PRIVACY GUIDE

GrapheneOS in depth

Complete guide

This guide explains how GrapheneOS works, the mobile operating system focused on privacy and security: what it hardens compared with Android, what it genuinely protects and what lies outside its reach.

From hardened Android to hardware security

Contents

INTRODUCTION AND BASICS

· How to use this guide 3

· The basic concepts 3

CHAPTERS

01 What GrapheneOS is and the problem it solves 5

02 Supported devices: why only certain phones 6

03 Verified boot and the security chip 7

04 App isolation: SELinux and seccomp 8

05 Memory: hardened_malloc, MTE and kernel hardening 9

06 Less attack surface: what is switched off by default 11

07 Improved permissions: Network, Sensors, Storage Scopes and Contact Scopes 13

08 User profiles and separate spaces 14

09 Unlocking, automatic reboot and duress PIN 16

10 Google Play in a box: Sandboxed Google Play 17

11 The browser and the built-in apps 18

12 Installation: from the browser and with the bootloader locked 19

13 Updates 20

14 What GrapheneOS does not protect 21

15 Comparison with stock Android and with iOS 22

16 Good practices 23

17 Sources 23

INTRODUCTION

How to use this guide

Audience and prerequisites. You do not need to know anything about operating systems or computer security. The "Before you start" section introduces the concepts used later: what an operating system is, what Android is, how permissions work, what storage encryption is, what "root" means and what the bootloader is. From there the guide climbs in difficulty step by step.

Project status. GrapheneOS changes often: new releases arrive every month and the list of devices grows over time. This guide describes the current situation, with official support for several Pixel models and an announced partnership with Motorola for future devices. If a detail may have changed, check the project's official website.

How to read it. If you are starting from zero, begin with "Before you start". Chapters 1 to 3 explain the big picture, the devices and the boot process. Chapters 4 to 9 describe the system's technical defenses. Chapters 10 to 13 cover apps, installation and updates. Chapters 14 to 16 explain the limits, the comparison with other systems and good practices.

BEFORE YOU START

The basic concepts

To understand what GrapheneOS adds you need six ideas: what an operating system does, what Android is, how apps are isolated, how a phone is encrypted, what root access is and what role the bootloader plays.

The operating system

The **operating system** is the program that controls the phone: it hands out memory, manages the screen, the network and the camera, and decides what each app may do. Everything else rests on it. If the operating system has a security flaw, no app can fully protect you, just as an armored door is not much use in a house with cardboard walls.

Android and AOSP

Android is the operating system of most of the world's phones. Its foundation is published as open source under the name **AOSP** (Android Open Source Project). Each manufacturer starts from that foundation and adds its own: apps, services and, in Google's case, the **Google Play services** (the store, notifications, location and many other functions that run with high privileges). GrapheneOS also starts from AOSP, but without Google's services, and hardens it from the inside.

Apps and permissions

Every Android app runs in a **sandbox**: a space of its own, isolated from the others, with its own files. If it wants to step outside that box for something specific, such as reading your contacts, using the camera or reaching the internet, it has to ask for a **permission**, and the system grants or denies it. Permissions are how you decide what each app knows about you.

Storage encryption

Everything a modern phone stores (photos, messages, passwords) is **encrypted**: it is turned into an unreadable jumble that can only be undone with a key. The key comes from your PIN or password combined with a hardware component inside the phone itself. There is one important detail: after the phone is switched on, the data stays encrypted until you unlock it for the first time. After that, even when the screen locks, part of the key stays in memory so the phone can keep working. That is why a phone that is switched off, or freshly restarted and not yet unlocked, is much harder to attack than one that has been on for days.

Root

Root (or "superuser") is an access level that can do anything on the system, skipping the isolation rules. Some users root their phones to customize them, but that opens a door wide: any app or flaw that obtains that level controls the whole phone. GrapheneOS does not include root and advises against it, because it weakens exactly the barriers the system is trying to reinforce.

The bootloader

The **bootloader** is the first program that runs when you switch the phone on. Its job is to load the operating system. A **locked** bootloader only starts software signed by a trusted key and checks that it has not been modified; an **unlocked** one starts anything, which lets you install another system but leaves the phone without that check. Installing GrapheneOS requires unlocking the bootloader for a moment, and at the end it is locked again, this time with the GrapheneOS key.

CHAPTER 1

What GrapheneOS is and the problem it solves

GrapheneOS is an open-source mobile operating system, maintained by a non-profit project funded by donations. It began in 2014 and is based on AOSP. It is not an app or a patch: it replaces the phone's entire operating system with another one that runs the same Android apps but adds many defenses.

The goal is twofold. On one side, **security**: making it as hard as possible for a flaw in an app, a malicious file or an attacker holding the phone to succeed. On the other, **privacy**: making sure the system does not send data to third parties on its own and that you control what each app can see.

The threat model

No tool protects against everything. Before deciding whether GrapheneOS suits you, it helps to know what it defends against.

THREAT	WHAT GRAPHENEOS DOES
Malicious or nosy app	Isolates each app more strongly and lets you trim permissions that standard Android does not (network, sensors, contacts, files)
Security flaw in the browser or in a file you receive	Makes it harder for the flaw to turn into control of the phone, with memory defenses and a stricter sandbox
Stolen, lost or confiscated phone	Reduces what can be extracted while the phone is locked: closed USB port, automatic reboot, long passwords, duress PIN
Tracking by system services	Includes no Google services and limits what the system itself sends
Well-resourced attackers with unknown exploits	Raises the cost of the attack, but cannot guarantee immunity

The table shows the approach: GrapheneOS does not promise invulnerability, but to **raise the price** of attacking the phone and to **trim what apps know about you**. What is left out (the carrier's network, what you yourself tell an app, the manipulation of people) is covered in chapter 14.

CHAPTER 2

Supported devices: why only certain phones

GrapheneOS cannot be installed on just any phone. The project officially supports only specific models, because its security depends on the hardware: a hardened operating system on a chip without the right protections would, in the project's own words, be impossible to make reasonably secure.

Current support. Official support covers 21 devices, from the Pixel 6 to the Pixel 10a, including the Pixel 10, 10 Pro, 10 Pro XL and 10 Pro Fold. The project recommends the Pixel 8 and later, which receive at least seven years of security updates from launch. One example: the Pixel 10 and 10 Pro have manufacturer-guaranteed support until August 2032.

What GrapheneOS requires from the hardware

The project publishes the requirements a device must meet to be worth supporting. The main ones:

- **Tagged memory** (MTE or equivalent) and control-flow protection (BTI and PAC or equivalents), explained in chapter 5.
- **Verified boot** with protection against old versions, for both the firmware and the system.
- **Hardware key storage** (StrongBox), key attestation and PIN-attempt limiting through a separate secure chip.
- **Isolated components**: the modem, GPU and others must not be able to read main memory without control.
- **Updates** for at least five years (current Pixels reach seven).
- **USB port** whose data can be disabled in hardware and with no debugging access while the phone is locked.

In practice, only Pixels met all of this when the project took shape, which is why until now they have been the only official support.

Beyond Pixel: the partnership with Motorola

The project has long said that extending support to any phone would go against its goals, and that it prefers to work with manufacturers that build devices meeting its requirements. That collaboration took concrete form in March 2026, when Motorola announced a long-term

partnership with the GrapheneOS foundation for "future devices designed to be compatible with GrapheneOS". As far as has been communicated, there is no confirmed model or launch date. It has also been said that manufacturers that do not meet the requirements, such as Fairphone, are ruled out.

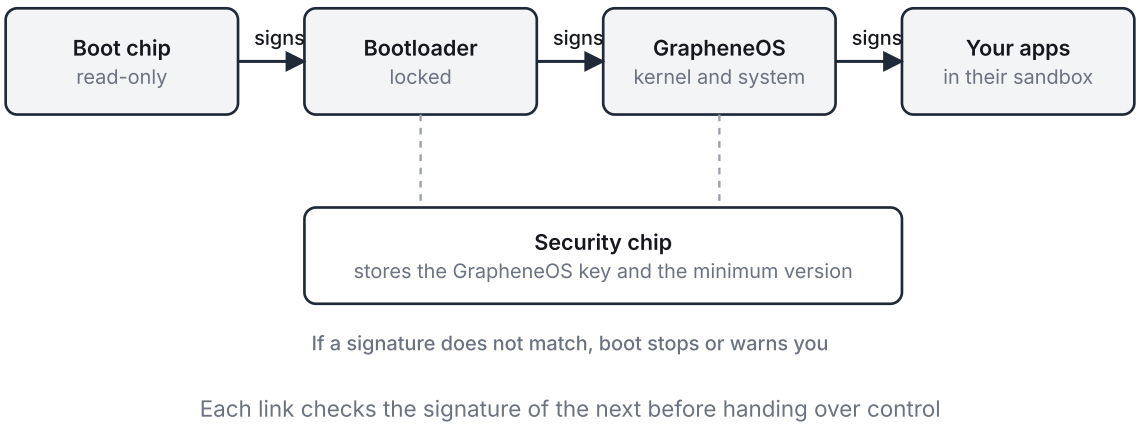
CHAPTER 3

Verified boot and the security chip

The first line of defense happens before the system starts. The aim is to make sure that what runs on the phone is exactly what GrapheneOS published, without a single modification.

The chain of trust

Think of a relay race. When you switch the phone on, a first program burned into the chip, which cannot be modified, checks the signature of the next one. That one checks the next, and so on up to the operating system. If someone has altered a piece, the signature does not match and the boot stops or warns you. This is called **verified boot**.



Each link checks the signature of the next before handing over control

For it to work, the final check needs a trusted key. With the bootloader locked, the Pixel accepts the GrapheneOS key, which is stored in the phone's **security chip** (a separate processor designed to hold secrets and resist physical attacks). Every time it starts, the phone displays a fingerprint of that key on screen. A yellow warning at boot means "system verified with a key that is not the factory one", and it is normal on GrapheneOS.

Protection against old versions

An attacker might try to install an old, vulnerable version of the system. That is why the chip remembers the most recent version number and rejects any earlier one (*rollback protection*). GrapheneOS extends that check to updates of system components, which are also verified when installed and when booting.

The security chip and PIN attempts

The security chip does another key job: it **limits attempts to guess your PIN**. The key that encrypts the phone is derived from your PIN and from a secret that lives inside the chip. If someone extracts the phone's memory and tries millions of PINs by brute force, the calculation needs the chip, and the chip throttles the attempts and spaces them further and further apart. That is why a long password, which an attacker would have to try for centuries even without throttling, adds protection to a barrier that is already hard.

CHAPTER 4

App isolation: SELinux and seccomp

On Android every app lives in its own box. GrapheneOS reinforces those boxes with two mechanisms from the system's kernel.

SELinux: a rulebook for every process

SELinux is a system of mandatory rules. Every program on the phone has a label, and a rulebook states precisely which labels can touch which resources. It works like the access control of a building with key cards: even if you get into one floor, your card does not open the others. What matters is that these rules **cannot be changed by the program**, not even if it manages to run its own code inside the app. GrapheneOS tightens those rules compared with standard Android and applies them to its own components as well.

seccomp: a list of allowed calls

When an app needs something from the system (opening a file, creating a connection), it makes a *system call*. **seccomp** is a filter that trims that menu: the app can only use the calls on a short list, and if it tries one that is not there, the system stops it. That way, a flaw in the app has fewer kernel tools within reach to escape its box.

The browser, the most exposed part

The browser processes content that comes from the internet with nobody checking it, which is why it is the attackers' favorite target. GrapheneOS improves the isolation of the processes that draw web pages (the *renderer*), so that a flaw while interpreting a site stays locked in and does not reach the rest of the phone. Chapter 11 looks at the Vanadium browser.

CHAPTER 5

Memory: hardened_malloc, MTE and kernel hardening

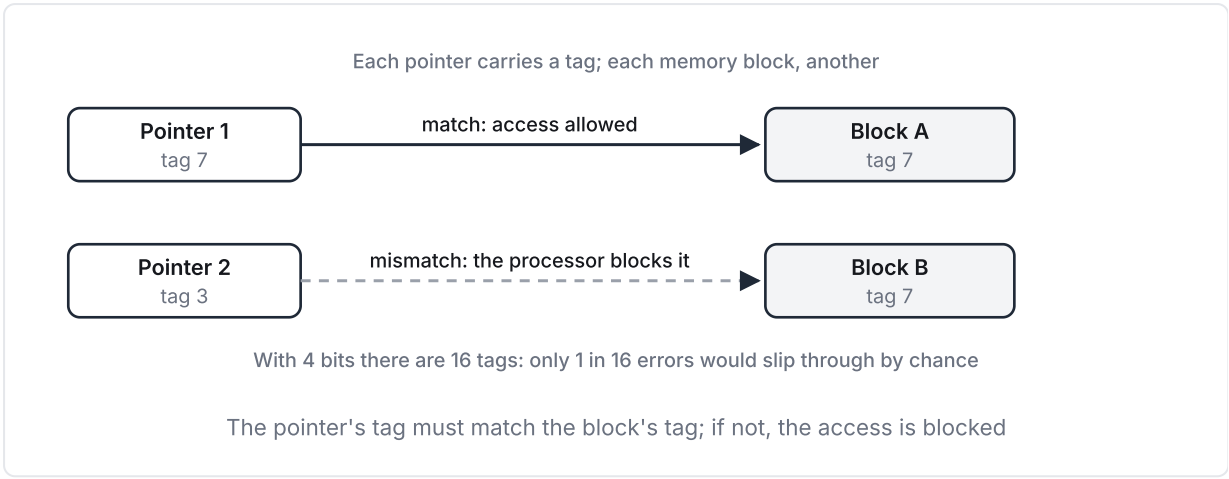
Most serious attacks against phones exploit **memory** errors: a program reads or writes where it should not and an attacker turns that into control. They are the most common flaws in code written in C and C++, and GrapheneOS puts a lot of effort into making them harmless.

hardened_malloc: a distrustful memory manager

When a program asks for memory, a component called the *allocator* hands it out. Android's stock allocator is fast, but trusting. **hardened_malloc** is GrapheneOS's own allocator, designed to make errors harder to exploit: it keeps its control data far from the program's data, wipes each block when it is freed, delays the reuse of freed blocks, leaves guard gaps between regions and adds random markers. If a program writes past its limit, it is far more likely to hit a trap and be stopped by the system than for the attacker to gain anything.

MTE: memory tags

Memory tagging (*Memory Tagging Extension*, MTE) is a processor feature, available on the most recent Pixels. It works like colored keys and locks: each chunk of memory gets a color (a tag) and every pointer that points to it carries the same color. When the program tries to access it, the processor compares the two colors. If they do not match, a memory error has occurred and the access is blocked on the spot. GrapheneOS uses MTE in hardened_malloc allocations of up to 128 KB, in the kernel allocators and in the Vanadium browser.



Key formula. MTE's protection can be measured. With a 4-bit tag there are 16 possible colors:

$$P_{\text{detection}} = 1 - \frac{1}{2^4} = \frac{15}{16} \approx 93.75\,\%$$

In words: out of every sixteen erroneous memory accesses, the processor stops fifteen; only one slips through unnoticed because, by pure chance, the wrong color happens to match the right one.

Breaking down the formula:

- **4 tag bits.** Each chunk of memory carries a 4-bit tag, that is, a number between 0 and 15. Four bits is what fits in the unused bits of a 64-bit pointer.
- **2^4 = 16 colors.** This is the number of distinct tags. If the attacker does not know which one is correct, they pick one at random out of 16.
- **1/16.** This is the probability of guessing the color by chance. It is the only way an error goes unnoticed.
- **1 – 1/16 = 15/16.** The complement: the probability that the error is detected.

An example with numbers. An attacker tries to read a chunk of memory that is not theirs. The memory has color 7 and the attacker's pointer carries 3. The colors do not match and the processor blocks the access. To get through, the attacker would need the pointer to carry 7 as well, something they do not control because tags are assigned at random. If they tried a hundred times, the system would most likely stop them within the first few.

Why it works: what decides the outcome is not what the attacker does, but a coincidence of colors that they can neither see nor choose. In addition, every tag mismatch is logged, and in practice the program closes at the first error, so an attacker rarely gets the second chance they would need.

Kernel hardening

The **kernel** is the central part of the operating system, and whoever controls it controls the phone. GrapheneOS adds several barriers to it:

- **Wiping of freed memory**: pages and blocks that are no longer in use are filled with zeros, so no leftovers remain.
- **More unpredictable addresses**: it uses a 48-bit address space with 33 bits of randomness (ASLR).
- **Signed modules**: only kernel modules signed with a key unique to each build are loaded.
- **Lockdown mode**: in forced-confidentiality mode, not even high-privilege code can read kernel memory.
- **ARMv9 hardware protections**: BTI and PAC, which prevent jumping into the middle of a function or forging return addresses, and in the kernel also CFI and ShadowCallStack.

On the randomness of addresses, the figure is this:

> $$2^{33} \approx 8.6 \times 10^{9}$$

> **In words:** the system can place programs in about eight and a half billion different positions, so an attacker who does not know where each piece is has to guess among an enormous number.

CHAPTER 6

Less attack surface: what is switched off by default

The simplest idea in security is this: what is switched off cannot be attacked. GrapheneOS reduces the **attack surface** by disabling dangerous features or trimming them to the essentials.

Compilation and dynamic code

Many attacks consist of injecting new code into a running program. GrapheneOS makes this harder in two ways. It replaces the just-in-time (*JIT*) compilation of Android's runtime with ahead-of-time compilation, and it blocks dynamic code loading in nearly all system processes.

For the apps you install there are three switches: dynamic code loading from memory, from storage, and WebView JIT.

Radios off

NFC, Bluetooth and UWB connectivity comes disabled by default. Radios are an entry route, and most people do not use all of them at once. In addition, Wi-Fi and Bluetooth scanning for calculating location is off by default.

The USB-C port

The charging port is also a data port, and it is a classic route for extracting information from a locked phone. GrapheneOS offers five modes:

MODE	WHAT IT DOES
Off	No data and no charging
Charging only	Never accepts data
Charging only when locked	When locked it only charges
Charging only when locked, except before the first unlock	Intermediate variant
On	Works as usual

The default is **charging only when locked**: as soon as the screen locks, the system cuts new data connections both in the hardware and in the kernel, and disables the port's alternate modes, such as video output. It is a stronger protection than Android's standard switch.

Debugging and traces

Native code debugging (*ptrace*) is blocked for system apps and can be blocked for yours. Screenshots strip the version and time data that other systems leave inside the image.

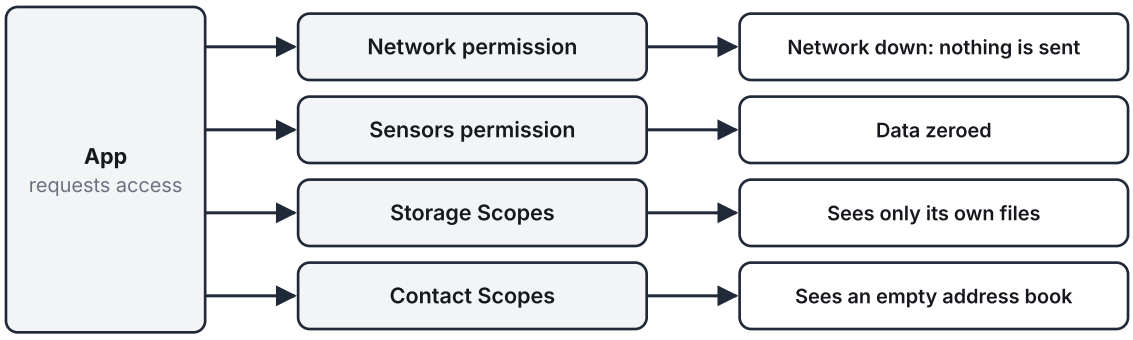
CHAPTER 7

Improved permissions: Network, Sensors, Storage Scopes and Contact Scopes

Standard Android lets every app ask for things and only lets you refuse some of them. GrapheneOS adds new permissions to trim what Android takes for granted.

Network permission

Android grants internet access by default to every app that declares it, and does not let you take it away. On GrapheneOS there is a **network permission** that can be removed, even at install time. If you take it from an app, the network simply appears down to it, so it does not show revoked-permission errors or crash. A notes app, a flashlight or a calculator does not need the network, and without it it cannot send data to anyone.



The permissions added by GrapheneOS trim what Android takes for granted

Sensors permission

The accelerometer, gyroscope or barometer do not ask for permission on Android, and they are used to identify you or to spy on what you type. On GrapheneOS a **sensors** permission blocks them and the app receives zeroed data. You can leave it blocked by default for the apps you install.

Storage Scopes: files made to measure

Normally, an app that asks for access to "files" sees all of the phone's storage. With **Storage Scopes**, the app believes it has the permission but only sees the files it created itself and the ones you add by hand. It is like lending a friend a drawer instead of giving them the key to the house.

Contact Scopes: an address book with a filter

With **Contact Scopes**, an app that asks for contacts receives an address book that looks empty, apart from the contacts or groups you choose to share. It also blocks all writes to the address book. Many apps ask for the full address book only to suggest friends or to upload it to their servers, and this way that upload gets nothing.

Secure clipboard

The **Secure Paste** feature prevents one app from reading what another has copied, with per-app and global switches. It stops a background app from seeing a password you have just copied.

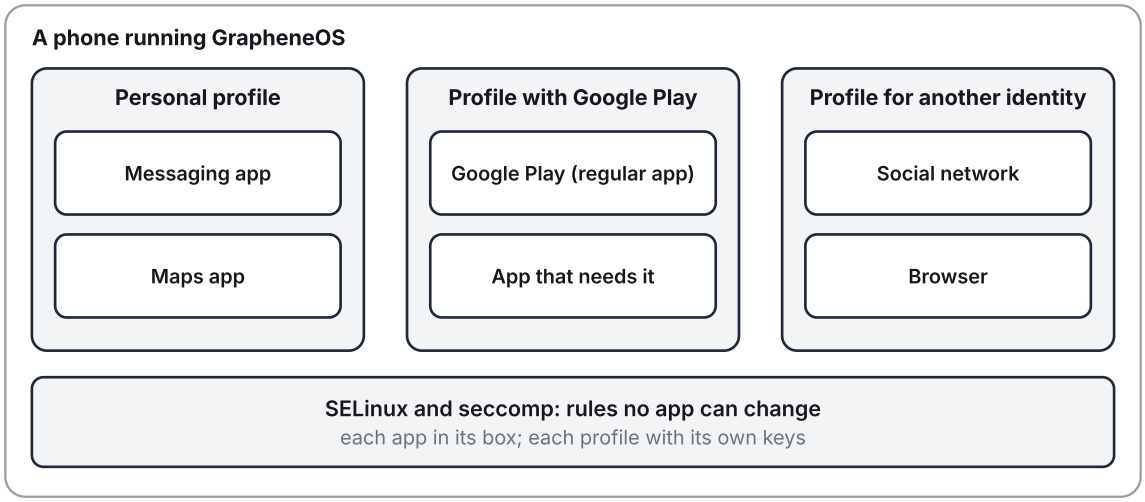
CHAPTER 8

User profiles and separate spaces

A single phone can hold several separate digital identities, and GrapheneOS takes this further than standard Android.

Profiles

A **profile** is like a phone inside the phone: it has its own apps, its own data, its own encryption and, if you want, its own VPN configuration. What is in one profile cannot be seen from another. GrapheneOS allows up to 32 profiles (31 plus a guest one), compared with the 4 of standard Android. Some typical uses:



Each profile is an isolated space, with its own apps and its own keys

- A personal profile and another for work.
- A profile where Google Play is installed, for the few apps that need it, and a clean one for everything else.
- A profile for a different identity, for example a social media account you do not want linked to your name.

When you finish using a profile you can tap **End session**: the system closes it and erases its keys from memory and from the hardware, so its data is encrypted again. You can also prevent apps from being installed in secondary profiles.

Why separating helps

Apps can infer the phone model and detect GrapheneOS, and they can also recognize a specific profile. If you want two identities not to be linked to each other, putting them in different profiles is the most reliable way. Android's *Private Space*, which hides apps behind a second lock, can be used for the same organizational purpose, but user profiles separate more: they have encryption keys of their own.

CHAPTER 9

Unlocking, automatic reboot and duress PIN

The locked phone is the first barrier if it is stolen or confiscated. These features reinforce it.

Longer passwords and fingerprint with a second factor

Standard Android limits the password to 16 characters. GrapheneOS allows up to **128**. Fingerprint unlock is limited to five attempts in total, and a second factor can be enabled: after recognizing the fingerprint, the system also asks for the PIN. If you prefer, you can disable the fingerprint for unlocking and use it only to confirm actions inside apps.

The strength of a password depends on how many combinations there are:

$$N = c^{L}$$

In words: the number of possible passwords grows explosively with length: each extra character multiplies the total by the number of characters available.

With a 6-digit PIN there are 10^{6} = one million combinations. With a 12-character password chosen from about 60 options there are around 2×10^{21} combinations. The difference between the two is fifteen orders of magnitude.

Scrambled keypad

PIN scrambling rearranges the digits on the keypad every time you open it. It helps against anyone looking over your shoulder and against fingerprints on the screen, which would reveal which keys you press. It works on the lock screen and on the SIM card PIN.

Automatic reboot

As explained in the basic concepts, a freshly restarted phone is much better protected than one that has been on for days. **Automatic reboot** restarts the phone if it has been locked for a while without anyone unlocking it: 18 hours by default, configurable between 10 minutes and 72 hours. Unlocking any profile resets the count. The mechanism lives in the system's boot code, so a system fault cannot sidestep it.

Duress PIN

GrapheneOS has a **duress PIN and password**. If someone forces you to unlock the phone and you enter that credential, the system irreversibly wipes the device and the eSIMs, without rebooting. You need to configure both the PIN and the password, and it is enabled in the security settings of the Owner profile. If the duress credential matched the real one, the real one prevails. It must be used with care: it is not a way to "pretend" to unlock, but a permanent wipe.

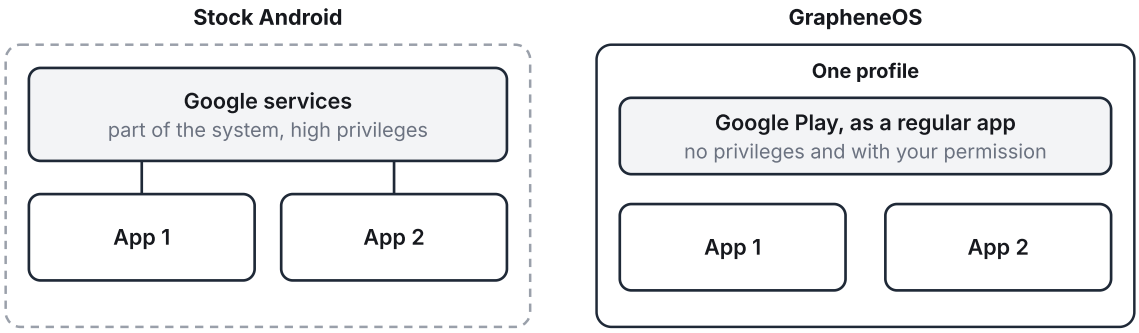
CHAPTER 10

Google Play in a box: Sandboxed Google Play

Many apps, from banks to maps, expect to find Google Play services. GrapheneOS does not include them, but it offers a way to use them without giving up isolation.

An app like any other

With **Sandboxed Google Play**, Google's services are installed as a normal app, **with no special privileges**, inside a profile's sandbox. A compatibility layer makes apps believe they are on an Android with Google. On standard Android those services are part of the system and can see almost everything; here, instead, they need your permission like any app, and only the apps in the profile where they are installed can use them.



On GrapheneOS, Google stays inside a box and only that profile's apps use it

Google Play as an unprivileged app, inside a profile's sandbox

What works and what gets rerouted

The store, in-app purchases and resource downloads work. Location requests go, by default, to a GrapheneOS implementation of its own, not to Google; you can change this if you prefer Google's service. Android Auto also works with reduced privileges. To install apps from the store you need a Google account.

Trade-offs

An app that refuses to work without certain Google checks may fail on GrapheneOS, because the system does not pass the integrity checks that Google reserves for certified manufacturers. Some banking and payment apps work; others do not. It is worth checking before relying on one.

CHAPTER 11

The browser and the built-in apps

GrapheneOS ships a minimal set of apps of its own, designed to reduce what is sent outward.

Vanadium, the browser

Vanadium is a hardened version of Chromium, and it is also the engine other apps use to display web pages. Among its protections: tagged memory, strict isolation of each site, JavaScript JIT disabled by default (with a per-site switch), third-party cookies disabled, WebGPU disabled and ad and tracker blocklists. It does not support extensions, and the project advises against adding them because they increase the attack surface and the digital fingerprint.

Other apps

- **Camera and PDF viewer**: apps of its own, isolated and hardened.
- **Auditor**: checks, with the help of another phone, that the system and the firmware are the original ones, through hardware attestation.
- **Seedvault**: encrypted backups.
- **App store of its own**, with a mirror of Google Play.

Network and services

GrapheneOS's servers are used to check connectivity, network time and browser updates. Their logs are deleted after a few days. DNS encryption protects the queries, but it does not hide which sites you visit, because the server name still travels in the clear inside the connection. To hide your IP you need a VPN or Tor.

CHAPTER 12

Installation: from the browser and with the bootloader locked

Installing GrapheneOS is a guided process done from a computer, and it erases all of the phone's contents, so copy first anything you want to keep.

What you need

- A phone with official support, preferably not sold by a carrier: carrier versions may have the bootloader locked in a way that cannot be opened.
- A computer running Windows, macOS, Linux or ChromeOS, a good-quality USB-C cable connected directly to the computer, with no hubs.
- A compatible browser, such as Chromium, Chrome, Edge, Vanadium or Brave with its protections disabled. Avoid private browsing and versions packaged as Snap or Flatpak.

The process, step by step

The web installer uses **WebUSB**, a technology that lets a page talk to the phone over the cable from the browser itself, with no programs to install.

1. **Enable OEM unlocking** in the phone's developer options.
2. **Enter fastboot mode**: switch the phone off and turn it on while holding the volume-down button.
3. **Unlock the bootloader** from the page and confirm it on the phone. This erases all data.
4. **Download and flash** the GrapheneOS release. The installer flashes the firmware and the system automatically; meanwhile you do not need to touch the phone.

5. **Lock the bootloader again** and confirm it. This erases the data once more and enables full verified boot with the GrapheneOS key.
6. **First boot**: OEM unlocking is disabled by default. It must stay that way.

Verifying the installation

At every boot, the phone shows a fingerprint of the key with which the system was verified. On the Pixel 6 and later it appears in full and you can compare it with the official published fingerprints. For a stronger check, the **Auditor** app uses hardware attestation, and needs a second Android device or an optional tracking service.

If you ever want to return to the original system, there is a flashing tool from Google: you first unlock the bootloader, erase the GrapheneOS key and install the factory system.

CHAPTER 13

Updates

Most security flaws are fixed with patches, so what really protects you is **receiving them on time**.

How it updates

The GrapheneOS updater looks for new releases every few hours, downloads them in the background and installs them onto a secondary copy of the system. A reboot is all it takes to activate them. If the first boot after the update fails, the system returns to the previous version by itself. Updates are signed, cannot be downgraded to old versions and the server is treated as untrusted. You can choose the allowed networks, the minimum battery level and whether the phone must be charging.

Android and kernel patches

GrapheneOS incorporates the monthly Android patches and recent versions of the long-term-support Linux kernel, and is usually ahead of the stock system. In addition, the project reports flaws to Google and detects patches that were left unapplied.

How long support lasts

The manufacturer's support sets the phone's useful life: five years on older phones and seven on the Pixel 8 and later. An unpatched phone is not safe, however good the system it runs.

CHAPTER 14

What GrapheneOS does not protect

This chapter is as important as the previous ones: it exists so you do not expect from GrapheneOS what it cannot give.

LIMIT	WHY
Mobile carrier tracking	The network locates you through cell towers. Only airplane mode cuts the radios completely
Closed hardware	The modem and part of the firmware belong to the manufacturers. GrapheneOS isolates them, but cannot review them
Installation identity	Apps can detect GrapheneOS, the model and the profile. For separate identities you need separate profiles
Apps that track you	If you use a social network, it knows what you do on it. The system only limits what else it sees
Encrypted DNS and visited sites	Encrypted DNS does not hide the sites you visit; to hide your IP you need a VPN or Tor
Apps that intercept traffic	Blockers that require installing their own certificate weaken the encryption of web pages
Phone theft	It does not include Google's factory reset protection
Manipulation of people	A deception that convinces you to give away a password or install something is not stopped by any technical defense

Manipulation, scam messages and phishing explain most account takeovers, and no operating system fully prevents them.

CHAPTER 15

Comparison with stock Android and with iOS

Each system has a different approach and it is worth looking at them without taking sides.

ASPECT	STOCK ANDROID (PIXEL)	GRAPHENEOS	IOS
Code	Partly open, partly closed	Open	Closed
Google services	Integrated with high privileges	Not included; optional in a box	Not applicable
Permission control	Android's standard ones	The standard ones plus network, sensors, files and contacts	The system's own permissions
Memory defenses	Android's own	Stronger: hardened_malloc, MTE in more components	The manufacturer's own
App compatibility	Maximum	High, with exceptions for apps that require Google certification	Maximum within its ecosystem
Hardware	Google's	Pixel, with Motorola announced	The manufacturer's, integrated

Each row has its nuances. iOS is a very carefully made system, but closed: it cannot be independently reviewed from the inside. Stock Android has the greatest compatibility, with more third-party services integrated. GrapheneOS bets on user control and on hardening, in exchange for a short list of phones and some apps that do not work.

CHAPTER 16

Good practices

The best system is weakened if it is used badly. These habits get the most out of GrapheneOS.

1. **Use a long password**, not just a six-digit PIN, and enable the scrambled keypad.
2. **Leave automatic reboot enabled** and change it to a short time if you are worried about theft or confiscation.
3. **Keep the USB-C port in the default mode** (charging only when locked), or a stricter one.
4. **Reduce permissions**: remove the network from apps that do not need it, use Contact Scopes and Storage Scopes, and block sensors by default.
5. **Separate with profiles**: one profile for Google Play and a clean one, and a different profile for each identity you do not want linked.
6. **Install few apps**, from trustworthy sources, and uninstall the ones you do not use.
7. **Always update** and do not disable the updater.
8. **Use Vanadium** and do not add extensions or browsers with other engines.
9. **Combine with a VPN or Tor** if you need to hide your IP: the system protects the phone, not your connection.
10. **Turn off the radios** you do not use and enable airplane mode when you do not want to be located.

CHAPTER 17

Sources

- GrapheneOS: features, FAQ and devices, web install and usage guide.
- Partnership with Motorola (March 2026): Android Authority.
- Collaboration with a manufacturer and device requirements: PiunikaWeb.