case, the file was run on both Windows 7 and Windows 10.

Let's look at Windows 7 first. You'll find the most interesting information in the 'Report' tab. Some of this was also available on the 'Overview' page, but you'll notice some extra information. For example, in the 'Signatures' section, it says that the file launches Equation Editor and that 'Equation Editor is an old Office component often targeted by exploits such as CVE-2017-11882'. Interestingly, this vulnerability is commonly exploited to target civil society [wayback machine](#).

In the 'Processes' section, you can see what processes the malware started after the file was opened. You can see four processes were started: WINWORD.EXE, splwow64.exe, EQNEDT32.EXE, and arnolded4874.exe – with the latter started twice. You're not expected to know what this all means, but these processes can be useful when trying to understand what is happening.

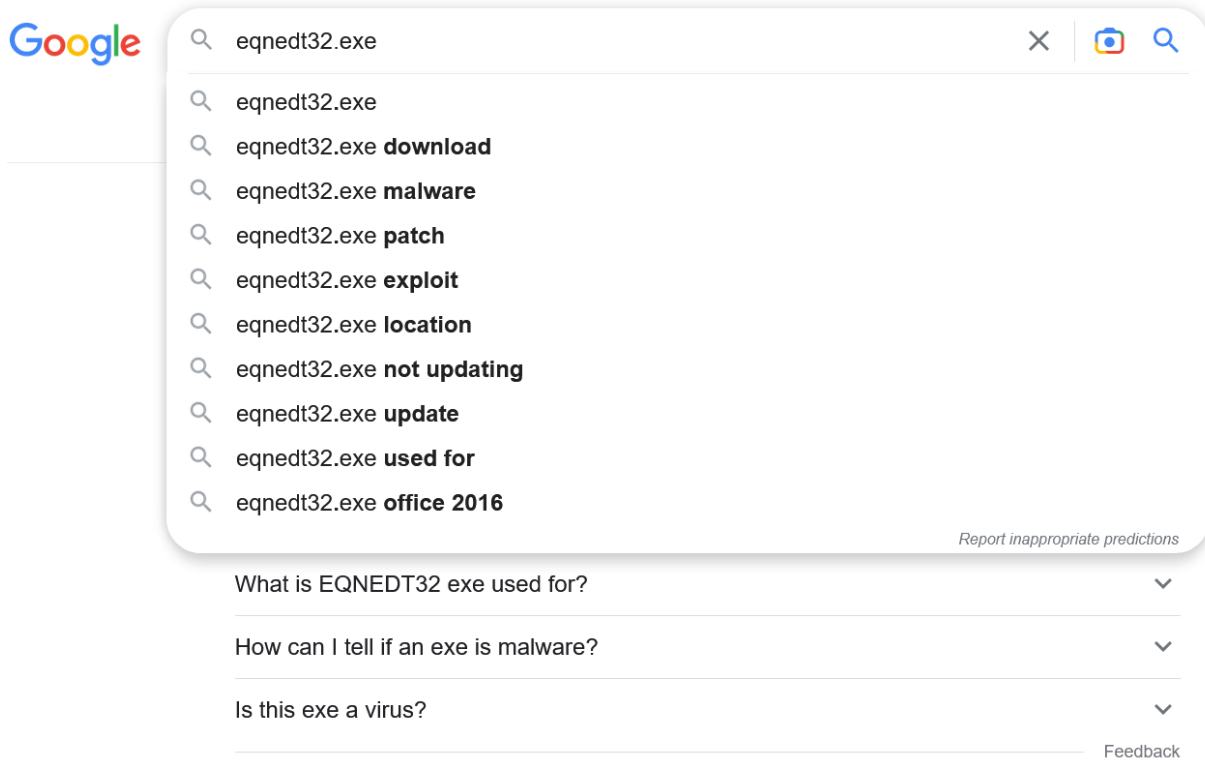
A search engine will tell you that WINWORD.EXE is Microsoft Word, which makes sense as this is a Word file: Word would run when you open the malware file. The second process, splwow64.exe, is also a legitimate program and has to do with making 32-bit programs run on a 64-bit operating system.

---

<sup>49</sup> PE stands for 'portable executable', while MZ are the first two bytes of a PE file. Both .exe Windows executables and .dll Windows libraries are examples of such files.



EQNEDT32.EXE is also legitimate, but looking at the search engine suggestions for this file, you'll see 'malware' and 'exploit' mentioned frequently, suggesting it is commonly used to execute malware. Indeed, this is the previously mentioned Equation Editor.

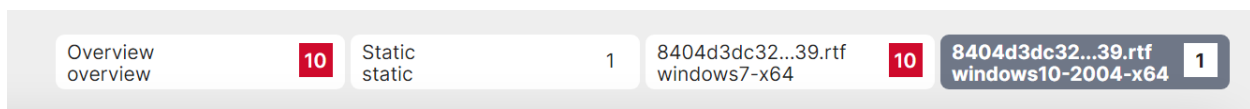


Finally, there are very few results for arnolded4874.exe in Google: at the time of writing, only one, and that is our own sample in Malware Bazaar. A file starting a uniquely named process is extremely suspicious!

Indeed, in the next section, 'Network,' we notice an HTTP request that was made to a URL containing this very file. So, the malware downloaded this file from the Internet and then ran it on your virtual system. Even ignoring all the previous red flags, this should be a very big one for an Office document.

If looking at these tabs didn't give you enough information, you may want to peek at the other tabs where you can see much more activity, such as all the network activity, which files were accessed, and which registry keys were read and set. Registry keys can relate to global settings on Windows. There's a lot of information here, and it takes experience to understand all the details, but you may find some clues here to understand any possibly malicious behavior better. Often, the 'Report' tab gives you all the information you need, though.

Remember, the sample was also run on Windows 10. Interestingly, nothing malicious happens in that case – note the score of 1, as opposed to a score of 10 (out of 10) for Windows 7.



There are various possible reasons for this. The most plausible reason in this case is that the exploit didn't work: CVE-2017-11882 was fixed late in 2017. Windows 10 is from 2020, so it includes the patch that fixed that vulnerability.



Other reasons why a malicious file may not show any malicious activity could be that the attacker is stopping the sandbox from running it by using anti-analysis techniques. Or an online resource that the malware tries to connect to is no longer available. Remember, just because a file is malicious doesn't mean that running it in a sandbox will always show malicious activity!

**Exercise 10.2.** Download the sample

[37419d3a8a50d2e5bc0eef676a37d6757ba43a64eff868edb4af5c386900235f](https://www.malwarebazaar.com/sample/37419d3a8a50d2e5bc0eef676a37d6757ba43a64eff868edb4af5c386900235f) from Malware Bazaar and run it inside Triage. Share as many indicators of malicious behavior as you can find in the 'Reports' tab.

**Exercise 10.3.** Download the sample

[a43e0864905fe7afd6d8dbf26bd27d898a2effd386e81cfbc08cae9cf94ed968](https://www.malwarebazaar.com/sample/a43e0864905fe7afd6d8dbf26bd27d898a2effd386e81cfbc08cae9cf94ed968) from Malware Bazaar and run it inside Triage. Look for malicious indicators. If the analysis doesn't give you a score of 10 out of 10, run it again, but this time, interact with the machine and click 'Next' in the warning as OneNote asks you to do. Does this change the behavior detected by the sandbox and the score?

**Exercise 10.4. (optional)** If you regularly use another online sandbox, run the samples from the previous two exercises in that sandbox, too. What indicators of malicious behavior do you find?

**Exercise 10.5. (optional)** Look for some recent malware and run it in a sandbox of your choice to become familiar with how the sandbox works. [Malware Bazaar](https://www.malwarebazaar.com/) is a great place to find a lot of new malware, but it is even better if you find an article that analyzes malware and upload that: then you can compare the sandbox output with the analysis in the article to understand your sandbox's results.

## Manually analyzing attachments

In the previous section, you learned how useful sandboxes are when it comes to analyzing email attachments or links found in emails. In most cases, a sandbox will give you all the information you need. But sometimes, you want to understand a potentially malicious attachment better because it helps you do your work or simply because you are curious. This section helps you understand how to perform manual analysis to explore further.

Email has been a common vector to spread malware since the late 1990s (when malware was still referred to as viruses<sup>50</sup>). Simply opening an email attachment would infect your computer, after which the malware would do bad things and use your email account to send a copy of itself to all your contacts.

For malware authors, things are a lot harder these days. Effective spam filters make it much harder to deliver an executable file<sup>51</sup> of any kind, including malware,<sup>52</sup> to a target. Computers also have better protections, such as built-in antivirus, against malicious files downloaded from the internet or included in archives like .zip files.

---

<sup>50</sup> Early computer viruses behaved similarly to biological viruses, in that they infected legitimate files and were run once the 'host' file was run. Such viruses are extremely rare these days, but the term 'virus' is still often used to refer to malware itself.

<sup>51</sup> A file that runs on the computer. This would include legitimate programs but also a lot of malware. On Windows, executables often have the .exe extension.

<sup>52</sup> In some cases, malware does not even end up in a spam folder. Spam filters discard a lot of emails without notifying the user, especially malicious ones. They may also automatically remove malicious attachments.



Thus, malware actors have been trying to find ways around this. Almost all of these involve convincing the recipient to act, whether by clicking a link, enabling macros, or otherwise bypassing security rules that prevent automatic infection with malware.

This turns malware delivery into a game of cat-and-mouse. Security vendors keep improving their detection for new file types, and software vendors such as Microsoft keep making it harder for their software to be abused by malware. But malware actors keep finding new ways to deliver malware.

As a consequence, your goal as someone who responds to security incidents – and the goal of this training module – shouldn't be to know how to analyze every possible kind of payload but rather to understand the basics and know where to look if you find a new type of payload.

Attachments and links are two different ways attackers can add a payload (which may be malicious) to an email. However, it is common for an attachment to contain nothing but a link to the actual payload or for a link to download a malicious file that is the real payload. Sometimes, an attachment has a link that downloads yet another file.

## Malicious Office documents

As mentioned previously, Microsoft Office files such as Word documents, Excel spreadsheets and PowerPoint presentations can contain one or more macros: bits of code that are automated. There are legitimate reasons why organizations use macros in such documents, but macros have long been popular among malware authors.

Around the turn of the century, macros would run automatically once a file was opened. This led to massive email worms: emails with attachments that, when the attachment was opened, would run a macro that emailed a copy of the attachment to every person in the address book. Understandably, Microsoft disabled the automatic running of macros. For about a decade, macro malware appeared to be a thing of the past.

However, around 2014, malware authors switched to a new tactic that used social engineering to get a user to enable macros: the file would, for example, show a blurred page, and macros were supposedly needed to be enabled to show the content, often saying something like 'for security reasons.' This became a major infection vector for many different kinds of malware.

Recently, Microsoft has made some changes that make it harder for malware actors to get users to run macros, so malware actors use them less frequently. However, you may still come across them in your work.

The best tool for analyzing Office documents is oledump.py, a tool written by Belgian security researcher Didier Stevens (he also wrote emldump.py, which we used previously). It is included in REMnux, which you probably set up in Chapter 6. If not, the video in the next exercise explains how to install it.

**Exercise 10.6.** Watch Didier's [YouTube workshop](#) on analyzing malicious documents. As you probably have REMnux installed, you can skip the first video about setting up oledump, but keep the following in mind if you want to follow along with what Didier does:

- If you want to run oledump on REMnux, just run oledump.py followed by the arguments, not `./oledump.py` as Didier does<sup>53</sup>.

<sup>53</sup> If you run a command on the command line, Linux looks for an executable file with that name in one of several directories (`echo $PATH` shows you which ones). If you want to run a file from the current directory, you need to add that directory and in Linux, the current directory is given by a single dot. This explains why in the video, where oledump is installed in the local directory, the command is preceded by `./`



- Precede plugin names with /opt/oledump-files/ as they are stored in that directory.
- This is a long workshop, and we recommend you just sit back and watch the video (possibly at a slightly higher speed; it can be a little slow) and not focus on all the details. It's really okay if you don't understand or remember everything. Just make sure that at the end, you know:
- How to use oledump to show the various parts of an Office file (exercise 1)
- How to use oledump to show VBA macros embedded in a document (exercise 6)
- That malicious VBA macros often use obfuscation of code (exercise 17) and strings/URLs (exercise 20)
- The Linux commands `less` and `head`. These are not explicitly explained in the workshop; [this website](#) [wayback machine](#) is a good introduction if you're not familiar with them.

**Question 10.7.** Didier explains why downloaders are generally more beneficial for malware authors than droppers. Can you think of an advantage droppers have over downloaders for malware authors? (See the appendix for the answer.)

Now, let's use oledump in practice. We'll do this on the file with sha256 hash `3d76f59c4dceb13546eb9c72a7c0f03fd335093583d326de9a314f3dbd5a77cc` that was uploaded to MalwareBazaar just before writing this guide. As is common, the download is a zip file protected with the password "infected."

MalwareBazaar already provides a lot of other information about the file that can be useful if you want to analyze it. But let's pretend we found this file attached to an email, and there is no public information about it available.

As Didier explained in his videos, oledump works on the zip file. We can use it to find if the file contains VBA macros, which it does at part 8:

```
remnux@remnux: ~
remnux@remnux:~$ oledump.py 3d76f59c4dceb13546eb9c72a7c0f03fd335093583d326de9a314f3dbd5a77cc.zip
1: 114 '\x01CompObj'
2: 4896 '\x05DocumentSummaryInformation'
3: 4896 '\x05SummaryInformation'
4: 7401 'Table'
5: 15599 'Data'
6: 441 'Macros/PROJECT'
7: 41 'Macros/PROJECT.m'
8: M 4850 'Macros/VBA/ThisDocument'
9: 3304 'Macros/VBA/ VBA_PROJECT'
10: 523 'Macros/VBA/dir'
11: 4896 'WordDocument'
remnux@remnux:~$
```

Now we can show the macros:

```
remnux@remnux:~$ oledump.py -s 8 -v 3d76f59c4dceb13546eb9c72a7c0f03fd335093583d326de9a314f3dbd5a77cc.zip
Attribute VB Name = "ThisDocument"
Attribute VB Base = "Normal.ThisDocument"
Attribute VB GlobalNameSpace = False
Attribute VB Creatable = False
Attribute VB PredeclaredId = True
Attribute VB Exposed = True
Attribute VB TemplateDerived = True
Attribute VB Customizable = True
Private Declare Function szergJHBjvcsiduhusigwqggfefevevbwsgfrbsjgbfvhjvJGyhjvgvIUGFyviyglbfohdfugdkgYgvkhvjvGKME
et Lib "shell32.dll" Alias
"ShellExecuteA" (ByVal dgXwZECK As Long, _
ByVal Ytkboyh As String, _
ByVal hUUVeOtmSPiuiFYroLV As String, _
ByVal FLHTKlGIRpWAPwCveT As String, _
ByVal zuttyjcyjNoZqtW As String, _
ByVal oMQEApJarEoyxLWYUhdvudgk As Long) As Long

Private Declare Function QuHGwbduuJB Lib "urlmon" Alias _
"URLDownloadToFileA" (ByVal mTVqSxmIQeDEfoAdw As Long, _
ByVal bzRRR As String, _
ByVal dmpXfprDTIqsBYTzgmFN As String, _
ByVal CjeediS As Long, _
ByVal ZhxXm As Long) As Long

Sub tKwABMGYVsOnfbnDsoc()
Dim mhVBKLJozszLKJnkjBBjvCgIUGuyVcgHCgFxcGDFcG As String
Dim erqJHBjvcsiduhusigwqggfefevevbwsgfrbsjgbfvhjvJGyhjvgvIUGFyviyglbfohdfugdkgYgvkhvjv As String
Dim YGUGfcFCDRsWXcgVjBhbikhhIBHiGvycDRxxrdcyGhkj As String
Dim lIDhjPjpxmRbcZfCIoiJiBBhvcFeDeersECytfTghGvGhdDxsErXfzDxfD As String
Dim QuHGwbduuJBmTVqSxmIQeDEfoJohUJBgVfcSxERrcuihBbhvvcfxyXfYgUvgvVhGyGyvhj As String
Dim UyguiGoAdwbzRrRdMpxvJvJvHVGlobfvjsefVjeIfuwewiou As String
erqJHBjvcsiduhusigwqggfefevevbwsgfrbsjgbfvhjvJGyhjvgvIUGFyviyglbfohdfugdkgYgvkhvjv = UubhuYbfhf("fyf/ddcc")
```



Not all macros fit on the screen. If you run this yourself, you can add `| less` at the end of the command to walk through the full output. You can use the scrollbar on the right of the terminal window to see more.

Let's step back for a moment. The file's creator used heavy obfuscation and VBA macros. Even if you are not a programmer, this should make you almost certain that this file is malicious. If you are performing your analysis just to check if it is malicious, you can stop here and draw your conclusions. But you may want to continue with your analysis to understand more about this file.

Given that downloaders are more common than droppers let's look for URLs. Didier's `http_heuristics` plugin doesn't work here (try this yourself to confirm), and that's not too surprising. If it were that easy to extract a URL from the document, security products would do so, and they might check the domain or URL against a blocklist. As a consequence, malware authors hide URLs pretty well and keep finding new ways to do so.

So, let's look for strings in the document: anything between double quotes (" . . . "). There are about a dozen of these, but most are too short to contain an obfuscated URL. The one exception is the string `fyf/higehsvj0tuofuopdeffg0npd/ujmj1ufn/xxx00;tquui`.

Now you can do two things. You can note that this string is the argument of the function `Uubhuyfbhf`, which is defined further in the code. If you understand a bit of programming, you can figure out what this function does and note that, indeed, it de-obfuscates the string to a URL.

You can also look at the string more carefully and note that a URL starts with `http://` or `https://`, which has a double `t` and a double `/`. At the end of our string, there is a double `u` and a double `0`. If you look a bit more carefully, you notice that `tquui` spelled backward is `iuuqt` and that is `https` shifted one letter in the alphabet.

If you then notice that `:` is followed by `;` in the [ASCII table](#) and `0` is followed by `/`, then you've figured the encoding out: to decode the string, we need to reverse it and then, for each character, take the previous one in the ASCII table. If you can program, you can write a short script that does that, or you can just manually de-obfuscate the string and find that it gives<sup>54</sup>

```
https://www.metkilit[.]com/feedcontents/iurgdfhg.exe
```

Indeed, if we had looked for our sample on [VirusTotal](#), we would have noticed that it does connect to the domain `www.metkilit[.]com`. But remember, in this experiment, we pretended there was nothing known on the file.

Now that we have a URL that is presumably where the malware looks for downloads, we can investigate this further if we want to. Later in this guide, we'll briefly look at analyzing URLs.

It is one thing to understand how the URL obfuscation works. It's another thing to be able to find it yourself. Two things are needed for this: luck and experience. Luck because the obfuscation, in this case, was fairly simple and also because you just need to see it. When you perform an analysis like this, it is always beneficial to work together: two (or more) people are far more likely to spot some pattern than one person.

---

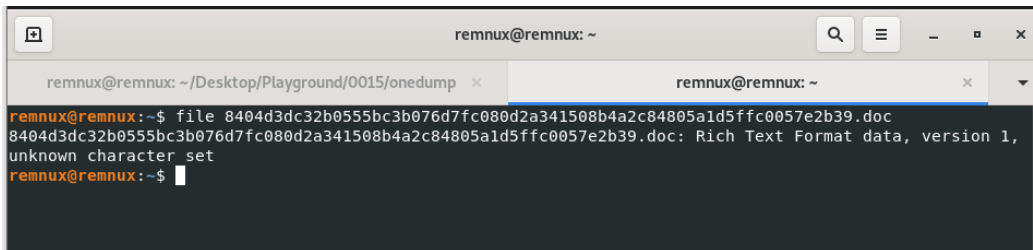
<sup>54</sup> Remember the good practice to add square brackets around the final dot in a domain name to 'defang' it; see Chapter 7.



Now let's look at another piece of malware:

8404d3dc32b0555bc3b076d7fc080d2a341508b4a2c84805a1d5ffc0057e2b39, the sample that we analyzed in Triage above. If we run oledump on the zip file, we get an error that it is not a valid zip file. You may think that unpacking the file helps<sup>55</sup>, but running oledump on the .doc file gives the same error.

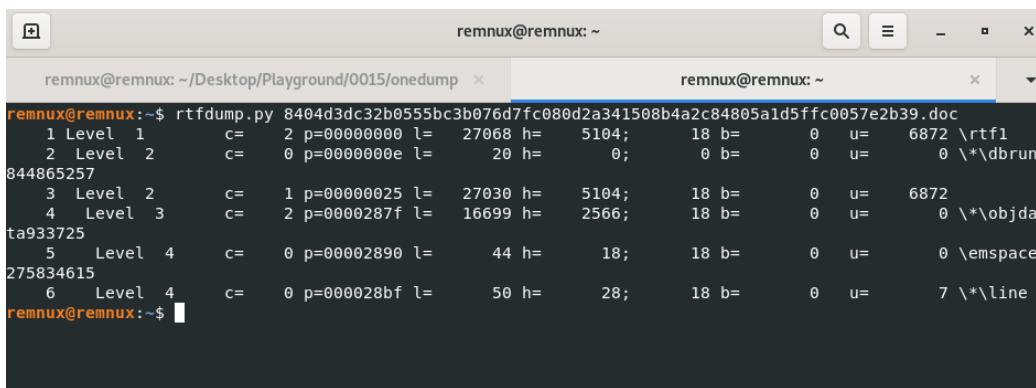
Thankfully, Linux has a helpful command `file`, which tells us this is a Rich Text Format file:



```
remnux@remnux: ~  
remnux@remnux: ~/Desktop/Playground/0015/onedump  
remnux@remnux:~$ file 8404d3dc32b0555bc3b076d7fc080d2a341508b4a2c84805a1d5ffc0057e2b39.doc  
8404d3dc32b0555bc3b076d7fc080d2a341508b4a2c84805a1d5ffc0057e2b39.doc: Rich Text Format data, version 1,  
unknown character set  
remnux@remnux:~$
```

Such RTF files (which often have the .rtf extension, but as you see, also come as .doc) have a different format compared to ordinary Word files. Thankfully, there is another tool Didier has written that is also included in REMnux: `rtfdump.py`.

As oledump, `rtfdump` gives you the various parts the document consists of.



```
remnux@remnux: ~  
remnux@remnux: ~/Desktop/Playground/0015/onedump  
remnux@remnux:~$ rtfdump.py 8404d3dc32b0555bc3b076d7fc080d2a341508b4a2c84805a1d5ffc0057e2b39.doc  
1 Level 1 c= 2 p=00000000 l= 27068 h= 5104; 18 b= 0 u= 6872 \rtf1  
2 Level 2 c= 0 p=0000000e l= 20 h= 0; 0 b= 0 u= 0 \*\dbrun  
844865257  
3 Level 2 c= 1 p=00000025 l= 27030 h= 5104; 18 b= 0 u= 6872  
4 Level 3 c= 2 p=0000287f l= 16699 h= 2566; 18 b= 0 u= 0 \*\objda  
ta933725  
5 Level 4 c= 0 p=00002890 l= 44 h= 18; 18 b= 0 u= 0 \emspace  
275834615  
6 Level 4 c= 0 p=000028bf l= 50 h= 28; 18 b= 0 u= 7 \*\line  
remnux@remnux:~$
```

Similar to oledump, we can dump the contents of each section. Unfortunately, things aren't as simple from here. When an Office document contains malicious VBA macros, the malware is a feature of Office – even if it is a rather unwanted one. But in this case, the malware acts as a bug. More precisely, it uses a vulnerability in Office named CVE-2017-11882. That makes the malware harder to find, in addition to the difficulties caused by obfuscation in the malicious code.

To decode the malware, an analyst would probably have to understand the inner workings of RTF files and the CVE-2017-11882 exploit and might even have to adapt the `rtfdump` tool. Indeed, Didier keeps adding new features to his tools in response to new challenges like these.

Being unable to decode malware can be frustrating and a little bit embarrassing! It is also the reality of analyzing malware. An important lesson with this example is we were able to analyze the file in Triage, which gave us all we needed. This provides an extra argument for using sandboxes instead of only relying on manual analyses.

<sup>55</sup> Remember from chapter 7 you will probably need to run `7z x [file]` rather than `unzip [file]`





**Exercise 10.8. (optional)** For reference, here is an [analysis](#) of a different RTF file that could be analyzed using rtfdump. Can you find the malicious code inside the RTF document using tools in REMnux?

Experience is gained by practice and by reading what others do. That is a long process, and even then, advanced malware analysts very often get stuck. Don't let that discourage you, and don't feel like you have to be able to handle any attachment. Even experienced analysts regularly get stuck.

## PDF attachments

PDFs are other common attachments. There was a time when vulnerabilities in Adobe Reader, the most popular PDF software, were very common. Using malicious PDFs was a common way to infect devices running a slightly out-of-date version of Reader.

This is less common these days, but malicious PDFs still exist. Sometimes, another document is embedded in them, such as a malicious Office document. Other times, they just contain a link that downloads the next-stage payload.

Again, Didier Stevens has built several [tools](#) for PDF analysis. He has also created a [video workshop](#) to demonstrate some of these tools. However, this workshop is 11 years old and mostly focuses on the kind of PDF malware that was common at the time. If you regularly deal with PDF files, you may still benefit from watching the workshop, but consider it optional.

Didier's main PDF tool is pdf-parser.py, which, as the name suggests, parses a PDF file.

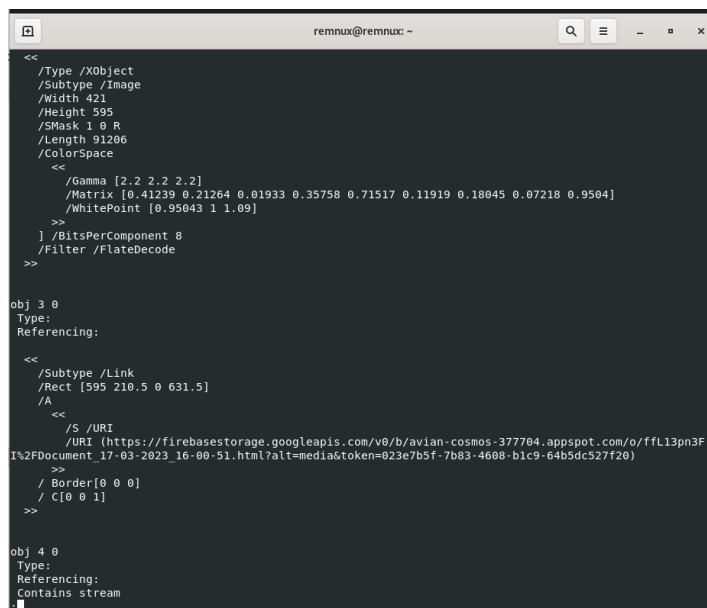
Let us again take a sample downloaded from Malware Bazaar:

[26509fa876a07966824bed8e5b0a6e0626b37d68355871fac49b2fa636d7fe6e](https://www.malwarebazaar.com/sample/26509fa876a07966824bed8e5b0a6e0626b37d68355871fac49b2fa636d7fe6e).

First, we run:

```
pdf-parser.py
26509fa876a07966824bed8e5b0a6e0626b37d68355871fac49b2fa636d7fe6e.pdf |less
```

on a single line to browse through the various components of the file. Note how a PDF file consists of various numbered objects. If we browse through the objects, we notice a URL in object 3, hosted on Google's Firebase storage. This is a legitimate service (if you didn't know this, a quick Internet search would have told you this), but it is [commonly used](#) to host malware.



```
remnux@remnux: ~
<<
  /Type /XObject
  /Subtype /Image
  /Width 421
  /Height 595
  /SMask 1 0 R
  /Length 91206
  /ColorSpace
  <<
    /Gamma [2.2 2.2 2.2]
    /Matrix [0.41239 0.21264 0.01933 0.35758 0.71517 0.11919 0.18045 0.07218 0.9504]
    /WhitePoint [0.95043 1 1.09]
  >>
  ] /BitsPerComponent 8
  /Filter /FlateDecode
>>

obj 3 0
Type:
Referencing:

<<
  /Subtype /Link
  /Rect [595 210.5 0 631.5]
  /A
  <<
    /S /URI
    /URI (https://firebasestorage.googleapis.com/v0/b/avian-cosmos-377704.appspot.com/o/ffl13pn3F
I%2FDocument_17-03-2023_16-00-51.html?alt=media&token=023e7b5f-7b83-4608-b1c9-64b5dc527f20)
  >>
  /Border[0 0 0]
  /C[0 0 1]
>>

obj 4 0
Type:
Referencing:
Contains stream
```

And that's it! We extracted the URL from the PDF, and that's all you need to know. (To be sure, you can check the other objects in the file to confirm there is nothing else.)

Another PDF to look at is [907e75030b0e09cec6524f612f1c7439b5260b57b43d515968f81ba69278ba77](https://907e75030b0e09cec6524f612f1c7439b5260b57b43d515968f81ba69278ba77).

To browse through the various objects, we can run `pdf-parser.py` on this file, followed by `| less`. Unfortunately, this does not produce a URL, so we need to study the objects more carefully. It can be helpful to look at the lengths of the objects: many are 10 bytes long, meaning they are unlikely to contain anything malicious.

We do see some larger objects, starting with object 61, which says `/Subtype /Image`. As you can probably guess, these contain images embedded in the PDF. But then there is object 82, which is an embedded file (`/Type /EmbeddedFile`). Adding the `-o` option to the end of the command we just entered lets us show just this object:

```
remnux@remnux: ~  
remnux@remnux:~$ pdf-parser.py 907e75030b0e09cec6524f612f1c7439b5260b57b43d515968f81ba69278ba77.pdf -o 82  
This program has not been tested with this version of Python (3.8.10)  
Should you encounter problems, please use Python version 3.7.5  
obj 82 0  
Type: /EmbeddedFile  
Referencing:  
Contains stream  
  
<<  
  /Filter /FlateDecode  
  /Type /EmbeddedFile  
  /Length 181498  
>>
```

The `pdf-parser` tool helpfully extracts the object using the `-d` option (meaning 'dump') followed by a filename. Because the object is encoded (see `/Filter /FlateDecode`) we can also use `-f` to decode it.

Here, we store the extracted data in a file, `test`, and then use the `file` command to find what the object is. As it turns out, it is an Excel document! If we wanted to, we could analyze this further using `oledump`.

```
remnux@remnux: ~  
remnux@remnux:~$ pdf-parser.py 907e75030b0e09cec6524f612f1c7439b5260b57b43d515968f81ba69278ba77.pdf -o 82 -f -d test  
This program has not been tested with this version of Python (3.8.10)  
Should you encounter problems, please use Python version 3.7.5  
obj 82 0  
Type: /EmbeddedFile  
Referencing:  
Contains stream  
  
<<  
  /Filter /FlateDecode  
  /Type /EmbeddedFile  
  /Length 181498  
>>  
  
remnux@remnux:~$ file test  
test: Composite Document File V2 Document, Little Endian, Os: Windows, Version 6.2, Code page: 1252,  
Name of Creating Application: Microsoft Excel, Create Time/Date: Sat Sep 16 01:00:00 2006, Last Saved  
Time/Date: Fri Jan 27 01:34:39 2023, Security: 0
```



**Exercise 10.9.** Use the `-d` option in `pdfdump.py` to extract one image from the ODF.

**Exercise 10.10.** Use the `pdf-parser` tool to analyze the PDF file [ad517cb885ee279ec6ca95cd7402da998ec5461461f745c2f075085ef49b4eb6](https://ad517cb885ee279ec6ca95cd7402da998ec5461461f745c2f075085ef49b4eb6).

**Exercise 10.11.** (*optional*) Open the two PDFs in Triage, or another sandbox of your choice, and confirm what the manual analysis showed.

## Other kinds of attachments

There are many other kinds of attachments used to spread malware, and malicious actors keep finding new ones to use. As mentioned previously, you should not expect to know how to analyze them all. If you come across a new file type that you don't know how to handle, look on the Internet. It's very likely that someone has written a tool and/or a guide on how to analyze this particular file type.

That someone may well be Didier Stevens. It's worth following his [blog](#) and his posts on the [Internet Storm Center blog](#). For example, in the early months of 2023, he wrote how to analyze [OneNote files](#) [wayback machine](#) and [HTA files](#) [wayback machine](#).

## Analyzing links

Unlike a malicious document, a web link (or a URL) doesn't really contain anything; it merely links to a resource somewhere else. This could be an HTML page, a piece of malware, or even a page giving a 404 error. What the link returns sometimes depends on how and when your device makes the request. Many URLs cease to work after some time, which is particularly true for malicious ones. However, even when this is the case, analysis is still possible.

First, there is the domain name that was used. That is often a big giveaway when you are trying to decide whether the link is or was malicious. Refer to Chapter 7 for a reminder on how to use VirusTotal to get more information on a potentially malicious domain name.

Secondly, you can look at what loads when you try to access the link. Here you have to be lucky: the resource may have been removed or changed. Even if the resource is still there, the malicious server may not serve it to you based on your IP address, web browser, or some other characteristic. And, of course, you will want to download it in a safe environment, such as REMnux. The optional Exercise 10.12 below gives you some guidance on how to do this.

Generally, it is safer to open the URL in a sandbox. This won't help you if the resource has been removed or changed, but if it is still there, it is a lot less dangerous to open it in a sandbox. Remember that in Triage, the 'Downloads' section gives you access to the downloaded files in case you want to analyze them further.

In the early 2010s, 'drive-by downloads' were a common way to infect computers. A malicious or infected website would detect that your browser or browser plug-in (such as Flash Player or Java) was vulnerable and use that vulnerability to install malware on your computer. Today, modern browsers work to block risky plugins



and typically install security updates automatically, making such drive-by downloads a lot rarer. They do still exist, though, and in some cases, they exploit zero-day vulnerabilities in browsers.

For this reason, opening suspicious links in a browser is still not a good idea, even if the risk is a lot smaller than it once was.

**Exercise 10.12.** Look for a recent malicious URL in [URLhaus](#) – a website by the same people that also maintain Malware Bazaar – and open it in Triage or another sandbox of your choice. Confirm that the downloaded content is visible in the ‘Downloads’ tab in Triage. If nothing happens, try a different URL. (This exercise deliberately makes you look for URLs yourself rather than suggesting one.)

**Exercise 10.13. (optional)** Read [this wayback machine](#) blog post on using `curl` to download the content from a potentially malicious URL and try it out with some recent URLs from URLhaus.

## Vulnerable email clients

Finally, there is a different kind of malicious email that is rare but still worth mentioning: one where a vulnerability in an email client is exploited directly by a specially crafted email. In March 2023, it was [discovered wayback machine](#) that a Russia-linked hacking group known as Fancy Bear or APT28 (or several other names) had been using a zero-day vulnerability in Outlook to infect computers this way.

There is no generic way to analyze such emails. Attackers may use different kinds of vulnerabilities. They are very careful about using zero-days; often, attackers do not want to reveal their ability to exploit them, lest the developers of the email software learn of them and patch them. Thankfully, such cases are very rare. But this highlights the importance of making sure mail clients are kept up to date – something that web email clients do automatically.



## Chapter 11: Website incident response

This chapter is about performing incident response for websites. Most civil society organizations have a website, and issues with them are common. Sometimes, this is merely frustrating: the organization temporarily loses a public presence but is otherwise able to continue its activities. Often, though, a non-functioning website seriously hampers the organization's ability to perform its tasks.

This module introduces you to websites, web servers and web hosting, as well as the most common platforms used by civil society organizations to publish their content online. However, you shouldn't expect to know everything at the end of this chapter. As Tolstoy might have said, each functioning website is the same, but each broken website is broken in its own unique way.

Your goal for this module should be to feel more confident about the basics so that when you handle an incident, you can work to find the missing details yourself. This will likely happen through a combination of the following:

1. Searching the Internet — don't forget that no matter how experienced you are as a security analyst, a search engine will be your most important tool;
2. Becoming more comfortable with the system by poking around.

To help you with that confidence, this module contains exercises aimed at helping you to become familiar with web servers. You may already have some (or a lot of) familiarity, which is why these exercises are entirely optional and marked as such. Decide for yourself how much the exercises ask you to do things you already know; if they do, just skip them!

**Exercise 11.1. (optional)** Set up a Linux (virtual) machine to use as a server. You may have a preference for a particular distribution, but if not, the latest version of Ubuntu should<sup>56</sup> be fine. In the case of Ubuntu, you can download a version tailored for servers, so it's best to go for that. If, in Chapter 7, you set up VirtualBox (or a different hypervisor), you can create your server there. Alternatively, maybe you have an old computer lying around that you're not using anymore that can be used as a server. Or, you can rent a virtual private server (VPS) from one of many cloud hosting providers such as Greenhost, Digital Ocean, Linode, Vultr, or AWS. This isn't free, though fairly inexpensive<sup>57</sup> and means your server is directly connected to the Internet, which is more 'real'.

Basic servers aren't very heavy on resources, and while you can keep the recommended defaults, it's also okay to use 1024MB RAM and a 10GB hard drive. Other than that, you can keep the defaults.

### Background, terminology and techniques

This section contains some background on relevant terminology and techniques. You may be familiar with most or all of them, in which case it is fine to skip.

---

<sup>56</sup> Ubuntu, and many other distributions, designate certain versions as LTS, short for 'long term support'. This server is guaranteed to receive security updates for a very long time. If you want to deploy a server in a real situation, this is the one to go for. For testing purposes, it doesn't really matter that much, but using a LTS version never hurts!

<sup>57</sup> Make sure you remember to remove the server when you don't use it anymore to prevent being billed for it in the future.



## Websites and web servers

A **web server** is a computer that serves **websites** to clients, typically **web browsers**. The two most used kinds of software that run web servers are Apache and nginx. Websites use a markup language called **HTML** to display content, with presentation taken care of by **style sheets** that use a language called **CSS**. Web servers typically keep **logs** of all the requests, which can be very helpful during investigations. A **content management system (CMS)** is an online system that can be used to manage a particular website; common examples include WordPress and Joomla.

It's probably not necessary to know the definitions of these terms specifically, but if you feel unsure about their exact meaning, don't hesitate to look them up!

## HTML

**HTML** is a markup language that web pages are written in. This is what the HTML of a very simple web page looks like:

```
<!DOCTYPE html>
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <div>
      <p><a href="https://www.internews.org/">This</a> is a link to the
<b>Internews</b> website.</p>
    </div>
  </body>
</html>
```

You don't need to be an expert in HTML to perform incident response on websites, but it helps to have a basic understanding and to be able to recognize tags, such as `<head>`, `<p>` and `<a>`, and attributes with their values, such as `href="https://www.internews.org/"` in the example above.

If you want to learn basic HTML, W3Schools has a good [interactive tutorial](#).

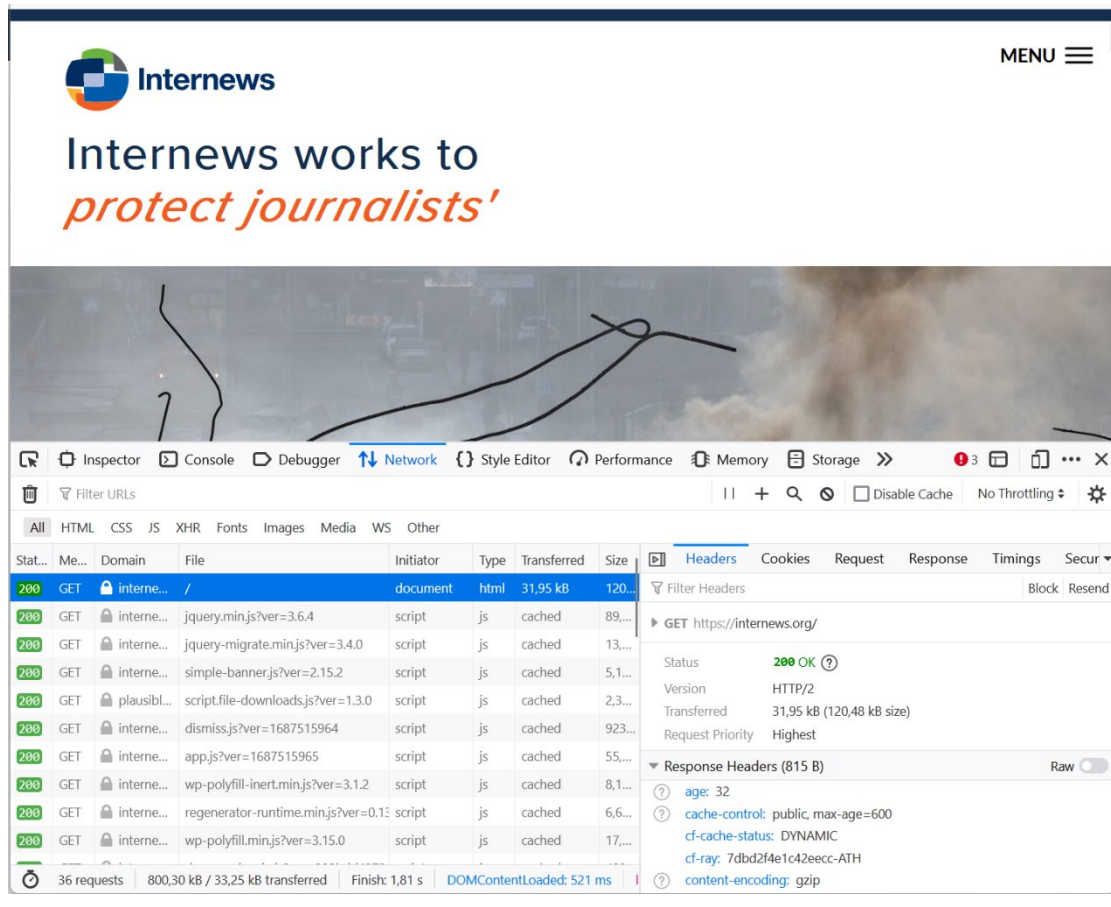
You can also view the source of various websites in your browser to get a good understanding of HTML but do note that it is very common for the content of web pages to be dynamically altered. As a result, the source often differs a lot from what you see in the browser. To view the actual HTML displayed by the browser, in most browsers, you can inspect this by right-clicking on the page and choosing 'Inspect.'

## HTTP

The protocol used to make requests to websites is **HTTP**, the **H**ypertext **T**ransfer **P**rotocol. In HTTP, a client sends a request over TCP to a web browser. The request starts with a request line that states the request method, the requested URL and the protocol version. The request line is followed by a number of headers that have the same format as email headers and an empty line.

If you're not very familiar with HTTP requests, note that most modern browsers allow you to inspect the traffic (similar to inspecting the HTML) and see HTTP requests and responses. It's a good idea to try to understand what happens when your browser makes a request to a website. Even better if this is a website run on a server you have access to, as you can also see the requests in the server logs.

One thing you'll notice if you do so is that a single HTTP request is usually followed by many subsequent requests, for example, to load images, style sheets and JavaScript code, with some of these requests coming from different web servers.



Analyzing the HTTP traffic related to a request to *internews.org* in Mozilla Firefox. You open this view by right-clicking anywhere on a website, choosing 'Inspect' and then going to the Network tab.

Here is a simple HTTP request to the Internews homepage, with some headers left out:

```
GET / HTTP/2
Host: internews.org
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
```

In this case, the method used, as seen in the request line, is **GET**. That is the most common request method used to request content from a web server. There are several other methods, though **POST** and **HEAD** are the only ones you're likely to encounter. Unlike GET, a POST request comes with a body that follows the empty line after the headers. POST requests are used, for example, to submit form content or to upload data to a website.





HEAD works the same as GET but only gets headers back. It is sometimes used for debugging or by bots to confirm whether a certain page exists on the website. You may thus see HEAD requests in your server logs.

The second part of the request line is the URL, or more precisely, the part of the URL following the hostname. In this case, there is just a / so the URL is `http://internews.org/`, the homepage of Internews.

Often, the URL (or at least the part before the query string – more on which in a bit) corresponds to a particular file on a web server, with the URL providing the location relative to the ‘web root,’ the part on the web server where the website is hosted. If the web root is `/var/www/html/`<sup>58</sup> then the URL `/foo/bar.html` will correspond to the file `/var/www/html/foo/bar.html` on the server. However, this certainly isn’t always the case. For example, WordPress usually uses ‘pretty URLs’ while the file called by the web server is `index.php` in the web root.

The third part of the request line is the protocol, in this case, HTTP version 2. Other common versions you may see are 1.1 – used since 1996 – and the newer version 3. The HTTP version is rarely relevant for you when you’re investigating a website.

## HTTPS

Most websites use **HTTPS**, often referred to as a secure version of the HTTP protocol. That is technically not true: HTTPS is just HTTP, with the difference being that the underlying connection is encrypted using TLS. The use of TLS and certificates ensures that the connection between the browser and the web server is encrypted.

The underlying protocol is still HTTP, though, and in server logs or when you view requests in a browser, you won’t notice any difference.

HTTPS is increasingly common for websites thanks to services like [Let’s Encrypt](#), which make it both free and easy to set up. If a website only uses standard HTTP, anyone able to listen in on the network – such as governments or Internet providers – can read, modify and block content<sup>59</sup> in transit. It is important to note that not using HTTPS doesn’t typically make it easier to hack a website or web server. In practice, however, not using HTTPS is often a sign of poor attention to security by the entity hosting the website, and many more serious things may be wrong with the website.

## Query strings and URL encoding

It is common for URLs to contain a **query string**. This is a string of key-value pairs followed by a question mark in the URL, separated by an ampersand (&). For example:

```
https://www.example.com/search?query=foo&option=bar
```

In this case, the browser ‘sets’ the key `query` to have the value `foo` and the key `option` to have the value `bar`. You no doubt have seen this used, for example, in search engines. Note that this is a way for browsers to ‘send’ data to the server other than POST requests. While POST requests can send much larger data, the URLs of GET requests are visible in the browser, allowing users to bookmark them and you, as an investigator, to see them in logs. That can be really helpful when looking into an incident.

---

<sup>58</sup> This notation assumes the web server runs Linux or a related operating system. This is in practice almost always the case, especially for the kinds of websites you will end up investigating. Many larger websites run on Microsoft’s IIS though.

<sup>59</sup> With HTTPS – and TLS in general – content cannot be seen or modified, but it is often still possible to see what server someone is connecting to. This can be used to censor certain sites, such as YouTube or the New York Times, but cannot be used to censor content within such sites, such as blocking specific videos or individual NYT articles.





To handle the case where, say, a value in the query string contains an equal sign, URLs are encoded using **URL encoding**. URL encoding is also used to encode non-ASCII characters, though many browsers will display these in their original form.

You don't need to understand the details of how URL encoding works, but having a basic grasp of it is probably a good idea, especially if you deal with websites in languages that use non-ASCII characters. The [Wikipedia article](#) is a good place to start and there are various [websites](#) that allow you to encode or decode URLs.

## Headers

Headers are a way for a client to send extra contextual information to the request. As mentioned before, they take the same form as email headers.

The most important header is the **Host** header. This tells the web server what hostname the request is for so that it, together with the request line, can determine the full URL. The presence of the hostname in the headers mean that a web server can handle requests for more than one hostname, and many do indeed<sup>60</sup>. On web servers, different hostnames often have their own log files.

A very important header is the **User-Agent**, which defines the browser (also called **user agent**) that is being used. In the example above, the agent is version 113 of Firefox, on a modern version of Windows.<sup>61</sup>

The user agent, which is often present in the server logs, is very helpful in understanding more about a particular request. It is important to note that the user agent is in no way authenticated, and there is never a guarantee that it is accurate. Web clients providing an accurate User-Agent header is nothing but good practice. For example, Google's crawler bot will clearly identify itself in HTTP requests, but many less benign bots will use the same user agent.

The various **Accept** headers are for the client to inform the server what kind of content it accepts. This is rarely useful for investigations, but these and other headers help provide the web server with a 'fingerprint' of the browser.

Occasionally helpful is the awkwardly misspelled **Referer** header, which indicates the URL from which the user accessed the page. It is not reliable, and browsers sometimes leave it off for privacy reasons, but if present, it will usually show up in the server logs. This can be helpful if you're investigating a website suddenly receiving a lot of traffic: you may discover another website linking to it.

There are many more headers sent by the browser. Except for cookies, more on which in a bit, you don't need to know them all or understand what they do, but if you're curious, look at some requests sent by your browser as it visits various websites. Beyond the scope of this guide, but something to be mindful of when you work with at-risk people and groups: headers could be used to [track an individual user](#).

## IP address

The connecting IP address is a very important artifact of the HTTP connection. It is used by some websites to deliver content specific to a particular region (think of localized versions of Google depending on your country) or even to block access from users in a particular region.

If you are investigating a web server compromise, you may be looking for certain website requests in the server logs. The IP address, usually present in these logs, is a great way to link various requests to the same

---

<sup>60</sup> This wasn't possible in version 1.0 of HTTP, which was used in the very early days of the web.

<sup>61</sup> There are [websites](#) that help you 'decode' a User-Agent header, but they aren't always accurate. For example, the example request was made on a Windows 11 desktop, while that site claims it was made on Windows 10.



user, whether it is a human user or a bot. You should beware of drawing conclusions about the location of the user – they may have used a VPN or Tor – but even on these services, IP addresses don't change very often.

## HTTP responses and status codes

In response to the HTTP request, the server sends an HTTP response. Like the request, this consists of two or three sections.

The first section is the status line, which looks as follows:

```
HTTP/1.1 200 OK
```

The status line consists of three parts: the HTTP version, the **HTTP status code** and a human-readable text explaining the response. Only the response code is relevant here. It will also appear in web server logs. Wikipedia has an [overview](#) of the various response codes.

The most common status code is 200, which means the request was successful. Generally, any response starting with a '2' means the response was successful. 1xx responses are informational (you probably won't come across them very often), 3xx responses indicate a redirect<sup>62</sup>, 4xx a client error (such as a user not being authorized to access the page or a page not present: the famous 404 response) and 5xx indicates a server error.

Errors can be very helpful when looking for an issue on a web server, but there are two important things to keep in mind.

First, not all errors experienced by the users will result in a 4xx or 5xx status code. For example, a regular user logged into a WordPress site trying to access a page restricted for administrators will generate a normal 200 response: the error, in this case, is generated not by the web server but by the PHP code.

Secondly, any web server on the public Internet will receive many requests from bots that try to look for certain content, for example, to find common vulnerabilities in the web server. Many of these requests result in errors, in particular 404 ones, because these URLs don't exist. This isn't anything to worry about; it just shows that a probe fails.

The status line is followed by a set of headers. To distinguish them from the headers sent by the browser, they are referred to as **response headers**.

Their format is the same as that of request headers. Some common headers are:

- Set-Cookie, which sets cookies;
- Content-Length, which tells the browser the size of the response;
- or Content-Type, which informs the browser of what type of content is sent to the browser, for example, whether it is an HTML page or a PNG image.

One loose group of response headers is collectively referred to as **security headers**. They mitigate various common attacks against web servers and websites, for example, by preventing the browser from sending the source in the Referer header if a link is clicked – which can be important for very privacy-sensitive websites – or by limiting what third-party content can be requested by the website, which could mitigate a type of attack called 'cross-site scripting' (or XSS).

---

<sup>62</sup> To be precise, a redirect at the server level. Sometimes web pages include JavaScript code that automatically load another page. That won't result in a 3xx redirect.



While security headers are a good thing in general and are something to consider when setting up or securing an organization's website, they probably aren't relevant when you're handling an incident. The website [securityheaders.com](https://securityheaders.com) allows you to check what, if any, security headers are present on a website.

Finally, the last section for most HTTP responses (but not for responses to HEAD requests, as well as for some responses where the HTTP status code is a redirect) contains the actual content of the response.

**Exercise 11.2.** Inspect the headers sent when you visit your organization's website or that of a partner. Also, look for the headers that are returned. Do you see any cookies being sent and/or returned? Do you notice anything unusual?

## Cookies

The term (web) **cookie** is familiar to anyone who has ever browsed the web. A cookie is a small piece of data that a web server sends to a browser (in a Set-Cookie response header) as part of the HTTP response. When a browser receives this, it acknowledges it, and if accepted by the user, the cookie is then included in subsequent requests (in a Cookie request header).

Cookies have developed a notorious reputation because they are capable of tracking an individual's activity across various websites.<sup>63</sup> but they also can provide crucial functions of a service, such as maintaining a 'session': HTTP natively doesn't link requests to each other, so it doesn't "know" what pages you have previously visited or whether you are logged in – or even the concept of being logged in<sup>64</sup>.

In a Content Management System (CMS), cookies maintain user sessions and can also collect user data with the aim of providing a seamless user experience.

If a malicious actor obtains a session cookie from a device, they may be able to 'continue' the session on their own device. **Infostealers** (short for information stealing malware) commonly steal session cookies from machines they have infected, as do rogue browser extensions. This is mostly relevant when you are looking into account hacks – many account hacks are in fact session hijacks – but it is worth noting that a session hijack, for example, of an administrator on the CMS, could lead to a website compromise.

## Dynamic content: PHP and JavaScript

In its most basic form, a website contains static content that would be served to anyone visiting the site. That's a great way to share information, but it is rather limiting considering the possibilities available on the public Internet. Most websites these days generate content dynamically.

Content could be made dynamic on the server side or on the client side. In the former case, the server sends a response that is customized in some way. In the latter case, content is served that contains code that the browser will customize. Almost all modern websites use a combination of both techniques.

There are many frameworks to create server-side dynamic content. While it is certainly not the only framework, the one you are most likely going to come across is **PHP**. CMSes like WordPress, Drupal and Joomla are written in PHP.

---

<sup>63</sup> Cookies aren't shared among websites, but many websites include advertisements from third-party servers. It is these ads that are used to track someone across multiple websites and lead to very targeted ads on websites you have never visited before.

<sup>64</sup> This isn't strictly true: there is something called [Basic access authentication](#). This is sometimes used for internal websites but rarely on the public Internet these days.

PHP is a programming language that is embedded in HTML code. A PHP page contains a mixture of code and HTML markup language. You don't need to be able to understand PHP to investigate website incidents, but having a very general idea of how it works might be helpful. [Wikipedia](#) has the following example of PHP.

```
<!DOCTYPE html>
<html>
  <head>
    <title>PHP "Hello, World!" program</title>
  </head>
  <body>
    <?php
      echo '<p>Hello, World!</p>';
    ?>
  </body>
</html>
```

The bit between `<?php` and `?>` is PHP code. It uses the `PHP` `echo` to add some text to the HTML output. In this case, it's just the phrase 'Hello, World,' but it could also include, for example, the name of the user visiting the website if the PHP code would know it.

There are three potential security risks related to PHP. The first is when a PHP file includes user-configurable content, for example, parameters in the query string, that are included in the output code<sup>65</sup>. If this content isn't properly checked, it could potentially allow someone to make the web server do something it's not supposed to do, such as modify content.

A second potential security risk is if someone can upload a PHP file itself. It is common for content management systems to allow for the uploading of images, and if it fails to properly check whether the uploaded file is really an image – and if it allows the uploading of images by non-admin users – this could lead to someone uploading a PHP file that, when called by a browser, runs commands on the web server. This is commonly referred to as a [web shell](#) because it gives an adversary the same kind of access as if they had access to a shell (that is, a command line) on the web server.

A third potential security risk is if there is some kind of vulnerability in PHP itself. While it is rare for this to be the root issue of a website incident, it is nevertheless good practice to keep PHP as up-to-date as possible.

Sometimes, during a website compromise, the core PHP configuration file, `php.ini`, is changed. Though this is less common on modern web servers, it is something to keep in mind during an investigation, and it won't hurt to check if `php.ini` has been changed. Sucuri wrote a [useful guide](#) [wayback machine](#) on how this file works. It is most important to look for the file's location and then use the `ls` command to see if it has been changed recently<sup>66</sup>.

The most common way for dynamic content to happen on the client side is **JavaScript**, a programming language that is interpreted by all modern web browsers and included in almost all modern websites.

Two potential security issues exist related to JavaScript. The first is when user-controllable content can modify the content of a web page and make it do things it is not supposed to do, something called **cross-site scripting** or **XSS**. XSS vulnerabilities are less common than they once were, but they are still sometimes

---

<sup>65</sup> In the PHP example above, the code could have included the statement `echo $_GET['name'];` which would add the contents of the `name` parameter from the query string to the HTML of the website. Without any kind of security check, this would allow someone to submit a name that includes HTML or JavaScript code that does malicious things on the web server.

<sup>66</sup> Note that a server update may also change `php.ini`, so a recent modification doesn't always imply something malicious!



found in websites or web frameworks. You don't need to understand the details of how XSS works, but [Wikipedia](#) is a good source for a casual introduction.

The second potential issue is linked to browsers. Handling code from Internet sources is a very difficult thing to do securely, and there have been several instances of vulnerabilities in browsers that allowed for specially crafted JavaScript code – and in the past also Java and Flash – that would allow for the running of malware on the device.

As browser security has improved a lot and browsers are often updated automatically, mass infections like this (through “**drive-by downloads**”) aren't very common anymore. In more targeted attacks, they do occur and sometimes include zero days. In particular, sometimes a legitimate website is compromised, and malicious JavaScript code is inserted to target visitors of this website, a technique known as a **watering hole**<sup>67</sup>. While pretty rare, this is something to remember when working with highly targeted communities.

In general, you don't have to worry about the details of PHP and JavaScript security issues. You just need to remember that vulnerabilities in content management systems and the plugins that come with them are regularly found and that these are the main sources of website hacks.

## Accessing a web server

When performing analysis on a web server, you will need to access that web server. It is unlikely that you can have physical access to it with a mouse and a keyboard: the server is most likely<sup>68</sup> hosted in a data center somewhere, probably as a virtual machine.

To do this, you'll probably use a protocol called SSH, a secure way of opening a 'shell' (think of a terminal) on a remote server. The command (in a Terminal window on Mac or Linux, or the command prompt on Windows<sup>69</sup>) you use for this is `ssh username@hostname` where `hostname` may also be the IP address. Digital Ocean has a helpful [tutorial](#) [wayback machine](#) for setting up SSH on a server, including some good security tips, but if you're just using it to connect to a server set up by someone else, the above command suffices.

Whether you use a Linux terminal on your local machine or use SSH to connect to a remote server, you will need some familiarity with the Linux command line. Here, too, Digital Ocean has a helpful [tutorial](#) [wayback machine](#), but there are many more, and don't hesitate to look for other tutorials (including videos, if that is your preferred format for learning new skills!).

One aspect of Linux that is very relevant is that of different users and permissions. Once again, there is a Digital Ocean [tutorial](#) [wayback machine](#), but the important thing to keep in mind is that there may be different users and groups of users on a system and files and directories may have access restricted to a particular user or group, or may be accessible to everyone. Moreover, there is an all-powerful root user that can, in principle, access everything. When investigating a web server, you will probably need to access certain files, such as server logs, as root.

However, rather than logging in as root – which is not only a bad idea but also disabled on many systems – you should run the commands as your own user but precede them with `sudo`. For example, to install nginx on a system, you would have to run `apt install nginx`

---

<sup>67</sup> The metaphor that gave the name to this kind of attack is looking for animals in the desert by going to the place where they drink water. In this case, an adversary would target a particular community by infecting a website they would visit anyway, for example a media site covering news for that community.

<sup>68</sup> But not always: for about a year, the web server of a company I worked at was an old desktop on the floor. It did have a big sticker on it warning people not to turn it off!

<sup>69</sup> If you have used the [PuTTY](#) tool in the past, you may be happily surprised that SSH works natively in Windows these days. Of course, PuTTY still works fine too.

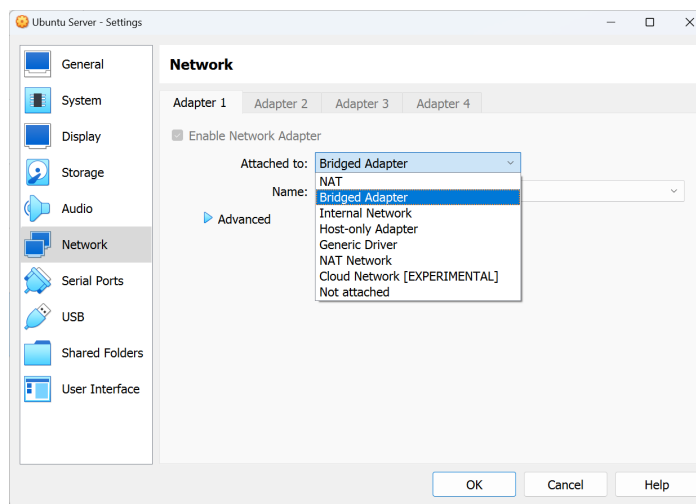


as root, but instead, you run `sudo apt install nginx` as your regular user. For a bit more on `sudo`, refer back to Chapter 4.

**Question 11.3.** Why is it better to use `sudo` than to log in as root? (See the appendix for the answer.)

**Exercise 11.4.** (optional) Connect to the server you previously set up using `ssh`. If your server is accessible from the Internet, you can just use the IP address assigned by the hosting provider. If you're using VirtualBox, you will have to do two things to be able to connect to the virtual machine from the host device.

First, in the settings for the virtual machine in Virtual Box, go to 'Network' and then change the first adapter from 'Attached to NAT' to 'Bridged Adaptor', as explained [here](#) [wayback machine](#).



Then, in a terminal, run the command `ip a` to find the machine's IP address, as explained [here](#), [wayback machine](#) and use this to SSH from the host to the virtual machine<sup>70</sup>. (Should you have trouble with this step, just to try out SSH, you can also run `SSH localhost` on the machine itself. This is not the most useful way of using SSH, but it will at least allow you to try out the command.)

**Exercise 11.5.** On the server, create a new user `test` (using the `adduser test`). Confirm this user doesn't have `sudo` access (run `sudo ls` and see what it says) and then follow [these instructions](#) [wayback machine](#) to give this user `sudo` access.

**Exercise 11.6.** On the server, install `nginx` using the command mentioned above<sup>71</sup>. Confirm it works by entering the IP address found in Exercise 3 in your browser.

<sup>70</sup> If you decided to install a different distribution, this command may be different.

<sup>71</sup> Anecdotally, `nginx` is more commonly used than `Apache` these days. If, for some reason, you think you're more likely to encounter `Apache` in your work, you can install that instead. If you have the time, you can even do both, but it's best to create separate virtual machines for that.





```
martijn@ubuntu:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:4b:da:90 brd ff:ff:ff:ff:ff:ff
    inet 192.168.15.80/22 metric 100 brd 192.168.15.255 scope global dynamic enp0s3
        valid_lft 42510sec preferred_lft 42510sec
    inet6 fe80::a00:27ff:fe4b:da90/64 scope link
        valid_lft forever preferred_lft forever
martijn@ubuntu:~$
```

## The `find` command

A very helpful command during investigations is `find`. This allows you to find files and directories on a Linux system satisfying certain properties: for example, certain names, certain extensions, or files and directories that have been accessed during a particular time period.

The latter is particularly helpful: you can find files that have been modified (including files that have been created) in the past day, week or month. Issues with a website often involve files uploaded or modified by an adversary; the `find` command is the tool to look for them.

**Exercise 11.7.** Use the `find` command to find all HTML files in the web root of the nginx server set up in the previous exercise. Then, use the command to find all files modified in the past 15 days, as well as the past 5 minutes. If you want to, you can create a new file (remember the `touch` command?) and re-run the last command to confirm it works.

You can use a search engine to learn how `find` works, but you can also use the documentation built into Linux by running `man find` — a good way to familiarize yourself with the `man` command.<sup>72</sup>

## Understanding the website's context

Before starting an investigation on an incident with a website, it is important to understand a bit about the context. If there are organizations you support with their websites, even on an ad hoc basis, it would be helpful to get this information in advance to make responding to possible future incidents easier.

The first question to ask is: Who set the website up? Was it the organization (or individual) themselves, or was this outsourced to a third party? In case it was a third party, is this party still involved in managing the website, or was the setting up a one-off job and is the organization responsible for the management now?

As mentioned before, the majority of website 'hacks' are caused by outdated versions of CMSs or their plugins. It is thus good to understand if anyone was responsible for keeping these up to date and, if so, whether they did that job. It is a sad reality, in civil society organizations but also in many smaller companies, that no one has the skills, confidence or the time for this job. Finding a strategy for the long-term maintenance of a site moving forward is thus a good mitigation strategy, even if that won't solve the immediate problem you're dealing with.

Speaking to the person or organization who developed the website can be useful, particularly for a customized website that may be more fragile or error-prone. Even necessary updates like plugin security patches could

<sup>72</sup> 'Man pages' as they are called can be a little overwhelming. For a quick understanding of how to run a specific command, an Internet search is usually faster. But it's helpful to know the option exists, and `man` works even when there is no Internet connection!



break key features. The original developer could provide helpful guidance. There is more on handling this Catch-22 situation in the Mitigation section below.

Also important to know is how the website is hosted. Does it have its own dedicated server, or is it shared with other websites? And in the latter case, are these websites managed by the same entity? If not, can you access the web server logs during the investigation?

On a shared host, issues (such as DDoS, more on which below) may be related to another site on the same server or the server itself. If this is a possibility, make sure to speak to the hosting provider, if only to confirm there is no incident on their side.

It is also good to note that whether a website runs on a dedicated or a shared host, this host is likely a virtual machine. There have been cases where an issue with the hypervisor — the host of the virtual machines — led to problems on the virtual machines running on it and, thus, on the websites running on those virtual machines.

## Web server logs

Regardless of which web server you're using, the server will almost certainly generate logs. If you have access to them, they can greatly help you in your investigations.

Logs are text files updated in real-time, where each entry is a new line. You'll likely find the files in a directory under `/var/log/`, such as `/var/log/apache2/` for an Apache server and `/var/log/nginx` for a server running nginx (unless the system administrator has changed something). The precise location may be different on certain servers, though, and on shared hosts, the name of the log files might include the name of the particular website.

Logs are regularly *rotated* to ensure the files don't become too big and for privacy reasons. That means the current log file's name is changed to either add a date or a number. Older log files are also compressed using an algorithm like gzip and will get the `.gz` extension<sup>73</sup>. After some time, log files are deleted.

The precise details of log rotation and its implementation will depend on the server setup. The most important thing is that, almost certainly, older logs — maybe from as little as a week old — won't be available. If checking the logs is essential for the investigation, do so quickly and back up the log file if necessary!

If you have access to a live web server, it can be helpful to familiarize yourself with the location of the various web server logs and how they are rotated. It might sound overwhelming in theory, but it is easier to understand when you see it in practice.

## Access logs

A web server typically generates two kinds of logs: access logs and error logs.

The **access log** contains a log of all the requests made to the web server, one line at a time, with new lines added to the bottom of the file. While web servers allow you to customize what log lines look like, the default is the same for all common web servers (this makes tools that parse web server logs agnostic to the server used) and will look something like this:

```
94.64.237.82 - - [30/May/2023:13:09:34 +0100] "GET /?xxx HTTP/1.1"
200 10132 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
Gecko/20100101 Firefox/113.0"
```

---

<sup>73</sup> The tool to decompress `.gz` files on Linux is `gunzip`, but you can also view such files without decompressing them first by using `zless`. There is also `zgrep`, which works like `grep` for `.gz` files (but also for normal files). Become familiar with these commands if you don't know them already!





Here `94.64.237.82` is the IP address the connection came from. The first of the two hyphens has an archaic meaning, and you're unlikely ever to use them in practice. The second hyphen occasionally contains the name of the authenticated user making the request, for example, on servers that use basic access authentication (mentioned in a previous footnote). But most systems that use authentication, such as WordPress, don't share the authenticated user with the web server; thus, the username won't appear here.

The next entry is the date, including the time zone, in square brackets. Understanding when an accident occurred is useful, and this gives you that information to the exact second (Note: It's important to know which time zone the server is running in, and it isn't always the one you'd expect!) Running the Linux `date` command tells you the current date and time and also tells you what time zone it's in.

Next, you'll see the full request line between double quotes: the method, the URL and the HTTP version. In this case, the URL is `/?xxx`, which is a trick you may find useful if you're working on a busy server and you want to test something and find your own request in the server logs: adding something that won't affect the returned content and that's easy to search for using the Linux `grep` command will make it easier for you to find your own requests in the logs.

The next entry is the status code (200 in this case), and the one after that is the size of the response in bytes. Though you'll rarely need to know the exact size, this can help you understand if very large responses are being returned from the server, which could explain a server that runs very slowly.

The entry after that is the aforementioned referer in double quotes, which can be useful to understand where a certain visitor came from.

The final entry, again in double quotes, is the user agent.

As HTTP is a stateless protocol, unless you are in the exceptional case where there is a username in the third part of an access log, there is no way to conclusively link one request to another request by the same user, whether that request is another web page that is visited, a resource such as an image included in the original web page or the website dynamically requesting some resource from the web server<sup>74</sup>.

In practice, you can usually be quite certain that requests made from the same IP address in quick succession came from the same user.

**Exercise 11.8. (optional)** If you're unfamiliar with the Linux commands `tail` and `grep`, go back to Chapter 4. Also, ensure you understand how you can use a vertical bar `|` to 'pipe' the output of one Linux command to that of another.

On the server, find the `nginx` access log file. How would you display any new entries to the access log in real-time, but only those that contain the string `xxx`? Confirm this works by accessing the IP address in your browser followed by `?xxx`.

If you have some more time, find out the basics of regular expressions as they are used by `grep` (this is a big subject; you don't really need to know more than the basics!). How would you only show requests coming from an IP address in the range `94.64.236.0` to `94.64.237.255`?

One thing to be mindful of is that if a website uses a security service such as Cloudflare (more on which below), this can complicate matters regarding reading access logs. It can mask the original IP address to provide security and some performance benefits. You may need to hunt for signs of the real IP data when

---

<sup>74</sup> This is very common in modern websites. For example, when you have Gmail open in your browser, it constantly makes requests to the Google server to check for new email, so that you'll see new emails appear without having to manually reload the page. If you want to see this behavior yourself, check the URLs requested when you open a website in a browser.



performing your analysis to get usable results. Not everyone configures Cloudflare correctly or has changed hosting IPs before going behind it. There may be holes or historical data to leverage. You could try to check the check DNS history of the domain (using a service like VirusTotal), look for direct mail servers that may be exposing information, scan for open ports, enumerate subdomains in the hopes one is not behind Cloudflare, or dive through internet-wide scanning data, using a service such as Shodan.io.

## Error logs

The **error logs** show errors from the web server. They are usually found in the same directory as the access logs and are named and rotated in a similar way.

It is important to keep in mind that many things that are wrong with the web server aren't actual errors. For example, someone abusing a malicious WordPress plugin to upload a **webshell** is totally normal behavior from the web server's point of view. The same is true for account compromises.

For this reason, the error logs aren't usually very useful during an investigation. That said, having a look at them if you can't immediately find the cause of the problem could be helpful.

The format of error logs varies more than that of access logs, in part because not all errors are linked to HTTP requests. Because there are many different errors you may see, the best method is to look for ones that seem relevant to the issue you're looking at and then use a search engine to find more details.

In the settings for most web servers, you can change the 'log level' to specify the severity of the errors that are logged. You can set it to log warnings, notifications or even debug information. ([Here wayback machine](#) is how to do that in nginx.) By increasing the amount that is logged, you may, in some cases, get crucial information to help you pinpoint a problem.

Again, remember that the issue with a website is usually not a server issue. Moreover, you may not have access to make this change on a shared host. If you do, though, make sure you remember to revert the changes in the configuration, as the error logs can get very big quickly, causing yet another issue.

## Logs and privacy

Logs contain some personal information, such as IP addresses, which can, in some cases, be traced back to an individual, or GET requests, which may include search queries. This is one reason why logs are rotated.

While this is generally not considered a major issue, for some organizations that serve high-risk individuals, it may be. The concern here is that an adversary may access logs through hacking or by making a legal request to the organization or the hosting provider. If this is a concern, it is possible to change what is stored in the logs in both [nginx wayback machine](#) and [Apache wayback machine](#).

Do note that by changing the logging format, you may make your logs incompatible with third-party tools that process logs, for example, to provide visitors' statistics.

## Redirects

Redirects are commonly used on hacked websites to send visitors to different content, typically some kind of scam. A **redirect** sends the browser to a different page than the one requested without any user action required. There are two kinds of redirects: server-side redirects and client-side redirects.

A server-side redirect is when the server sends a 3xx status to the browser in response to a request, together with a Location header that indicates what the new URL is. Multiple redirects can follow each other. Browsers typically stop if there are too many redirects or if there is some loop; in practice, this often indicates some kind of error.



There are different kinds of 3xx statuses (Mozilla has a [helpful overview](#)<sup>wayback machine</sup>), but in practice, you needn't worry too much about their differences.

A client-side redirect is when code loaded by the browser (usually JavaScript) makes the browser load a different page. Unlike server-side redirects, client-side redirects can be configured to wait a certain amount of time before redirecting.

It is good to keep in mind that it is not uncommon for malicious redirects to contain some kind of anti-analysis behavior. For example, they may not work for users who are logged in, for people who visit the site directly rather than arrive via a search engine, or for people visiting from IP addresses that have already been redirected.

If you're trying to reproduce a redirect that you've been made aware of but that doesn't work for you, try various things such as accessing the site from a private browser session, through the Tor browser<sup>75</sup> or by clicking on a search engine result link.

Finally, sometimes a redirect doesn't happen on the main page but only on certain sub-pages. Make sure to test for that, too.

## Rewrites

Somewhat related to redirects is the concept of a (URL) **rewrite**. This is when the server 'translates' the URL into a different one on the same server without any interaction with the browser. Understanding them, or at least being aware of the existence of rewrite rules, can help you understand a website you're less familiar with.

For example, the URL of a WordPress blog post may be `https://www.example.com/2023/06/23/hello-world`, but this is locally translated to `https://www.example.com/index.php`. This PHP file uses the original URL to locate the correct page in the database and sends this to the browser. This all happens on the server side: the browser will never learn about `index.php`.

Another very common example is the default behavior of most browsers. If no file is specified in a URL, the server serves the content of `index.html` (or `index.php`, etc) to the browser.

Like server-side redirects, rewrites are configured in the server settings or in an `.htaccess` file.

## .htaccess

Redirects and various other settings are set in the configuration files of the web server (web servers typically have multiple such files). For Apache, there is another way to serve settings: by using a `.htaccess` file in a web server directory. It is common for malicious actors to modify or create `.htaccess` files to create malicious redirects.

This file contains settings that apply to the directory and any subdirectories. For reasons of efficiency<sup>76</sup>, it is always preferable to use the server's configuration files rather than a `.htaccess` file. The advantage of using `.htaccess` is that configurations can be set even if you don't have access to the server settings, which is common on shared hosting plans.

---

<sup>75</sup> Probably the fastest way to get a different IP address

<sup>76</sup> If you're curious, the official nginx [explains](#) why there is no such equivalent for that server: "You can't do this. You shouldn't. If you need `.htaccess`, you're probably doing it wrong." (That said, there are some third-party nginx plugins that add this feature to the server.)



Some CMS, like WordPress, can use `.htaccess` files on Apache servers to handle URL rewrites. When investigating, look for such files on a site running Apache and note that `.htaccess` also applies to subdirectories<sup>77</sup>, so look for the file in parent directories too.

The format of `.htaccess` is similar to that of the Apache configuration format and is [explained well](#) [wayback machine](#) by the Apache website.

**Question 11.9.** The website `https://www.example.com/` running Apache has the web root at `/var/www/html/`. You notice malicious redirects taking place for visitors on `https://www.example.com/foo/bar/`. Where should you look for `.htaccess` files that may cause this behavior? (See the appendix for the answer.)

## DDoS attacks

A **denial-of-service** (DoS) happens when some kind of service becomes unavailable due to the requests it receives, typically the quantity of them. We call these requests a distributed denial-of-service or DDoS when these requests come from many different resources.

While not every DDoS happens with malicious intent — a site may have just become very popular — they often are. The availability of relatively large botnets of compromised devices makes this a relatively cheap attack performed by governments, hacktivists (a group of hackers with an idealistic motive), or by cybercriminals. In the latter case, it could be an attempt to extort money.

Very occasionally, a DDoS attack is performed to distract the organization from another more serious attack. This is something to keep in mind if you support a very high-risk organization whose website has been under a DDoS attack.

### Is it a DDoS attack, or is the site just very popular?

If you are investigating a suspected DDoS incident, it's best first to determine if this is an attack or if something else is happening.

Assuming the organization has not been contacted by some entity who took responsibility for the DDoS, regardless of this entity's motive, and there isn't a public statement either (sometimes hacktivist groups will 'brag' about successful DDoS attacks on social media), this may require some digging.

If you have access to the web server logs<sup>78</sup>, they are your first place to go. Is the number of requests unreasonably large? What is *reasonable* really depends on the website, but you can often get a good idea by comparing the requests for each day or even each hour<sup>79</sup>: if they suddenly increased, this is a strong indicator that this is what caused the website to become unavailable.

If you do not find any increase, it is worth checking whether a recent change was made to the website. Perhaps a new plugin was added, or an existing one was updated. If this plugin is buggy and makes each request to the server take up a lot of resources, it might be that regular traffic becomes too much for the server.

---

<sup>77</sup> To be precise, subdirectories of the web root

<sup>78</sup> If not, you will probably want to contact the entity that hosts the website.

<sup>79</sup> Becoming familiar with various Linux commands can be really helpful here. For example, the following command shows the number of requests for each hour, taking into consideration all access logs in the current directory:

```
zcat *access.log* | cut -d \[ -f2 | cut -d: -f-2 | uniq -c
```



It is also worth checking if other resources on the same server may be under attack, making the site you're looking at collateral damage. If you don't have access to these sites, contact the hosting provider.

If you find an increase in traffic, you'll want to see what caused it. If you notice a certain page occurring a lot in the referer in the logs, it may be that the site just received a lot of links. (This is sometimes called 'slashdotted' after the once popular tech site.) Some of the mitigation methods described below may still work, but it might also be a temporary problem.

One typical sign of a DDoS attack is many different URLs that are being requested that often don't make sense. This could be done by the attackers because responses are often cached<sup>80</sup>, so making the same request many times isn't as effective as a DDoS strategy.

It can also be helpful to see the location from where the requests are made. Is this very different from what is normally seen by the website? This may be an indication that this was a DDoS attack.

Organizations that run their own infrastructure may also see DDoS attacks at the network layer (as opposed to the 'application layer,' which includes HTTP). Mitigating such threats is less trivial and would almost certainly involve the hosting provider and/or a DDoS mitigation service like the ones mentioned below.

Such DDoS attacks are less common for civil society, based on anecdotal data from civil society as well as [statistics from anti-DDoS service Cloudflare](#) [wayback machine](#). Cloudflare also has plenty of [documentation](#) on DDoS attacks, including the [OSI model](#), which explains the use of the term 'layer' in the previous paragraph.

## DDoS mitigation

It may be tempting to add extra resources, such as more memory or CPUs, for a website under DDoS attack, but if it was a real attack, as opposed to just more genuine requests the server can handle, that is neither the cheapest solution nor a very effective one: the adversary typically has an easier time scaling their resources than the target does.

DDoS mitigation typically involves separating human users from bots, often by using a combination of techniques. These will include looking at properties of the HTTP requests, intelligence on recent DDoS attacks (many of which come from botnets, the IP addresses of which are often tracked), or making users solve a CAPTCHA. Manually blocking certain IP addresses or certain request types is rarely, if ever, a good idea given the distributed nature of DDoS attacks; the one exception is when a misconfigured client keeps sending a lot of requests.

It is possible to set up [fail2ban](#) on the server to provide some basic protection against DDoS. While fail2ban is already installed on some servers, in most cases, it will be more effective to use a third-party service.

Two commercial solutions that are free for at-risk organizations are available: Cloudflare's [Project Galileo](#) and Google's [Project Shield](#). They can make a real difference but will require some configuration to be most effective.

Two other companies specialize in DDoS protection for at-risk organizations: [eQualtie](#), whose website protection service is called [Deflect](#), and [Qurium](#). Both services work well in their default settings and have the advantage that they are designed with at-risk communities in mind. Moreover, they could be interested in performing research on the threat to benefit the wider community.

In all cases, it will be good to understand both the short and long-term funding of the protection offered. If there is a fee to be paid for the service, there might be emergency funds through organizations like [Digital Defenders Partnership](#) that could help with that. You could also qualify for Cloudflare's Project Galileo, which gives civil society and nonprofits pro bono access to its premium-level security tools.

---

<sup>80</sup> Cached means stored in a location that requires very little resources to access.



If you think you may need any of these services in the future, it will be helpful to familiarize yourself with them, including onboarding procedures. If you can find a contact, that would be even better.

## Content management systems

A **content management system** (CMS) is a system that allows one or more people to manage the content of a website. Typically, the CMS provides users with a web interface that makes creating and editing web pages relatively easy and something that an organization can do without needing to understand HTML or PHP.

Most CMSs, including the ones listed below, are written in PHP, with the content of individual pages stored in a database, usually MySQL. Many CMSs also allow for user comments on individual pages; these, too, are stored in the database. Images are uploaded as files and stored as such on the web server.

Most CMSs allow for **plug-ins** (also spelled plugins and sometimes called **add-ons**) to be added to enhance the functionality of the website. Plug-ins typically come with their own PHP files (and sometimes other kinds of files, such as JavaScript or CSS) that are uploaded to the web server. **Themes** are used to change the way the website looks, and they also involve files (such as CSS files and images) uploaded to the web server. Both plug-ins and themes are available for free or can be bought for a fee.

A CMS typically allows for multiple users to each have their own login, and logins come with various privilege levels. Users with the most privilege are able to manage every aspect of the website, including managing other users, while users with a more basic privilege level are only able to create new posts or pages<sup>81</sup> on the website; sometimes, at a slightly more basic level, users are able to create posts, but they require an administrator to approve them.

Sometimes it is possible for website visitors to create their own account, for example, to leave comments on posts, though comment functionality doesn't always require an account to be created.

## General

Though each CMS is different, many of the security aspects that are relevant to handling incidents apply to most of them.

One obvious thing is the security of accounts with high privileges, such as site administrators or editors. Someone taking over such an account would be able to post content on the website, change the whole look of the website (sometimes called **defacement**), or redirect visitors to another site. If hidden content is stored in the database, they may have access to this as well.

The same risks apply here that apply to any account hijacking: very weak or reused passwords and the lack of multi-factor authentication. The same mitigations apply here, too, with MFA being an obvious one. If the CMS supports it, using less obvious names for the administrator users also mitigates such attacks: 'john' is better than 'admin,' and 'john4891' is even better. In most cases, the username isn't displayed anywhere on the website, and you can choose a nice display name such as 'John Smith.'

If the CMS allows for user comments, you'll likely receive some comment spam, often text or links trying to sell something. This isn't a security risk in itself, but it can be annoying. If comments are displayed automatically, they can degrade the user experience on the website. Note that it is often possible to disable comments on a per-post level or throughout the website, but spam bots still find ways to post comments sometimes.

---

<sup>81</sup> In the language of some CMSs, a 'post' is part of a blog, whereas a 'page' is a standalone page, for example an 'about us' page. It's possible to have a website that only consists of pages, or only of posts (if the website is just a blog). The distinction rarely matters; both pages and posts are stored in the database.





The main security risk with CMSs comes with vulnerabilities in the CMS itself, plug-ins, or themes.

As mentioned before, if a CMS allows for the uploading of images and these aren't properly checked, someone might be able to upload a PHP file and thus be able to manage the website and/or the web server.

Other common issues are the ability to create new users with arbitrary privileges, modify the privileges of existing users, or SQL-injection, which would allow someone to modify the contents of the database by sending very specific requests to the web server.

You don't have to understand the details of these kinds of issues to do an investigation. What matters is that vulnerabilities in CMSs, plug-ins and themes are often being found, and when that is the case, they are often exploited very quickly.

If you're dealing with an incident involving a CMS, it is important to check if its various components are up to date: you can compare the version that is installed with that of the official website or repository. Often, the CMS itself will warn when a new version of a component is available.

If a component is out of date, check to see if there are known security issues with the particular component; the official website will often tell you if there is such an issue; otherwise, a quick Internet search may tell you if an issue is known. If there is a specific component to the incident, for example, a page visitors are redirected to, a specific user that is created, a search for that may also provide you with more information on the incident, or sometimes other websites with the same issue.<sup>82</sup>

It is also possible that an administrator unknowingly installed a malicious plugin or theme, for example, a free version of a paid plugin that comes with some malware installed.

Finally, there is an unfortunate, frustrating reality when it comes to CMSs, plugins and themes. Especially on sites that are somewhat customized, things may depend on each other in a way that prevents maintainers from updating a vulnerable component. For example, an essential plugin only works with an older version of WordPress in which a vulnerability was found.

There is no simple one-size-fits-all solution for this. Some of the hardening techniques discussed below, such as installing a web application firewall, may help, but ideally, you will find a long-term solution for this. Changing to a static website, as also discussed below, may be such an option.

## WordPress

**WordPress** is the most popular CMS, and some estimates suggest that about half the world's websites run WordPress. Originally created as a tool to publish blogs, it has evolved into a powerful CMS that can be used to manage all kinds of websites.

WordPress is open source and, like any other major CMS, runs on PHP. It uses a MySQL database on the server to store content.

Installing WordPress is pretty straightforward and doesn't require much technical knowledge, yet if you do have such knowledge, the possibilities to make the site do what you want are unlimited.

The main WordPress directory contains a number of PHP files, with the most important ones being `index.php` and `wp-config.php`. The former is the main PHP file handling all the requests to the

---

<sup>82</sup> If you find many other websites that are unrelated to the one you're working on with the very same issue, you can assume this attack wasn't targeted.



WordPress instance, while the latter contains various configuration settings. Of particular importance are the credentials for accessing the database, though it is rarely necessary to connect to the database directly.<sup>83</sup>

For investigations, the most important subdirectory is `wp-content/`. This directory contains plugins, themes and file uploads, such as images used on the site. Many security issues related to the website can be found here.

**Exercise 11.10.** (*optional*) On your nginx install, add a WordPress site. There are many online guides to achieve this, and it might be a good exercise to search for a guide yourself, but if you're stuck, [this<sup>wayback</sup>machine](#) is a good guide to follow. Do note that you'll want to add `sudo` to all of the commands there (or, as a less secure alternative, run everything as root).

Be mindful that getting this to work may take a bit more time than you think because it's easy for things to go wrong, and you may end up having to tweak the settings of VirtualBox. This can be a great learning experience if you have the time, but don't persist if you feel it's not really helping you.

Installing WordPress — or any CMS — isn't something you typically need to do during the response to an incident, but if you've never worked with WordPress, this gives you some familiarity.

If you've been using VirtualBox, it might be a good idea to take a snapshot before starting the installation process. This way, you can revert to this snapshot and install another CMS if you want to become familiar with that one, too.

## Joomla, Drupal and Magento

Another popular CMS is **Joomla**. Like WordPress, it is written in PHP and can be extended through plugins and themes.

Here, too, the main file is `index.php`, with the configuration (including database credentials) stored in `configuration.php`. You won't have difficulty finding themes and plugins in directories with those respective names.

Other CMSs you may encounter are Drupal, which tends to be popular among the tech community, and Magento, a popular CMS for web stores.

As a general rule, the less popular a CMS is, the less likely it is to be exploited in large-scale attacks. However, less popular CMSs also tend to be scrutinized less for security issues and thus be more insecure, something that may be exploited by someone who wants to target the particular organization.

## Server issues

In some cases, the issue is not with the site itself but with the server that is running it. Just like other operating systems, Linux servers need to be patched regularly to fix exploitable vulnerabilities that could lead to malicious actors gaining access to the server.

**Question 11.11.** Why is a vulnerability in a (Linux) server connected to the Internet potentially more dangerous than a vulnerability in a Windows laptop or an Android phone? (See the appendix for the answer.)

---

<sup>83</sup> Should you really need to connect to the database directly, make sure you are comfortable with using the MySQL query language. You will then know how easy it is to make fatal mistakes – such as 'dropping' a table – and that it is vital that you create a backup of the database first.





Running forensics on Linux is a big and broad subject that goes beyond the scope of this guide. As with any kind of forensics, you will have to look for things that are unusual, which requires some understanding of what is usual. Thankfully, the majority of website issues aren't related to the underlying server, so this is not a scenario you're likely to experience often.

Should you need to, the first place to look is the various log files found in `/var/log/` and its subdirectories. For example, you will be able to see who logged into the server and when; this may show a new account has been created or a user logging in at an unusual time or from an unusual location, suggesting an account compromise.

One thing to be mindful of is that you'll probably see many failed attempts for users with common (English) names, such as Mike or John. Assuming basic password security is applied, such attempts are harmless, and you can generally assume the attacks are not targeted.

Beyond that, you can use third-party tools (such as the previously mentioned [MultiRBL](#)) to check for any known issues. A server's IP address appearing on many blocklists could mean it has been sending spam, which is one common way compromised servers are utilized.

Finally, in some cases, servers have been compromised because of an issue with the hypervisor they are running on that allows unauthenticated users to access the system and its servers. If this could be an issue, do check with the hosting provider.

## Cleaning up an infected website

Cleaning up an infected website is no trivial task, and what needs to be done depends on the extent of the infection, as well as on what preparatory measures have been taken. Here are some general steps to take when cleaning up a website.

### 1. Disable the website

In some cases, it may be desirable to turn off the website while you are fixing it. This is the most guaranteed way to prevent anything from happening while you're working on it. If you have access to the server software, you can just turn off the server. You can also consider using the configuration of the server to redirect all requests to a simple static HTML page with basic information on the organization and a note that the server is being worked on.

When you are fixing a CMS issue, you can also choose to turn on maintenance mode if the CMS supports that. [WordPress](#) [wayback machine](#) does, as does [Drupal](#), while [Joomla](#) [wayback machine](#) supports taking the website temporarily offline.

### 2. Fix vulnerability.

If there is some kind of vulnerability, such as a malicious plugin, this is the time to fix it. You can update the plugin or remove it altogether. Make sure you check other potential issues, too; there may be other outdated plugins and themes or other issues with the website that didn't cause the incident this time but could be exploited in the future.

### 3. Restore from backup or remove malicious content

If a recent backup was taken of the website, you can restore the site's content from that backup. Ensure you check how recent the backup was and check for any new genuine content that may have been added after the backup was created. You'll probably want to consult the organization to ensure the content is as up-to-date as possible when restoring the site.



If no backup is available or if the backups somehow aren't usable, you will need to clean up the website manually. In this case, it might be helpful to create a full site backup before doing so in case you make a mistake.

The previously discussed `find` command will be helpful to discover which files have been created or modified through the compromise. Depending on the nature of the file, you will want to remove them or restore them to a previous version. If it concerns a plugin, it might be better to reinstall it rather than try to fix it.

Don't forget to check the content in the database, which you can almost certainly manage through the CMS. Look for created or modified posts and pages, but also for newly created users and other modified settings.

#### 4. Re-enable website and verify it works

If you had previously disabled the website, re-enable and check everything. Also, have people from the organization check as well to make sure everything works fine for them.

#### 5. Harden the website

If you have the time to do so, this would be an excellent moment to harden the website to prevent future hacks. If anything, the organization will never be more receptive than now to hear feedback on the importance of this. Hardening is discussed in the next section.

#### 6. Keep checking

You can never be 100% certain you have truly fixed all the issues. Especially in the period after an incident, regularly check the website for issues – or ask the organization to do so. This will be a relatively easy process as you will know what to look for.

### Hardening websites

Many website issues are, to some extent, preventable or at least mitigatable. Not just by performing good security practices, such as keeping CMSs and their components up to date, but also by hardening the website and server in various ways. Also included in this section are various ways to make responding to incidents easier in the future.

One kind of hardening can be done at the CMS level by using multi-factor authentication (assuming it is available) for all users, or at least those with administrative rights. If that's not possible, using less guessable usernames, as discussed above, will make a great difference.

Cleaning up the plugins and themes of a CMS may also help. Remove those that aren't actively used and evaluate how essential those that are used are.

### Web application firewalls

A **web application firewall (WAF)** is a special kind of firewall that monitors incoming HTTP traffic and blocks malicious traffic. While a WAF is never 100% perfect, it can make a huge difference, especially against known attacks, such as those exploiting outdated plugins or themes. Almost all attacks against websites are known.

A special kind of WAF is installed as a CMS plugin. In the case of WordPress, there are free plugins by security companies [Sucuri](#) and [Wordfence](#). Both plugins include the ability to check for file integrity to help you notice changes that could indicate a site has been hacked. Sucuri also published some useful [advice](#) [wayback machine](#) on hardening a WordPress site.



## Security Headers

Security headers, as mentioned before, can make a website more secure by mitigating various kinds of attacks. They are particularly relevant for sites that may be the target of phishing attacks against users or administrators. Make sure to understand the implications of a particular security header before enabling it.

There are many guides on how to enable them; [this wayback machine](#) explains how to set up many common security headers in both nginx and Apache. A WAF may also include the option to enable certain security headers and may even have them enabled by default.

## Added logging

During your investigation, you may have wished certain logs were available. If possible, take the time to add them. What is being logged at the server level can be set in the server settings.

CMSs typically don't create logs themselves, but it is often possible to do so using plugins. Don't hesitate to install such a plugin if you find it useful, but keep in mind that this plugin also needs to be kept up to date!

## Backups

Backups are essential during site recovery, not just from attacks but also from mistakes or errors. There are various plugins that can create backups in the CMS, either automatically or manually. In the latter case, make sure that these are indeed created.

Many hosting providers also offer options to create and store backups, often for a small fee. The advantage here is that backups are typically stored elsewhere so that even in a total compromise of the server, the backup won't be affected.

Two things should be kept in mind when it comes to creating backups. The first is that, like logs, backups are often rotated, which means they are only stored for a certain period. If this period is shorter than the time it takes to discover a website compromise, you'll end up in a situation where all backups are compromised, too. Keep in mind that some compromises can be quite subtle, especially if they apply some tricks to prevent the site owner from learning about them.

A second thing to keep in mind is that if a site contains sensitive information, this information will be stored in the backup, too, which may be a concern if it is stored elsewhere. That said, this is likely an issue for a minority of websites, as storing sensitive information in a web server generally isn't a great idea.

**Question 11.12.** What would you take into consideration when deciding whether a site would need to be backed up daily, with backups being stored for a week, or weekly, with backups being stored for a month? (See the appendix for the answer.)

## Static website

A final way to really harden a website is to turn a website that uses a CMS into a static website, where there is no dynamic content, which almost always means PHP files. This seriously reduces the attack surface.

The way this works is usually by keeping a version of the website with the CMS on a server that isn't accessible to the public Internet<sup>84</sup> and then using a plugin that creates a static copy of the website, which is then pushed to the live web server. There are many such plugins available for various CMSs.

---

<sup>84</sup> This can be done on a server that runs locally, in a virtual machine on someone's computer, or by using a second server in the cloud with strong access control. In any case, one should consider that the dynamic version is easily accessible by the organization itself!



While a static website can certainly make a huge difference, you should consider whether this really works for the organization. They should be comfortable using the plugin. They should also be aware that there will be a small delay before new content gets posted; this option is more suitable for those organizations that don't publish fresh content several times a day.

Having a static website is also a great option for a website that isn't actively maintained anymore but that the organization still wants to keep online.

Finally, make sure that pushing new content live, which involves copying data to the web server, doesn't create a weakness itself, for example, because a weak password or protocol is used to copy content to the server.



## Chapter 12: iOS incident response

**iOS** is Apple's mobile operating system that powers Apple's iPhone devices. It is the second most used mobile operating system, after Android, and (at least in 2023) the only other one you're likely to encounter. A very closely related operating system, **iPadOS**, runs iPads. They are so closely related that if you ever need to perform forensics on an iPad, this guide should suffice for that, too.

Under the hood, iOS is a Unix-like operating system (also referred to as \*nix), which means that should you end up looking at log files, you'll notice the same file structure you may be familiar with from Linux or Android. There are also many similarities with macOS, the operating system for Apple's Mac computers, including many programs available on both.

This guide is meant to help you become familiar with iOS in a way that you can perform some basic forensics on iPhones and iPads to check their security and confirm infection or, just as importantly, confirm a device is likely clean.

Unlike Android, for which really cheap devices exist, iPhones are expensive. It is also not generally possible to create a virtual environment for iOS to "play around." It is thus possible that you don't have an iOS device. Because of this unfortunate reality, some exercises in this module may not be possible. Luckily, that is not a huge problem, as it is possible to read through the module and understand how iOS works, even if you don't have a device at hand.

### iOS versions

Just like there are different iPhone versions, there are different releases of iOS. New versions of each get released about once a year, while iOS gets security updates many times throughout the year, too, as and when updates are needed.

A newer iOS release typically runs on earlier iPhones, going back several versions, and you can install security updates on iOS without having to release to the latest version.

For example, at the time of writing (August 2023), the latest iOS version is version 16, released in September 2022. It can run on iPhones as old as the iPhone 8, first sold in 2017. The latest version of iOS 16 is 16.5.1 — this is the only version of iOS 16 that has the latest security updates installed. However, the latest version of iOS 15, which currently is iOS 15.7.7, should also be considered secure.

In 2023, Apple started releasing 'Rapid Security Responses' that apply important security fixes between updates. These are, by default, installed automatically. They are denoted in the software number by a letter in parenthesis, for example, iOS 16.5.1 (c).

Apple keeps an overview of all the security updates, while the Wikipedia page on iOS version history is also helpful to check if you want to understand if an iPhone is up to date. The question here is not if the latest version of iOS is installed but if the latest version within a specific release is.

However, even though the latest version of earlier iOS releases should be secure against all known exploits, there are two things to keep in mind. The first is that sometimes, Apple includes security features that mitigate



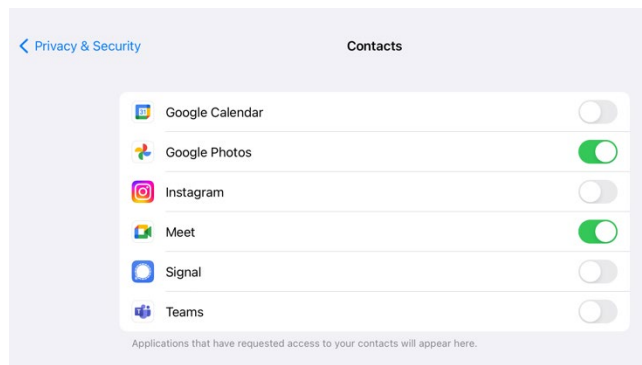
threats that only work from a specific iOS release. For example, in 2022, Apple introduced Lockdown Mode to mitigate against advanced spyware, which is available from iOS 16.

Secondly, it has happened in the past (and may happen in the future) that a vulnerability is found in an older iOS version that is impossible to fix in that version. For those reasons, running the latest iOS version is generally the most secure thing to do.

## Apps and app permissions

Just like on Android, everything on an iPhone happens inside an app. Apps are isolated from each other and don't have access to things like the camera, the location, or files on the device unless they've explicitly been given this permission.

Compared to Android, the permissions are less granular: apps have either been granted permission or have not. You can check the various permissions in the Settings under Privacy & Security. For each kind of permission, you see which apps have requested and which apps have been granted the permission.



*Six apps have requested access to the contacts on this iPad; only two of them have been granted this access.*

One useful feature of recent versions of iOS (iOS 15.2 and later) is the 'App Privacy Report' that you also find under Privacy & Security. This is disabled by default, but when enabled, it keeps track of all access to various sensors as well as network activity by the various apps.

This can be really helpful when trying to find continuous unexplained activity, for example, a location being shared, photos being taken, or domains being contacted. You can enable this feature, have the device be used normally for some time, and then check the App Privacy Report.

Do note that a lot of data ends up in the App Privacy Report: every domain contacted by any app running on the device, including browsers<sup>85</sup>. You can get a scary amount of information on how someone uses their device by looking at the report.

If you want to use this feature during an investigation, it is strongly recommended that you have the user enable it themselves and have them look through the report with your guidance without you actually seeing their activity.

---

<sup>85</sup> The one exception, according to Apple, is network data from browsers running in private mode.

**Exercise 12.1.** (optional) If you have access to an iPhone (or iPad) with a recent version of iOS, turn on the App Privacy Report and use a few apps: browse a few websites, attempt to take a photo or upload a picture to social media. Confirm all this activity in the App Privacy Report.

## The App Store

A few exceptional cases aside, the only way to install apps on iOS is by downloading them from Apple's **App Store**. You can think of this as the equivalent of Google Play, from which users can download Android apps, but there are two notable differences.

The first is that on Android, it is possible to download apps from other locations, such as app stores belonging to a device manufacturer like Samsung or Huawei, third-party app stores, or even manually installing the APK file. The difficulty of this can vary, but a user can manually install an app that isn't checked by Google, and that may be malicious. This is exploited by some developers of malware, including some less advanced spyware, to socially engineer someone to install the spyware manually, obviously without them realizing what they are doing.

That is different on iOS, where such 'side-loading' isn't possible. Hence, every app installed on an iPhone has been vetted by Apple to some extent and must follow the rules set by Apple.

The second difference is that malicious apps slipping through Apple's defenses are rare and anecdotally less common than the same thing happening on Google Play. And those apps that slip through tend to be apps engaged in various kinds of fraud rather than apps that steal sensitive data<sup>86</sup>.

Falling for a scam isn't great and still a concern, of course, but this isn't what users are most concerned about, especially at-risk users.

For these two reasons, malware other than advanced spyware is extremely rare on iOS. Years of media stories of hacking and spyware have led to a commonly held belief that it is easy to hack into phones. This isn't true in general and is even less true for iPhones. Keep this in mind when you're investigating a possible incident on an iPhone.

## Third-party app stores in the future

Apple keeping its ecosystem tightly secured definitely has security benefits. It also has downsides, including the fact that Apple has the power to decide what kinds of apps are allowed on iPhones. Sometimes, Apple denies apps that benefit at-risk people from being published on the App Store. One example is VPN apps that aren't made available for iOS users in China [wayback machine](#).

Though not necessarily out of concern for at-risk users, legislators in the EU have taken notice of this, and in its 2022 Digital Markets Act [wayback machine](#) (DMA), they have demanded that Apple open up its ecosystem for third-party app stores. When this will happen is not clear yet, nor what form it will take; Cory Doctorow wrote a good analysis [wayback machine](#) of the situation for the Electronic Frontier Foundation.

---

<sup>86</sup> It should be noted that the same is true for 'malicious' Android apps found on Google Play. The permission model for both iOS and Android means that an app that wants to steal data needs to somehow be open about it, and that makes it a lot harder to stay under the radar.





What matters in this context is that by the time you are reading this, the App Store may not be the only way people can install apps on an iPhone, at least in some parts of the world. One cannot predict what consequences this will have, but it is likely that at least some at-risk users will deliberately choose to install apps from third-party app stores: these app stores may have cheaper versions of the original apps. It may also be true that users will be socially engineered to install questionable or even malicious apps.

Whatever will happen in practice, what was said above about the permission model will still apply. It just means that you will have to study individual apps that use sensitive permissions a bit more carefully and be especially mindful of apps not installed from the App Store<sup>87</sup>.

## Apple ID and iCloud

To set up an iPhone, one needs an **Apple ID** account that is tied to an email address. Any email address can be used for this, not just addresses on Apple services. The account is also used for other Apple services, including **iCloud**, Apple's online storage service.

When investigating a possible incident on an iPhone, it is important to consider the Apple ID. This gives access to iCloud backups, many of which are generated automatically. Someone with access to the Apple ID password and the ability to confirm 2FA codes has access to the backups.

What is backed up to iCloud depends on the permissions given to individual apps and whether the app supports iCloud backup. For example, currently, both WhatsApp and Telegram support it; Signal doesn't.

On the AppleID website or on any device the user is logged into (go to the user's profile at the very top of the Settings), you can check what other devices the account is also logged into. An unknown device listed here is a big red flag, but don't forget to check if it could be a very old device: devices the user never logged out of are included here, too, even if they've been turned off for years!

Also, consider how likely it is that someone managed to get access to that account if two-factor authentication is enabled. It would likely have involved some kind of physical access to a device, possibly combined with some pressure, and in that case, the adversary could have just read all the data from the device directly. That said, they could maintain longer-term access by logging the account into a different Apple device they control or by using some service that promises to read iCloud backups<sup>88</sup>.

## Mobile Device Management

Sometimes, larger organizations want to keep some kind of control of the devices their staff uses (both organization-provided devices and personal devices used for work), and they use a service called **Mobile Device Management** (MDM). This allows them to enforce certain security controls, such as a password policy. On iOS, one of the things an MDM can do is manage apps. These could be regular apps from the App Store, but also special apps the organization has written themselves. You can read more about MDMs on iOS on [Apple's website](#).

There is a scenario where a malicious actor registers as an MDM, then uses social engineering to convince targets to enroll in their MDM and then uses this to distribute malware. It is not a particularly likely scenario,

---

<sup>87</sup> Is it likely that then, as on Android, you will be able to find the source of the app from the settings.

<sup>88</sup> These services probably aren't as reliable as is claimed, but it may be a possibility for some adversaries

given the steps involved and the fact that should Apple discover it, they can immediately terminate the campaign.

It has been [used](#) [wayback machine](#) by malicious actors in the past, though, so it's worth quickly checking if the device is enrolled in an MDM. This can be done in the Settings, under General and then VPN & Device Management. Note that some NGOs, especially larger ones, may use MDMs, and the user may not be aware of this: if this is their personal device, they may simply have been told to install something.

## Jailbreaking

Another way to install apps from outside the App Store is to 'break out' of the restrictions imposed by Apple; this is called **jailbreaking**. Unlike on Android, where a user can change these restrictions, jailbreaking isn't possible on iOS without exploiting some kind of vulnerability.

The scenario where this is done by a malicious actor, possibly using a zero-day exploit, is discussed later in the section on advanced spyware, but sometimes people decide to jailbreak their own devices. This is, for example, done by more technically inclined people who like 'hacking' their own devices, but in some parts of the world, it is also common to jailbreak iOS to bypass certain restrictions, such as the aforementioned ban on VPNs in China.

Make sure you are familiar with how likely jailbreaks are in the country or region you work in, and also ask the person whose phone you are investigating if it could have been jailbroken (maybe they didn't do so themselves, but someone else gave the phone to them). Do note that it isn't a very common scenario.

A jailbreak of this kind is easily ruled out by checking if the iOS version is up to date; Apple patches vulnerabilities that could lead to jailbreaks very quickly. In all known cases, updating iOS undoes the jailbreaking and effectively removes any apps installed this way. You can also use the [iVerify](#) app, which can be installed for a small one-off fee, provides information on security updates, and detects most jailbreaks.

## Safety Check

Apple devices allow various kinds of sharing between family members, friends, and colleagues. This is great for convenience and for collaboration, but it also comes with risks: relationships can turn hostile, or the device of a trusted contact ends up in the hands of an adversary.

For this reason, sharing is generally not recommended for at-risk people, but it is still fairly common. Thankfully, Apple has created a very helpful [Personal Safety User Guide](#). This guide includes a number of checklists that allow you to quickly stop sharing data with a person or device you don't trust anymore. The guide is also available as a [PDF](#).

## Advanced spyware

Ever since the **Pegasus** spyware was [first discovered in 2016](#) [wayback machine](#), Pegasus has been the main example of advanced spyware targeting iPhones. Technically very sophisticated, many of its targets have been journalists and civil society members.

It isn't the only one, though, and there are several other kinds of advanced spyware for iPhones, with names such as [Predator](#) [wayback machine](#) and [Quadream](#) [wayback machine](#). Together, I'll refer to these as **advanced spyware**.



Advanced spyware tends to use **zero-day vulnerabilities** (vulnerabilities that have not been disclosed publicly and therefore not yet patched by Apple; see Chapter 4) and often **zero-click exploits**, where there is no user interaction required. Even when there is user interaction required, this rarely involves more than a single click, which makes infection far more likely than the many clicks required by more basic spyware<sup>89</sup>.

Advanced spyware is really hard to deal with but also, thankfully, pretty rare. It isn't sold on the open market and not even on the cybercriminal underground. Those who can buy it, usually governments and militaries, often pay tens of thousands of dollars per target. This ballpark figure can help you rule out many people as potential targets of advanced spyware.

Thankfully, there are two things specific to iPhones that help with the detection. The first is that iPhones keep many traces of past activity, which allows researchers to discover infections that happened a long time ago, even if the phone has been rebooted many times since. This is especially helpful because advanced spyware tends not to be able to maintain persistence: it doesn't stay active on the phone following a reboot, and regular (for example, daily) rebooting is very sensible advice given to potential targets.

The second thing that helps us is the Apple ecosystem, which allows Apple to get some insight into infection attempts by advanced spyware. Since 2022, Apple has been sending notifications on email and iMessage of infection attempts it has seen, though it should be noted that these notifications aren't sent in real-time and often refer to infection attempts going back months or even years.

Android malware was covered in Chapter 8. The previous points should make it clear that advanced spyware on Apple is currently easier to detect than it is on Android, which doesn't necessarily mean it is more prevalent. Advanced spyware for Android is mostly a big blind spot<sup>90</sup>.

## Coordinating with Analysis Experts

The fact that Apple often sends out notifications a long time after the infection attempts is an indicator of how hard it is to detect advanced spyware. The 'unknown' aspect of zero-day vulnerabilities means that it is impossible to predict how future advanced spyware will infect iPhones and, thus, where to look. The developers of spyware also tend to spend a lot of effort hiding their activities, not just from the users but also from researchers.

There are three organizations that have specialized in detecting and analyzing such spyware, in particular as they target civil society: **Access Now**, **Amnesty Tech** and **Citizen Lab**. They actively work with civil society organizations around the world, and it is okay to decide trying to do any analysis yourself simply isn't worth your time.

In that case, especially if you believe people whom you support may be targets of such spyware, it will be helpful to proactively reach out to one or more of these groups and discuss what you can do in case spyware is suspected on the iPhone of someone you are supporting.

---

<sup>89</sup> Such basic malware is actually extremely rare on iPhone, because it would require tricking a user into bypassing Apple's built-in protections. Bypassing this, even if you try hard, isn't something that Apple lets you do. This is why more basic malware is far more common on Android, where security protections can be bypassed by the user.

<sup>90</sup> One thing that makes the iPhone a more attractive target for adversaries though, is the uniformity of the iPhone ecosystem. A single exploit works against all of iOS, or at least a specific version of iOS. Someone trying to target Android will likely need different exploits to target Samsung, Huawei, Pixel, etc.



Even if you decide to perform some analysis on your own, it is likely that you will need the help of these organizations at some point: they are more likely to know where to look for the latest spyware and have indicators that aren't (yet) publicly known. Much as open research is great and beneficial for the community, in this case, the adversaries are reading this research too and will adapt their techniques as soon as they find out how their spyware is being detected.

## Triaging

Whether you do some work on your own or coordinate the analysis with third parties, there is a lot of work involved. Therefore, an important first step is to triage and determine whether the person is a likely target.

The ballpark price of each use of advanced spyware mentioned above shows that most people will never be targeted: the likely adversary is likely not willing or able to pay that amount of money, or they may not even have access to such spyware<sup>91</sup>. It is okay not to pursue a thorough investigation if the person simply isn't likely a target of spyware.

It is also worth noting that many people believe they are targeted by advanced spyware because their phone misbehaves: for example, it crashes a lot, or the battery runs out quickly. And while spyware suspicions should be taken seriously, there are many far more plausible innocent explanations for a misbehaving phone, including the fact that a phone may be quite old.

**Question 12.2.** Give one reason why spyware causing a phone to misbehave might not be in the interest of the actor deploying the spyware. Also, give one reason why it might be in their interest. (See the appendix for the answer.)

Many people who are potential targets of advanced spyware have been traumatized and that often makes them hypervigilant. This is very understandable. An understanding of this trauma and an ability to support them is often far more important than having the technical skills to look for spyware. It is also a lot harder.

Sometimes, even if you yourself are certain a phone isn't infected, performing an analysis can be helpful to give someone peace of mind. But before spending significant time and effort on this, it is worth asking yourself: is a negative result going to make them feel better, or will it simply reinforce their belief that they have been targeted by 'even more advanced spyware'?

## Trust

The risks people who are targets of advanced spyware face, as well as the trauma mentioned above, means that trust is even more important than usual. It can help to give the person you're supporting as much control of the situation as possible.

Show them what you are doing and why you are doing it, and explain which other organizations you are involving and what reputations they have. If a backup needs to be made, ensure it is encrypted with a password only they know, and have them enter it directly when the process asks for it. Make sure they remember it, though!

---

<sup>91</sup> People being targeted with spyware by (ex-)partners is a real thing, commonly referred to as **stalkerware**. But unless these people have high positions in government or military (or corrupt connections there), they simply don't have access to advanced spyware, as it isn't sold to individuals.



Also, explain what the backups do and do not contain.

One thing that can be helpful is to ask them to bring an empty external hard drive to store the backup. This gives them even more control of the situation and has the added benefit of not clogging up your own systems with backups.

Finally, keep in mind that trust is even more important than detecting spyware. Allow people to stop the process at any time if they have doubts or don't feel safe, and communicate this clearly.

## MVT

The most popular tool to analyze iPhones for traces of advanced spyware is the [Mobile Verification Toolkit](#), or **MVT**, developed by Amnesty Tech.

Before using it, it is best to understand the possibilities and limitations of MVT. It is an open-source tool that checks iPhones (and Android phones) against a list of indicators (IoCs; see Chapter 7). By default, these indicators are public (though it is possible to add private lists, should you have access to them), which means you'll only be able to look for known spyware, traces of which can be found in known places.

Because it is open source, spyware authors have access to MVT, too, and it is likely they are making some efforts to hide their spyware from places where MVT looks. That said, the tool is constantly being developed, and the lists of indicators are constantly being updated.

MVT can be installed on both macOS and Linux, including REMnux, but it isn't pre-installed in any of them. To install, simply follow the [instructions](#) on MVT's website. Installing new software on Linux through the command line can be a bit fiddly because a specific set-up may break something in the installation process, but as of writing, the instructions work in both REMnux and in a recent installation of Ubuntu 22.04 LTS.

The MVT guide presents jailbreaking the iPhone as an option. Aside from the fact that you can't 'just' jailbreak an iPhone (as you will have read earlier in this chapter), this is strongly recommended against unless you really know what you are doing<sup>92</sup> and you don't need to hand back the device later on<sup>93</sup>. That said, it will give you extra data that can be very useful to analyze an infection.

Installing the [libimobiledevice library](#) is strongly recommended. Again, it is confirmed the instructions work on both REMnux and Ubuntu 22.04 LTS. You can run `ideviceinfo` to confirm it works, though, at this stage, it will probably tell you that no device is connected.

If you run MVT on a virtual machine, you will need to make sure the iPhone, which you connect to the host machine via a USB cable, is visible to the virtual machine. To do this in VirtualBox, you first need to allow "this device [that is, the computer] to access photos and videos" by clicking 'Allow' on the phone.

Then, in the Settings of the virtual machine, choose 'USB' and then the small icon with a '+'.' Add the iPhone that appears in the menu.

---

<sup>92</sup> To the point where you wouldn't need this chapter in the first place.

<sup>93</sup> In some cases, the person has obtained a replacement phone. This is rare, because of the obvious costs involved (though maybe you've found a generous funder), but otherwise an ideal situation as it allows for longer term analysis. Do note that trust is even more important here as you have access to very sensitive data.

To connect the iPhone, go to the Devices menu of the virtual machine in VirtualBox and in the USB menu, ensure the iPhone has a tick box in front of it. If not, click on it. You will be asked to click 'Trust' on the iPhone and will likely have to enter the phone's passcode.

Note that connecting a device through USB to a virtual machine can be a bit frustrating. Sometimes, restarting the virtual machine, reconnecting the phone, and trying again solves an issue you have.

You can confirm a device is connected by running `ideviceinfo` again. It should give information on a device.

Now, *hopefully*, you are ready. However, changes made to both Linux and iPhones might mean that the version of the `libimobiledevice` you installed isn't recent enough<sup>94</sup>. In that case, as the MVT guide recommends, you can install the latest version from GitHub, but this is only recommended if you are already comfortable with installing software from GitHub. Otherwise, there is a decent chance you will end up in a rabbit hole of trying to fix one small error after the other.

Now you are ready to follow the instructions to [create an encrypted backup](#) and then [check that backup](#) by running `mvt-ios` on it. It will likely prompt you to download the default indicators, though if you have them, the command also lets you provide your own indicators.

The commands allow you to choose a location for the backups, and this is where you can choose an external hard drive, as recommended above. Make sure the hard drive is connected to the virtual machine.

Several records are created by this process (though more would have been created if you had chosen a full filesystem backup), and the guide [explains](#) what they are. For signs of infections (to be precise, matches against the indicators), look for JSON files with 'detected' in the name.

**Exercise 12.3. (optional)** Install MVT on a virtual machine and connect an iPhone to it. Check the iPhone for advanced spyware as described above.

## iTunes backups

An alternative to creating a backup through the command line is to use iTunes. This creates the same backup format and, as in the command line option, lets you set a password for encryption, which is strongly recommended. It is more user-friendly than using the command line, though you'd still need the command line to run `mvt-ios`.

An advantage to using an iTunes backup is that you can ask the phone user to do this themselves, which can be helpful if it is not possible to do the analysis in person. You can then ask for the backup to be sent. Note that the backup consists of a directory with many files inside, and you will want to use `zip` to turn them into a single file.

This process is far from ideal, as this means you will have access to the decrypted data. If you absolutely have to do this, make sure the backup is sent on a different channel than the password<sup>95</sup>. Moreover, run the analysis in a virtual machine of which you have taken a snapshot before the process and return to the

---

<sup>94</sup> When writing this guide, it worked smoothly on Ubuntu 22.04 LTS, but didn't on REMnux

<sup>95</sup> Signal or WhatsApp with disappearing messages set to a short period of time are ideal for this purpose



snapshot once you are done (if `mvt-ios` detects an infection, copy the detection files off the virtual machine before doing so!).

Whatever backup format you choose, you will probably want to store the backup somewhere (or even better, as suggested above, have the user store the backup themselves). This can help you check the backup at a later stage, as new indicators are discovered, and the phone may have been factory reset<sup>96</sup>. Of course, it won't be able to find any infections that happened after the backup was generated. Make sure the user doesn't forget their password, as an encrypted backup is useless without it!

## **iMazing and iVerify**

There are two other tools that can be helpful in detecting advanced malware.

The first is **iMazing**, mobile device management software that runs on Windows or macOS and allows you to manage an iPhone connected through USB. As of 2021, it includes checks for advanced spyware, using the same methodology and indicators as MVT (which is free and open source and thus can be used in other software).

iMazing gives you less control than MVT does, and you may not trust its developers as much as those of MVT, who work directly with civil society organizations. However, for a quick check, it can be a very helpful tool and is something that someone to whom you cannot offer in-person support can run themselves with relatively little guidance.

In that case, in particular, note that at least the limitations of MVT apply to iMazing, too and be mindful of a false sense of security.

Then there is also the aforementioned **iVerify**, which checks against known indicators of advanced spyware. Here, it is very important to realize that iVerify is installed as an app and thus is limited to what an app can do. It cannot dig deeply into the operating system to look for indicators but may have access to various messaging apps, which are sometimes used as the infection vector of advanced spyware, especially when it doesn't use a zero-click exploit.

This is a severe limitation, and I would be wary of using iVerify to detect advanced spyware. However, the app has many other benefits, including warning when new updates for the iPhone are available, and it is worth installing it for that reason alone. That it may report on some traces of advanced spyware is a nice bonus.

---

<sup>96</sup> Unlike a reboot, a factory reset removes traces of advanced spyware from a phone





## Appendix: Answers to questions

### Chapter 4

**Question 4.3.** The output `ls -l` shows the contents of the current directory, excluding hidden files and directories, with more details for each entry; `ls -a -l` gives the same output as `ls -al`.

**Question 4.4.** Directories have a line starting with `d`.

**Question 4.5.** The size of the file in bytes.

**Question 4.11.** Because it contains everything on the file system: every file and directory accessible on the device.

**Question 4.12.** This command is `head`.

**Question 4.14.** In that case, the errors of `tail` are 'written' to `/dev/null`.

**Question 4.18.** You will have to run `grep "\.html" bigdata.txt`, using the backslash to 'escape' the dot, which has a special meaning of a wildcard, and the quotes are so that the escape is done in the string, not in the command itself (you don't need to understand this latter bit).

**Question 4.19.** You can get this by running `du / 2> /dev null | grep html`.

### Chapter 6

**Question 6.3.** If you unzip the file in a directory also accessible by the host, an antivirus product may detect it and report it. Depending on your work environment, you may have to have an awkward conversation with your IT department. For some context, antivirus typically scans any new files, and unzipping a file creates a new file on the device.

### Chapter 7

**Question 7.1.** There is no risk of accidentally clicking it. And security products won't block 'bad' file names, whatever that would mean.

**Question 7.3.** This is the empty string. A hash can be calculated of every piece of data, including no data at all! Keep this in mind, as sometimes a file you're analyzing happens to be empty (because something went wrong somewhere). You aren't expected to remember the hashes, of course; just remember that a search engine is a very useful tool in threat analysis.



**Question 7.4.** While one can never be certain, this suggests the file was used in multiple different campaigns, which in turn would suggest the campaign wasn't particularly targeted.

It is also possible that an actor, for example, a mercenary spyware vendor, is using the same file under different names in multiple campaigns, but all three names are somewhat generic and don't suggest a very targeted campaign.

**Question 7.5.** Here are some reasons:

- This domain is relatively new (2021, whereas the other domains are from 2005 or earlier)
- There is an unusual top-level domain (.bid). You don't have to know all TLDs, but you probably don't come across legitimate .bid domains very often. That is a red flag.
- The number of detections for this domain – at the time of writing – is 12, whereas the other domains are detected by 0 or 1 products.
- The domain name itself 'feels' fake. This is hard to quantify, but when you do more threat research, you'll come across a lot of domains like this that try a little too hard to look legitimate.

**Question 7.6.**

1. No, in the 'relations' tab, it shows the file connects to some IP addresses directly.
2. In the 'details' tab, you see the permissions requested by this file. Permission to access the microphone isn't mentioned.
3. In the 'community' tab, someone has linked a report by security company SentinelOne.

**Question 7.7.** There are now (2 December 2022) six files contacting this domain! If you look at the other domains contacted by these files (all legitimate domains), you notice the same domains (and often the same URLs) keep reappearing. This suggests that the six files are related.

**Question 7.8.** If you look around a bit, you'll find that `145.239.231[.]82` was one of the IP addresses this domain has pointed to. Two other domains that have pointed to this IP address are `update-driversonline[.]club` and `updater-driversonline[.]club`. From the names of the domains, it seems pretty clear they are related.

This is a very common situation in threat intelligence: you cannot 100% prove that they are related; it would just be too much of a coincidence for them not to be. That is usually enough evidence.

**Question 7.9.** A data breach of the organization or of a third party may have led to a list of names and email addresses being leaked and thus made available to the actor behind this non-targeted campaign. Another possibility is that someone at the organization or at a third party has malware on a device that stole their contact details, which include names and email addresses.

**Question 7.10.** The entity sending the commands, which might be a law enforcement agency, a security researcher or a group of such entities, would be interacting with other people's devices without their permission. Also, the malware might not be completely understood – after all, the entity isn't the one who wrote it – and what they do may have unintended consequences.



## Chapter 8

**Question 8.1.** Malware is always a possibility, but there are many other options that are probably more likely. It might be a compromised account (for example, Google, with location access turned on), but it may also be someone who just happened to see her. It might also be that she posted something on social media that showed she was at the bar or that her presence was shared (semi-)publicly in some other way, for example, through a guest list or a photograph that was taken.

## Chapter 9

**Question 9.1.** Even with STARTTLS used, SMTP connections are encrypted between mail servers and between email clients and mail servers. You could call this 'point-to-point' encryption, which is different from end-to-end encryption in which the email is encrypted all the way between the mail clients, and the mail servers aren't able to see the emails. Note that it is possible to encrypt the content of emails using something like PGP, but some headers and the SMTP transaction data still need to be available to all the mail servers handling the emails.

**Question 9.2.** To use SMTP to send an email from a mail server to a client, the client would need to be listening to connections from the Internet. This is often not possible without making changes to the firewall, and many users don't have access to their firewalls (think of mobile devices).

**Question 9.7.** The decoded text is:

Liberté  
Égalité  
Fraternité

**Question 9.8.** The decoded text is:

This text contains only ASCII characters. There was no reason to use base64 to encode it. If you see this in practice, it is always suspicious: sometimes, the text is base64 encoded to evade spam filters or security products. However, because base64 encoding isn't encryption, it isn't hard for such scans to look inside base64 encoded text.

**Question 9.9.** You never know whether the website you are submitting content to is storing the data somehow and, if it does, who it may be shared with.

**Question 9.11.** A DKIM signature is added by the mail server handling the outgoing email. Because Internews uses Microsoft for its email, they add the DKIM signature. (Note that it would be possible for Internews to manage its own DKIM keys and upload the private key to Microsoft's servers – for this reason, the question wasn't great and probably won't be included in future versions of the training.)

**Question 9.12.** Each new mail server that handles the email adds a Received header (and possibly more header). This means that if one server signs *all* headers, the signature will not be valid for all headers as soon as the mail is sent to another mail server. This is why a DKIM header explicitly lists all the signed headers.



**Question 9.14.** Here are some possibilities:

- Check the whois data for the domain (for example, by running the whois command on Linux)
- Find a public statement from Facebook, such as this one, that confirms the domain is used by Facebook.
- Check the IP address that `facebookmail[.]com` points to (`dig facebookmail[.]com`) and then check the PTR record for that IP address (using `dig -x`) to confirm this is a `facebook.com` IP address. You could also look at the email itself and may notice a Received header confirming the email was received from a `facebook[.]com` server.

## Chapter 10

**Question 10.1.** Using SMTP to the server `cp5ua.hyperhost[.]ua` with username `arnoldlog@steuler-kch[.]org` and password `7213575aceACE@#$.`

### Malware Config

**Extracted**

<b>Family</b>	agenttesla
<b>Credentials</b>	<b>Protocol:</b> smtp
	<b>Host:</b> cp5ua.hyperhost.ua
	<b>Port:</b> 587
	<b>Username:</b> arnoldlog@steuler-kch.org
	<b>Password:</b> 7213575aceACE@#\$
	<b>Email To:</b> arnold@steuler-kch.org

**Question 10.7.** A doesn't depend on a remote server which may have been shut down at the time the malware was opened.



## Chapter 11

**Question 11.3.** Logging in as root means that every command you run is executed with root permissions. If you make a mistake, the consequences can be much bigger.

**Question 11.9.** They could be in `/var/www/html/foo/bar`, `/var/www/html/foo/`, or `/var/www/html/`.

**Question 11.1.** Such a Linux server is receiving incoming traffic from the Internet that can be used to exploit the vulnerability. An Android device, on the other hand, while also being based on Linux, cannot be contacted directly from the Internet. (This may be a bit confusing. After all, an Android device does receive messages — such as app updates — from the Internet, but these are initiated from the device.)

## Chapter 12

**Question 12.2.** A misbehaving phone could alert its user that something is wrong, which in turn could lead to the phone being analyzed for spyware and the spyware being detected. At the same time, sometimes the primary goal of spyware isn't information theft, but intimidation of the target and misbehavior could be part of that intimidation.

