

Decision

of the Court of First Instance of the Unified Patent Court
Local Division Mannheim
delivered on 12 June 2026

CLAIMANT

Irdeto B.V., Taurusavenue 105 - 2132 - LS Hoofddorp – NL

Represented by Holger Stratmann

DEFENDANTS

1) **DJI Europe B.V.**, Bijdorp-oost 6 - 2992 - Barendrecht – NL

2) **DJI GmbH**, Industriestraße 12 - 97618 – Niederlauer – DE

3) **Solelectric GmbH**, Ubstadter Straße 23 - 76698 - Ubstadt-Weiher – DE

Defendants represented by Tobias J. Hessel

PATENT AT ISSUE

EP 2 831 787 B1

PANEL/DIVISION:

Panel of the Local Division in Mannheim

DECIDING JUDGES:

This decision is delivered by the legally qualified judge Böttcher acting as deputy of the presiding judge, judge-rapporteur Johansson, legally qualified judge Sender and technically qualified judge Jauregui Urbahn.

LANGUAGE OF THE PROCEEDINGS

English

SUBJECT-MATTER

Infringement action and Counterclaim for revocation

DATE OF ORAL HEARING

21 April 2026

1. SUMMARY OF FACTS

1. This case concerns alleged infringements of EP 2 831 787 B1 (hereafter referred to as EP'787, the patent-in-suit or the Patent).
2. Irdeto B.V. (hereafter referred to as Irdeto) is a software security and media technology company, based in the Netherlands, with customers in more than 20 countries. Irdeto delivers inter alia solutions for streaming services, to protect video games from piracy and implement secure keyless entry systems for the automotive industry.
3. Irdeto is the registered owner of European Patent EP'787, which is in force in Germany and the Netherlands and relates to a method and system for preventing and detecting security threats to an operating system and certified applications operating on an electronic device. The patent application was filed on 30 March 2012 and published on 4 February 2015. The notice of the grant of the Patent was published by the EPO on 8 July 2020. No opposition was filed.
4. Claim 1 of the patent-in-suit, separated into features, reads as follows:
 1. *A method for preventing security threats on a device,*
 - 1.1 *the device having*
 - 1.1.1 *an abstraction layer comprising an operating system kernel and being operable between device hardware and application software,*
 - 1.1.2 *the abstraction layer having a secured software agent embedded within the abstraction layer,*
 - 1.2 *the method comprising the secured software agent performing the steps of:*
 - 1.2.1 *inserting one or more upcalls to the secured software agent at points in the operating system kernel where a user-level system call from an application would result in access to an internal operating system kernel object;*
 - 1.2.2 *receiving, from the operating system kernel via at least one of the one or more upcalls, a request to an internal operating system kernel object;*
 - 1.2.3 *determining whether the request is a valid request based on information located in a secured data storage; and*
 - 1.2.4 *limiting access to the abstraction layer based on the determining step, wherein the secured software agent has sole access to the secured data storage.*
5. The Patent also includes dependent claims 2 to 13, and an independent system claim 14 based on any one of the preceding claims.

6. The Patent was opted out from the exclusive jurisdiction of the UPC on 20 March 2024. The withdrawal of the opt-out was declared and registered on 10 April 2025.
7. The DJI Group is a company group that operates worldwide and sells commercial unmanned aerial vehicles (drones) for aerial photography and videography. Defendants 1) and 2), i.e. DJI Europe B.V. and DJI GmbH, belong to the DJI Group. DJI Europe B.V. is a Dutch company which acts as a reseller of DJI products in the EU market. In the DJI webstore (www.djistore.de), DJI Europe B.V. is also mentioned as the manufacturer, within the meaning of the Product Liability Act, for numerous DJI products, such as the DJI MAVIC 3 PRO. DJI GmbH is a German company, owned by DJI Europe B.V., and according to the commercial register, DJI GmbH is active in
 - 1.2.5 “sale and rental of remote-controlled model aircraft, remote-controlled aircraft, some of which are equipped with image recording devices, single or multi-rotor remote-controlled helicopters, image recording devices (in particular cameras, film cameras) and associated optical and electronic equipment, in each case including accessories, and the manufacture and conversion of the aforementioned equipment, the provision of technical services for the aforementioned equipment, in particular repair and maintenance, training in the operation of the aforementioned equipment, and media and marketing services”.
8. Defendant 3), Solectric GmbH, is based in Germany. It operates the DJI webstore at the URL www.djistore.de, where it offers and sells, inter alia, camera drones such as the DJI MAVIC 3 PRO and the DJI MAVIC 3 CLASSIC. Solectric GmbH also runs the DJI flagship stores in Germany and Austria.
9. On 15 April 2025, Irdeto initiated an action for infringement against DJI Europe B.V., DJI GmbH, Solectric GmbH and a fourth defendant (SZ DJI Technology Co., Ltd., based in China), for direct and indirect infringement of EP’787 by involvement in the distribution of DJI drones and their controllers. Additional requests for cease and desist orders (requests II.1.2, 1.2.1, 1.2.2 and 1.2.3) were added in the Reply to the Statement of Defence. The attacked embodiments are all DJI drones and their controllers that run a Linux/Android operating system with the security modules Security-Enhanced Linux (SELinux) and device-mapper verity (dm-verity). Examples of these drones are the following DJI models: Mavic 3 series, Air 3, Mini 2 SE, Mini 3 series, Mini 4 series, Inspire 3, Mavic 3 Enterprise series, Mavic 2 Enterprise Advanced, Mavic 3M, Matrice 30 series, Matrice 350 RTK, Matrice

3TD/3, FlyCart 30, T60/T25P, T50/T25, and T40/T50P, and the respective controllers (e.g. RC-RM330).

10. As the service on SZ DJI Technology Co., domiciled in China, lagged behind those on the other defendants by several months, and as the parties found no common ground on how to harmonize the procedural processes, the proceedings against SZ DJI Technology Co. (who allegedly manufactures the attacked embodiments in China and sells them directly or through intermediaries worldwide) have been separated from these proceedings and have become CFI_1391/2026 and CFI_639/2026 (cf. order of 16 March 2026). Before separation, now Defendants 1) to 3) (hereinafter collectively referred to as “Defendants”) were originally addressed as Defendants 2) to 4) and SZ DJI Technology Co. as Defendant 1).

11. On 14 August 2025, the three remaining Defendants in these proceedings submitted a Statement of Defence which included a Counterclaim for revocation.

12. The Claimant has submitted conditional applications to amend the Patent, but these applications have been found inadmissible (cf. order dated 15 April 2026, following the interim conference).

13. The value of the infringement action against Defendants 1) to 3) has been set to EUR 1.000.000. The value of their counterclaim has been set to EUR 2.000.000 (cf. order dated 15 April 2026, following the interim conference).

2. REQUESTS OF THE PARTIES

2.1 Infringement action

14. The Claimant requests

- I. a declaration by the Court that European Patent EP 2 831 787 B1 has been infringed by the Defendants by supplying and or offering to supply the drones and controllers referred to under II.1 within the territory of the Federal Republic of Germany and/or the Kingdom of the Netherlands to persons not entitled to exploit the patented invention;

II. that the Defendants are ordered to

1. cease and desist, within one week after notice on the enforcement according to Rule 118.8 RoP, from

1.1 supplying or offering to supply,

within the territory of the Federal Republic of Germany and the Kingdom of the Netherlands,

any person other than a party entitled to exploit the patented invention, drones and/or their controllers, in particular the DJI Mavic 3 Pro drones and/or controllers, having flash devices

that are suitable and intended for preventing security threats on a device, the device having an abstraction layer comprising an operating system kernel and being operable between device hardware and application software, the abstraction layer having a secured software agent embedded within the abstraction layer, the method comprising the secured software agent performing the steps of

inserting one or more upcalls to the secured software agent at points in the operating system kernel where a user-level system call from an application would result in access to an internal operating system kernel object;

receiving, from the operating system kernel via at least one of the one or more upcalls, a request to an internal operating system kernel object;

determining whether the request is a valid request based on information located in a secured data storage; and

limiting access to the abstraction layer based on the determining step, wherein the secured software agent has sole access to the secured data storage,

wherein limiting access to the abstraction layer through the secured software agent includes the secured software agent preventing loading software code to extend functionality of the abstraction layer; and

wherein the software code is a loadable kernel module.

for use in those countries.

(EP 2 831 787 – claims 1, 3 and 4 – indirect infringement)

in particular,

1.1.1 the operating system is a Linux operating system and the kernel is a Linux kernel, wherein optionally the secured software agent is compliant with a Linux Security Module;

(EP 2 831 787 – claim 2 – indirect infringement)

and/or

1.1.2 wherein limiting access to the abstraction layer through the secured software agent includes the secured software agent validating the loadable kernel module, and preventing loading is based on a successful validation, wherein optionally the secured software agent validates the loadable kernel module based on security information unique to the loadable kernel module stored in a secure store;

(EP 2 831 787 – claim 5 – indirect infringement)

and/or

1.2 supplying or offering to supply,

within the territory of the Federal Republic of Germany and the Kingdom of the Netherlands,

any person other than a party entitled to exploit the patented invention, drones and/or their controllers, in particular the DJI Mavic 3 Pro drones and/or controllers, having flash devices

that are suitable and intended for preventing security threats on a device, the device having an abstraction layer comprising an operating system kernel and being operable between device hardware and application software, the abstraction layer having a secured software agent embedded within the abstraction layer, the method comprising the secured software agent performing the steps of

inserting one or more upcalls to the secured software agent at points in the operating system kernel where a user-level system call from an application would result in access to an internal operating system kernel object;

receiving, from the operating system kernel via at least one of the one or more upcalls, a request to an internal operating system kernel object;

determining whether the request is a valid request based on information located in a secured data storage; and

limiting access to the abstraction layer based on the determining step, wherein the secured software agent has sole access to the secured data storage,

wherein limiting access to the abstraction layer through the secured software agent includes the secured software agent blocking a debug utility request,

for use in those countries.

(EP 2 831 787 – claims 1 and 10 – indirect infringement)

in particular,

1.2.1 wherein the operating system is a Linux operating system and the kernel is a Linux kernel, wherein optionally the secured software agent is compliant with a Linux Security Module;

(EP 2 831 787 – claim 2 – indirect infringement)

and/or

- 1.2.2 the method further comprising the secured software agent determining whether the debug utility request attempts to attach to any one of a certified application and a component of the abstraction layer, and blocking the debug utility request based on the determination;

(EP 2 831 787 – claim 11 – indirect infringement)

and/or

- 1.2.3 wherein the debug utility includes a process tracing system call to the abstraction layer, wherein optionally the debug utility is any one of ptrace and ADBd;

(EP 2 831 787 – claim 12 – indirect infringement)

and/or

2. cease and desist, within one week after notice on the enforcement according to Rule 118.8 RoP,

within the territory of the Federal Republic of Germany and the Kingdom of the Netherlands

from offering, placing on the market, or importing or storing for those purposes drones and/or controllers including at least one CPU and memory, the system arranged to carry out the method described in item II.1;

(EP 2 831 787 – claim 14 – direct infringement)

3. provide counsel for Claimant, within 30 days after service of the judgement, with information in an electronic Excel table structured for each month of a calendar year and according to patent infringing products, prepared and verified by an independent auditor – or any other professional the Court deems suitable for providing such statement –, about the extent to which the

Defendant has committed the acts referred to under II.1. and II.2. since 8 August 2020, stating in each case

(i) the names and addresses of manufacturers, suppliers and other previous owners;

(ii) the names and addresses of the commercial customers and the points of sale for which the products were intended;

(iii) the quantities of products delivered, received or ordered, as well as the prices paid for the products concerned; copies of the relevant purchase documents (namely invoices, or alternatively delivery notes) must be submitted as proof of the information provided, whereby details requiring confidentiality outside the data subject to disclosure may be redacted;

4. provide counsel for Claimant, within 30 days after service of the judgement, with an orderly statement in an electronic Excel table structured for each month of a calendar year and according to patent infringing products, prepared and verified by an independent auditor – or any other professional the Court deems suitable for providing such statement –, about the extent to which the Defendant has committed the acts referred to under II.1. and II.2. since 8 August 2020, stating in each case

(i) the individual deliveries, itemised according to delivery quantities, times and prices as well as type designations and the names and addresses of the customers;

(ii) the individual offers, itemised by offer quantities, times, prices, type designation and the names and addresses of the commercial recipients;

(iii) the advertising operated, broken down by advertising media, their circulation, distribution period and distribution area;

(iv) the prime costs broken down by the individual cost factors and the profit realized

whereby

the Defendants reserve the right to disclose the names and addresses of the non-commercial purchasers and the recipients instead of the Claimant to a sworn auditor located in Germany, to be designated by the Claimant and bound to secrecy towards it, provided that the Defendant bears his costs and authorises and obliges him to inform the Claimant upon specific request whether a particular purchaser or recipient is included in the list;

5. permanently remove the products described above under II.2. from the distribution channels by the Defendants taking back these items, if necessary enforcing their surrender with claims for surrender to which it is entitled or, at the Claimant's discretion, arranging for the destruction of these items at the respective owner's premises at the Defendants' expense;
6. recall the marketed products referred to above under item II.2. from the commercial customers with reference to the patent infringing condition of the products established by the court (judgment of the Local Chamber of Mannheim of [date]) and with the binding undertaking to reimburse any fees and to bear any necessary packaging and transport costs as well as customs and storage costs associated with the return and to take back the products, whereby the Claimant is to be provided with a sample of the recall letters as well as a list of the addressees with their names and postal addresses or - at the Defendants' discretion - a copy of all recall letters;
7. surrender to a bailiff to be appointed by the Claimant for the purpose of destruction, at their - the Defendants' - expense, the products referred to in item II.2 which are in Defendants' direct or indirect possession or ownership.

III. Any failure to comply with

1. the cease and desist order(s) referred to under item II.1. and II.2. will render the Defendants liable to pay to the Court a recurring penalty payment of EUR 250,000 for each violation of, or non-compliance with, the order(s), plus EUR 100,000 for each day, a part of a day counting as an entire day, that the violation or non-compliance continues, or a recurring penalty of EUR 5,000 for each

infringing product with which the order(s) is/are violated, or another amount as determined by this Court in the proper administration of justice (Art. 63(2), 82 UPCA and R. 354(3) RoP);

2. the claim for rendering of accounts and information referred to under item II.3. and II.4. will render the Defendants liable to a recurring penalty payment of up to EUR 10,000 to be determined by the Court and payable to the Court for each day of delay, within a period of 30 days from the date of service of the judgment and the date when the requirements referred to in Rule 118.8 of the Rules of Procedure have been met, whatever is later;

IV. Defendants are liable

1. to pay the Claimant an adequate compensation for the acts referred to under item II.1. and II.2. and committed from 30 April 2012 to 8 August 2020, and
2. for all damages resulting from the patent infringement referred to under II.1. and II.2. that occurred since 8 August 2020.

V. The Claimant has the right to publish the decision in whole or in part in public media, in particular on the Internet, at the Defendants' expense and in compliance with the General Data Protection Regulation.

VI. Defendants are ordered to bear reasonable and proportionate legal costs and other expenses incurred by the Claimant in these proceedings and orders, insofar such costs are to be determined in separate proceedings for the determination of such costs, that the Defendant pays to the Claimant by means of an interim award of costs an amount of EUR 124,000 within 14 days after service of the judgment in this matter (Art. 69 UPCA, R. 118.5 and 150.2 RoP).

VII. The judgment shall be immediately enforceable (Art. 82(1) UPCA). Claimant may provide security in the form of a bank guarantee or a savings bank guarantee, the

amount of security to be determined separately for each enforceable claims of the decision, with the following proposed individual amounts:

Claim II.1. and 2. (injunctive relief): 1,500,000 EUR

Claim II.3. and 4. (rendering of accounts and information): 50,000 EUR

Claim II.5. and 6. (recall and removal): each 100,000 EUR

Claim V. (publication of the judgement): 50,000 EUR

Claim VII. (costs): is provisionally enforceable for the Claimant against security amounting to 110 per cent of the amount to be enforced

15. The Defendants request

- I. that the infringement action be dismissed,
 - II. that the Court orders the Claimant to bear the costs of the infringement action,
 - III. that the Claimant be ordered to reimburse the costs of the infringement action on a provisional basis (R. 150.2 RoP) in the amount of EUR 200.000,
- or, in the alternative,
- IV. that enforcement of the decision is made subject to the prior provision of a security by the plaintiff to the other parties in at least the amount mentioned in the unredacted version of the Statement of defence (R. 352.1, 354.2 RoP), which may be provided by a written, irrevocable, unconditional and unlimited guarantee from a credit institution authorized to do business in the territory of a member state of the Unified Patent Court ("UPC"),
 - V. that the defendants be allowed to avert the enforcement of the decision by providing security, which can be provided by a written, irrevocable, unconditional and unlimited guarantee from a credit institution authorized to do business in the territory of a member state of the UPC, without regard to the provision of security by the plaintiff (R. 9.1 RoP).

2.2 Counterclaim for revocation

16. The Defendants request

- I. that the Patent is revoked entirely with effect to the territory of Germany (DE) and the Netherlands (NL),
- II. that the Claimant be ordered to bear the costs of the Counterclaim for revocation,
- III. that the Claimant be ordered to reimburse the costs of the Counterclaim for revocation on a provisional basis (R. 150.2 RoP) in the amount of EUR 200.000.

17. The Claimant requests

- I. that the counterclaim for revocation be dismissed,
- II. that the patent-in-suit is maintained as granted,
- III. that the Defendants shall jointly and severally bear the Claimant's costs of the Counterclaim for Revocation,
- IV. that the Defendants shall jointly and severally pay the Claimant, within six weeks after the service of the decision to be rendered in the Counterclaim for Revocation proceedings on one of the Defendants, an interim award of the procedural costs.

3. POINTS AT ISSUE

18. The parties are in dispute about different aspects of the case at hand

3.1 Infringement

19. According to the Claimant, SELinux and dm-verity form (together) a secured software agent, in the sense of the patent-in-suit, that acts in accordance with the claimed features. The SELinux functionality of this combined agent uses the LSM framework and inserts hooks/upcalls at boot time, and receives the request from the kernel via an upcall (thereby fulfilling features 1.2.1 and 1.2.2), while the dm-verity functionality of this combined agent validates the request by determining whether access shall be granted based on information

located in its sole access secured data storage (thereby fulfilling features 1.2.3 and 1.2.4). As a result, all features of claim 1 are, according to the Claimant, fulfilled. In Claimant's view, the implementation of SELinux and dm-verity also results in the realization of the features of the asserted dependent claims and claim 14.

20. The Defendants opine that if the Court finds that the patent-in-suit is valid, this conclusion must be based on a claim construction that also leads to the attacked embodiments not falling within the scope of the patent's claims. Here, the Defendants argues inter alia that SELinux and dm-verity (together) does not form a secured software agent in the sense of the patent-in-suit, that SELinux and dm-verity does not work together on a same user-level system call, that dm-verity is not mounted on LSM through hooks/upcalls like SELinux and that dm-verity does not perform validation in the attacked embodiments according to feature 1.2.4 at runtime (after the upcall), as required by the patent-in-suit.

21. For further details, it is referred to the parties' briefs and the corresponding Exhibits.

3.2 Counterclaim for revocation

22. The Defendants base their counterclaim for revocation on the following grounds of Article 138 EPC in conjunction with Article 65(2) UPCA:

- added matter (Article 138(1)(c) EPC).
- lack of disclosure in a manner sufficiently clear and complete for the invention to be carried out by a person skilled in the art (Article 138(1)(b) EPC),
- lack of novelty (Article 138(1)(a) in conjunction with Articles 52 and 54 EPC),
- lack of inventive step (Art. 138(1)(a) in conjunction with Articles 52 and 56 EPC).

23. The Defendants argue that the subject-matter of the patent-in-suit lacks novelty in view of

- The Prior Linux version,
- US 2009/0271844 A1 (D1)
- The article "Linux Security Modules: General Security Support for the Linux Kernel" (D6)

24. Defendants further argue that the subject-matter does not involve an inventive step, based on the following combinations:

- The Prior Linux version alone.
- US 2009/0271844 A1 (D1), alone or in combination with US 2005/0005101 A1 (D2) or US 2011/0047542 A1 (D3) or US 2009/0007233 A1 (D4).

- US 7,644,271 B1 (D5) in combination with US 2009/0271844 A1 (D1).
- The article "Linux Security Modules: General Security Support for the Linux Kernel" (D6) in combination with US 2005/0005101 A1 (D2) or US 2011/0047542 A1 (D3) or US 2009/0007233 A1 (D4) or US 7,644,271 B1 (D5).

25. The Claimant rejects the attacks on the validity of the patent-in-suit as unfounded.

26. For further details, it is referred to the parties' briefs and the corresponding Exhibits.

4. GROUNDS FOR THE DECISION

4.1 Admissibility

27. Neither jurisdiction nor competence have been challenged by the Defendants. The Court also notes that two Defendants are based in Germany while the third Defendant (based in the Netherlands) is involved in the alleged infringements in Germany.

28. The Patent was opted out from the exclusive jurisdiction of the UPC on 20 March 2024. The withdrawal of the opt-out was declared and registered on 10 April 2025.

29. Thus, the UPC has jurisdiction, and LD Mannheim has competence to hear the case.

30. The admissible requests for cease and desist orders that were formally added by the Claimant in the Reply to the statement of defence (II.1.2, 1.2.1, 1.2.2 and 1.2.3) are only specifications of requests that were included in the statement of claim in substance (cf. SoC, paragraph 39) and does not amend the case. The Defendants' defence have always covered also these alleged infringements.

4.2 The substantive scope of the patent-in-suit

4.2.1 Background and object

31. The patent-in-suit relates to a method and a system for preventing and detecting security threats to an operating system and certified applications operating on a device (cf. paragraph [0001] of the description of the patent-in-suit).

32. According to the patent-in-suit, mobile phones, tablets, games consoles, set top boxes, televisions, personal navigation devices, and other consumer electronics devices have increasingly become open devices and may e.g. include open source operating systems

such as Linux or Android (cf. paragraphs [0002] and [0003]). Such open devices are, according to the patent, more susceptible to hacker attacks and increasingly support the capability for consumers to download, install, and execute third-party software applications, which also increases the potential security concerns (paragraphs [0004] and [0005]).

33. The Patent describes that third-party software may contain malware, i.e. malicious software such as e.g., worms, viruses, Trojans, rootkits, and backdoors, that may cause e.g. breach of consumer privacy, identity theft, fraudulent use of the device or related services and network stability problems (paragraph [0006]). It is also mentioned that although applications may be certified, a certification process can be circumvented or not exhaustive, which may result in consumers unknowingly downloading malware onto their devices (paragraph [0007]).
34. In paragraphs [0008] and [0009], rootkits are described as a particular type of malware that enables continued privileged access to a computer while actively hiding its presence from administrators by subverting standard operating system functionality or other applications. An attack by rootkit malware typically uses vulnerability or capability to install a payload such as additional malware components that can continue to operate behind the scenes, receiving and executing new instructions from a remote server (e.g. surreptitious uploading of private user information, sending spam or launching distributed denial-of-service (DDOS)-attacks). It is also mentioned that many rootkits use loadable kernel modules to modify the running operating system kernel to execute the payload.
35. According to paragraph [0010], rootkit detection is difficult because a rootkit may be able to subvert the software that is intended to find it. Known detection methods include using an alternative, trusted operating system, behavioral-based methods, signature scanning, difference scanning, and memory dump analysis. Furthermore, removal of a rootkit can be complicated or practically impossible, especially if the rootkit resides in the kernel or when dealing with firmware rootkits.
36. Paragraphs [0011] to [0024] of the patent-in-suit describes several existing approaches to platform security (i.e., security intended to address one or more of the security problems noted above), such as process isolation, access control, private application programming interfaces (APIs), application certification/signing, application licensing services, secure

boots, virus detection and intrusion prevention software and virtual machines. These paragraphs also describe specific problems relating to some of these methods, e.g.:

- The effectiveness of *process isolation*, i.e. to ensure that each application and parts of the system run in its own process and dedicated memory space, is limited by the fact that most applications that are useful to users must access operating systems services outside its “sandbox”.
- *Access control* is a functionality that inter alia makes decisions about whether to grant a requesting application access to Operating Systems (OS) services outside its sandbox. Such functionality is limited by the fact that they may be combined with the concept of permissions and that the grantor of such permissions might not understand the implications of such approval (it might e.g. be the consumer), or that such permissions may be modified by malware or an attacker. In this context, the patent-in-suit mentions that some operating systems have access control frameworks that enable different access control models to be implemented, e.g. that the Linux Security Module (LSM) enables different access control models and functions to be implemented as loadable kernel modules. In paragraph [0045], Mandatory Access Control (MAC) is explicitly mentioned as an example of a security function that can be implemented using LSM.
- *Application certification/signing* generally includes static verification (e.g. scanning the object code prior to execution) of the application (e.g. to verify that private API’s are not called by the application) and dynamic verification (e.g. to verify the “stability” of the application during execution). This functionality is limited by the fact that comprehensive verification is not readily automated and exhaustive, so malicious operation can avoid detection during the verification process by being embedded in the application and only execute at a pre-specified time following the certification/signing process. Other problems with application certification are the sheer number of applications, which makes it cost-prohibitive to perform exhaustive verification before certification, and that a hacker could modify or replace the information used to verify

the integrity of the application to allow a hostile third party to load an unauthorized or pirated application onto the device by a consumer.

- *Application licensing services* have the limitation that such protection against application piracy can be readily circumvented by hackers, by modifying the application to extract such license verification checks.
- *A secure boot loader/secure boot*, i.e. identification or integrity verification of the boot software and the OS kernel during device start-up to ensure that only the intended boot software and OS kernel are loaded onto the device, has the weakness that this mechanism can be circumvented by dynamic attacks occurring during the boot process. Once the secure boot loader has been bypassed, the OS can be modified to bypass other security functions that may be present in the OS, and there is no way to restore device security for devices already deployed in the field once the secure boot process has been compromised.
- *Virus detection and intrusion prevention software* to detect malware and mitigate any damage are usually based on a set of attributes such as being of a specific length and having a specific sequence of bytes at a certain location to identify a specific piece of malware. These signatures/attributes are only understood once the malware has been deployed, meaning the malware may have already caused damage. Additionally, these signature-based types of solutions must be constantly updated and the anti-virus software itself can be modified or disabled by malware to prevent such detection.
- *Virtual machines*, such as the Java™ virtual machine (JVM), are designed to allow the safe execution of applications obtained from potentially untrusted sources letting applications run in a restricted sandbox which is designed to protect the user from misbehaving code or malware, but this comes with performance limitations and limitations in terms of the functionality.

37. In addition to the specific problems that relates to respective method, the patent-in-suit describes that many of these methods have the common problem that the OS that supports such security functions can be subverted or bypassed by modifying portions of the operating system that perform such functions, and that, once an attack has occurred, there is no mechanism in place to recover platform security for devices that have been sold or

licensed or otherwise distributed to consumers so that these methods provide only “static security” (paragraphs [0018] and [0021]).

38. According to the description (paragraph [0024]), each of these security methods form part of a static platform security functionality, as shown in prior art Figure 1:

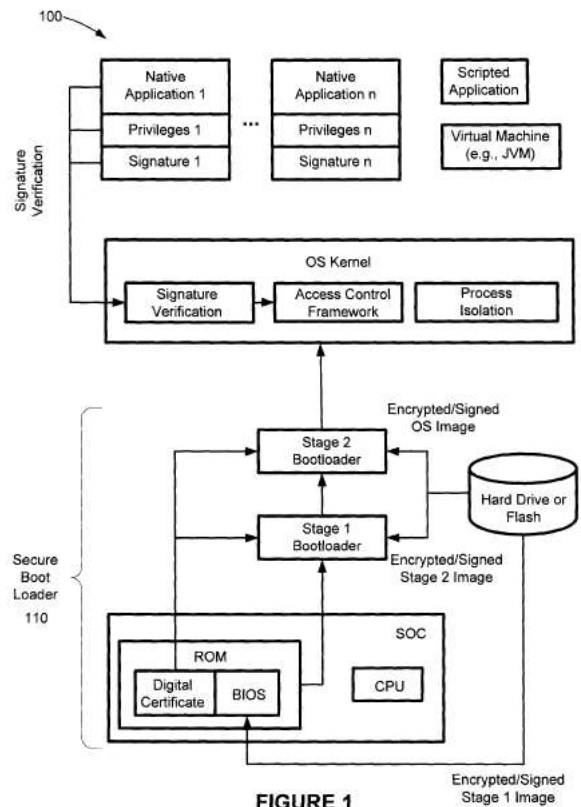


FIGURE 1
(Prior Art)

39. The purpose of the invention is to provide an improved security mechanism for preventing unauthorized use of a device and digital assets on that device that overcomes some of the problems mentioned above and the limitations of static platform security (cf. paragraph [0025]).

4.2.2 Solution

40. The patent in suit solves this object by providing the method of claim 1, which can be structured as follows:

1.	A method for preventing security threats on a device,
1.1	the device having

1.1.1	an abstraction layer comprising an operating system kernel and being operable between device hardware and application software,
1.1.2	the abstraction layer having a secured software agent embedded within the abstraction layer,
1.2	the method comprising the secured software agent performing the steps of:
1.2.1	inserting one or more upcalls to the secured software agent at points in the operating system kernel where a user-level system call from an application would result in access to an internal operating system kernel object;
1.2.2	receiving, from the operating system kernel via at least one of the one or more upcalls, a request to an internal operating system kernel object;
1.2.3	determining whether the request is a valid request based on information located in a secured data storage; and
1.2.4	limiting access to the abstraction layer based on the determining step, wherein the secured software agent has sole access to the secured data storage.

41. This method can be illustrated by Figure 2A, which is a schematic showing of an embodiment as applied to an Android OS, and Figure 3, which is a schematic showing of certain aspects of dynamic platform security functionality in accordance with the embodiment of Figure 2A:

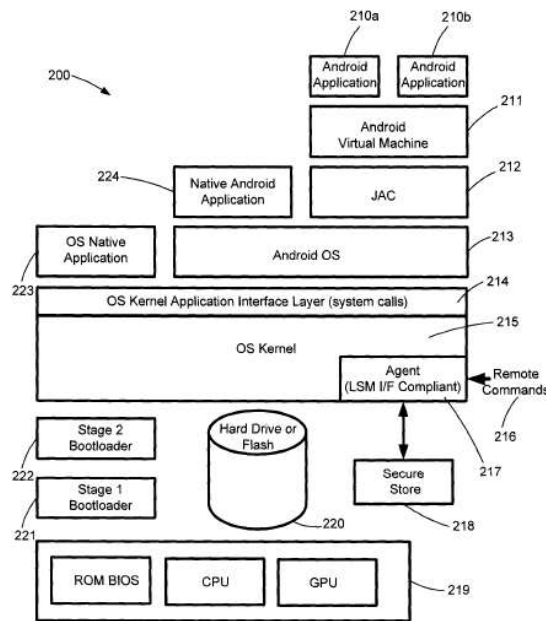


FIGURE 2A

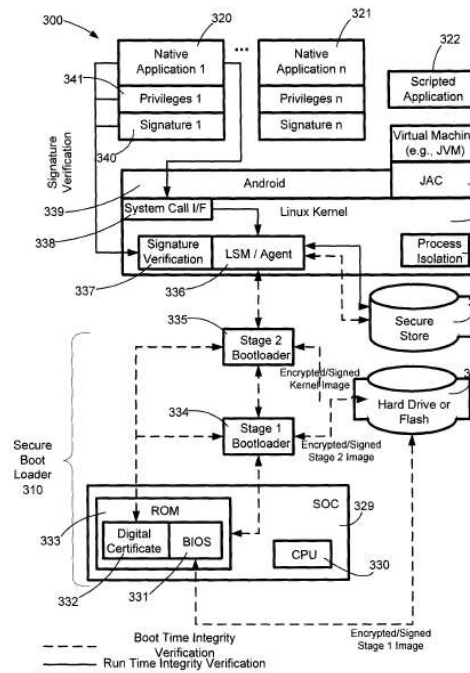


FIGURE 3

42. The patent also includes the following dependent claims 2–13 and independent system claim 14:

2.	The method of claim 1 wherein the operating system is a Linux operating system and the kernel is a Linux kernel, wherein optionally the secured software agent is compliant with a Linux Security Module.
3.	The method of any claim 1 or 2 wherein limiting access to the abstraction layer through the secured software agent includes the secured software agent preventing loading software code to extend functionality of the abstraction layer.
4.	The method of claim 3 wherein the software code is a loadable kernel module.
5.	The method of claim 4 wherein limiting access to the abstraction layer through the secured software agent includes the secured software agent validating the loadable kernel module, and preventing loading is based on a successful validation, wherein optionally the secured software agent validates the loadable kernel module based on security information unique to the loadable kernel module stored in a secure store.
6.	The method of any one of claims 3 to 5 wherein a portion of the secured software agent is incorporated into a kernel utility for loading software code

	to extend the functionality of the abstraction layer, wherein optionally the kernel utility is insmod.
7.	The method of claim 1 or 2 wherein limiting access to the abstraction layer through the secured software agent includes the secured software agent blocking over-writing at least one pointer to system calls to the abstraction layer.
8.	The method of claim 1 or 2 wherein limiting access to the abstraction layer through the secured software agent includes the secured software agent blocking writing a system call table having at least one pointer to a system call.
9.	The method of claim 7 or 8 wherein the secured software agent blocks writing to a memory range containing the system call table.
10.	The method of claim 1 or 2 wherein limiting access to the abstraction layer through the secured software agent includes the secured software agent blocking a debug utility request.
11.	The method of claim 10 further comprising the secured software agent determining whether the debug utility request attempts to attach to any one of a certified application and a component of the abstraction layer, and blocking the debug utility request based on the determination.
12.	The method of any one of claims 10 and 11 wherein the debug utility includes a process tracing system call to the abstraction layer, wherein optionally the debug utility is any one of ptrace and ADBd.
13.	The method of any one of the previous claims wherein the secured software agent maintains the relevancy of the information located in the secured data storage (218).
14.	A system comprising device hardware including at least one CPU and memory, the system arranged to carry out the method of any one of the proceeding claims.

4.2.3 Skilled person and claim construction

43. Some features of claim 1 need further explanation. Based on CoA Order of 26 February 2024, in case CoA_335/2023, *NanoString v. 10x Genomics*, the UPC approach to claim interpretation is as follows.
- The patent claim is not only the starting point, but the decisive basis for determining the protective scope of a European patent under Art. 69 EPC in conjunction with the Protocol on the Interpretation of Art. 69 EPC.
 - The interpretation of a patent claim does not depend solely on the strict, literal meaning of the wording used. Rather, the description and the drawings must always be used as explanatory aids for the interpretation of the patent claim and not only to resolve any ambiguities in the patent claim.
 - However, this does not mean that the patent claim merely serves as a guideline and that its subject-matter also extends to what, after examination of the description and drawings, appears to be the subject-matter for which the patent proprietor seeks protection.
 - The patent claim is to be interpreted from the point of view of a person skilled in the art.
 - In applying these principles, the aim is to combine adequate protection for the patent proprietor with sufficient legal certainty for third parties.
 - These principles for the interpretation of a patent claim apply equally to the assessment of the infringement and the validity of a European patent.
44. In this case, the person skilled in the art is a computer engineer with several years of experience and familiar with different kinds of Operating Systems (OS) like Unix, in particular Linux or Android, as well as with security mechanisms for those OS. This skilled person would understand the features as follows.

Feature group 1, more specifically 1.1

1. A method for preventing *security threats* on a *device*
 - 1.1 the device having
 - 1.1.1 an *abstraction layer* comprising an operating system kernel and being operable between device hardware and application software,
 - 1.1.2 the abstraction layer having a *secured software agent embedded* within the abstraction layer,

45. The **device** can be any consumer electronic device that runs an operating system, e.g. Linux, see paragraphs [0002], [0003] and [0032] of the patent-in-suit.
46. According to the description, the invention is particularly concerned with **security threats** caused by installing malware - such as worms, viruses, Trojans, rootkits, and backdoors – which can cause identity theft or fraudulent use of the device (paragraph [0006]). Such security threats can, according to the patent, be caused by loadable kernel modules which modify the operating system kernel (paragraphs [0008] to [0010] and [0076]), but also by e.g. the use of the system call table (paragraphs [0080] to [0084]) or debugging (paragraphs [0085] to [0088]). Thus, the patent is directed to this kind of security threats.
47. The Claimant argues – with reference to the description (e.g. paragraphs [0035] and [0036], and Figures 2A to 3) – that the **abstraction layer** corresponds to, or is part of, an operating system kernel, while the Defendants argue that the abstraction layer is a different and broader concept than the operating system kernel.
48. The wording of feature 1.1.1 clearly states that the abstraction layer is operable between device hardware and application software and that it comprises, i.e. includes but is not limited to, an operating system kernel. An operating system acts as an intermediary between the hardware and the application programs and it includes a kernel. Thus, the abstraction layer, which is a layer that abstracts and separates the software from the hardware while encompassing the former so that there is no need to use hardware-specific software, is in some aspects an operating system comprising the OS kernel (cf. paragraph [0026]), but it is not necessarily limited to an operating system or its kernel and may provide additional functionalities.
49. The fact that the description contains statements and figures indicating that the abstraction layer would correspond to a kernel in which the secured software agent is embedded (e.g. paragraphs [0035] and [0036]) does not lead to a different conclusion, since this only

relates to special embodiments and the broader claims are the decisive basis for determining the protective scope (supra). The Court also notes that those parts of the description relate to embodiments based on the Linux kernel, while it follows from the claims, including dependent claim 2 (see also e.g. paragraph [0032] in the description) that the patent-in-suit in principle also covers other operating systems and/or kernels.

50. It follows from features 1.1.1 and 1.1.2 that the ***secured software agent*** must be embedded in the same (identical) abstraction layer as the kernel (cf. feature 1.1.2: “the abstraction layer”). In this context, the person skilled in the art would understand ***embedded*** to mean that the secure software agent in its entirety, or at least the functionality responsible for validation (feature 1.2.3), resides in this abstraction layer. Otherwise, the validation process could be tampered with, which these features are intended to avoid (cf. feature 1.2.4). All embodiments place the secured software agent within the OS kernel as part of the abstraction layer (cf. e.g. Figures 2A-C, 3, paragraphs [0035] and [0036]). Nothing else follows from dependent claim 6 according to which a portion of the secured software agent is incorporated into a kernel utility for loading software code to extend the functionality of the abstraction layer, wherein optionally the kernel utility is insmod. Such kernel utility also resides in the operating system and therefore in the abstraction layer.

51. If validation (feature 1.2.3, cf. infra) is carried out by means of integrity verification (cf. e.g. paragraph [0046]), both the calculation of the hash value of the object to be validated and its comparison with a known good value for the hash function are functionalities responsible for validation. Consequently, according to the description, the function (integrity verification) is embedded in the secured software agent which, in turn, is embedded in the abstraction layer (paragraph [0046], Figure 3).

Feature group 1.2

- 1.2 the method comprising the secured software agent performing the steps of:
 - 1.2.1 inserting one or more **upcalls** to the secured software agent at points in the operating system kernel where a **user-level system call** from an application would result in access to an **internal operating system kernel object**;
 - 1.2.2 receiving, from the operating system kernel via at least one of the one or more upcalls, a **request** to an internal operating system kernel object;
 - 1.2.3 determining whether the request is a **valid request** based on information located in a **secured data storage**; and
 - 1.2.4 limiting access to the abstraction layer based on the determining step, wherein the secured software agent has **sole access** to the secured data storage.

52. Even though claim 1 is not limited to Linux (supra), the person skilled in the art reading the patent-in-suit would understand that "**upcalls**" relate to the essential characteristics of the generally known mechanism from Linux Security Module (LSM) used to enforce security policies within a Linux kernel, i.e. the hooks. These hooks (upcalls to the agent) are inserted at points in the kernel where a user-level system call from an application is about to result in access to an important internal kernel object (paragraphs [0035] and [0045]), cf. "would result in access" in feature 1.2.1. Thus, the person skilled in the art would also understand that "**user-level system calls**" relate to the fundamental interface between an application and the kernel in an operating system such as Linux, that allows user programs to request services from the kernel, and that "**internal operating system kernel objects**" relate to mechanisms provided by the kernel, such as inodes or task control blocks storing information about a process or metadata.

53. Consequently, it follows from features 1.2 and 1.2.1 that if an application at user level tries to access an important object in the internal operating system kernel, the kernel will – unless access to the object is blocked for other reasons (e.g. because the object is deleted) – send an upcall to the secured software agent requesting the determination specified in features 1.2.3 and 1.2.4.

54. The upcall(s) sent from the operating system kernel to the secured software agent will include a **request** to an internal operating system kernel object. The substance of this request, which essentially stems from the user-level system call, can, according to the description, be e.g. a request for loading operating system components when resetting the device (Figure 4), a request for loading an unsigned application from a developer device (Figure 5), a request to install an application (Figure 6), a running application requesting kernel services (Figures 7 and 9), a request to install a loadable kernel module into the OS

kernel to extend the running OS kernel (Figure 10), a request to (over-)write memory containing the system call table (Figure 11), or a debug request (Figure 12).

55. When the secured software agent receives the request sent from the kernel via the upcall, it shall determine whether this request is a **valid request**. To this end, the secured software agent shall use information from a secured data storage to which it has sole access. It follows from features 1.2.1 and 1.2.2 in conjunction with feature 1.2.3 ("the request"), and is confirmed by the embodiments, that the validation relates to and is therefore triggered by the very same request that is received via an upcall (feature 1.2.2). This dynamical validation makes it possible to identify any tampering that may have occurred, for example, between the secured booting of the system or the secured installation of the application making the request and the time of the request (e.g. Figures 5 to 7, paragraphs [0055], [0057], [0058], [0059] and [0066]; Figure 9, paragraph [0073]).
56. The Claimant has modified its position in this regard during the proceedings and now argues that since features 1.2.1 and 1.2.2 define the usage of "upcalls" of the type found in LSM, which primarily are used to regulate access to the kernel, the person skilled in the art would understand that the validation of the request to an internal operating system kernel object in feature 1.2.3 is performed in addition to the mandatory access control via the upcalls. In other words, the determination of validity in feature 1.2.3 refers – according to the Claimant – to other aspects than access control, in particular integrity verification. This is disputed by the Defendants, who in summary argue that the validation shall be done against any rules or policies stored in the secured data storage.
57. Since the validation relates to and shall be triggered by a request sent via an upcall, and this upcall shall be the result of a user-level system call, the person skilled in the art would understand that the validation must take place during runtime. Thus, validation that only takes place during boot time is not sufficient. This is in line with the problem to be solved by the patent-in-suit, i.e. to inter alia overcome the limitations of static platform security (paragraph [0025], see mn. 39). The criticized disadvantages of this static approach are that it can be bypassed or subverted (cf. paragraphs [0018] and [0021]). In fact, the validation during runtime, i.e. a dynamic approach as opposed to the static approach, runs through the entire description and figures (cf. e.g. Figure 2 in combination with Figure 3 ("dynamic platform security functionality", in contrast to Figure 1 "static platform security"); Figure 8 in combination with Figure 3 ("validation of a user application request during runtime"),

and paragraphs [0034], [0036], [0038], [0039], [0043], [0046], [0047], [0048], [0049], [0055], [0057], [0058], [0059], [0063], [0066], [0067], [0073], [0074] and [0084]). As already mentioned, this dynamic approach helps to ensure that security features cannot be bypassed or subverted, as might be the case with a static approach.

58. Furthermore, it is true that validating the request must be more than just checking whether the request has the required permissions, as done by Security-Enhanced Linux (SELinux), a module of the LSM framework. The patent itself states that the LSM access control framework is well known (paragraphs [0014] and [0035], see also paragraph [0061]), and it is undisputed that also SELinux was well known at priority date. Moreover, the patent criticizes such legacy access control framework that is based on permissions granted as not sufficient, because the permission may be ill-granted or tampered with (paragraphs [0014], [0061]), and refers to additional factors such as OS kernel integrity, application integrity, application context, and the privileges granted by any given trusted root authority (e.g. paragraph [0037]). Consequently, the embodiments do not limit validation to merely granting access in accordance with permissions attributed to an application like it is in the case of SELinux. They rather perform an additional step, for example signing an enhanced permission container manifest of an unprotected asset like an application and storing it in an application secure store 512/610 for reference of installation, whereby in the event of such an installation request the applicant's signed enhanced permission container manifest is validated through integrity verification and, if successful, stored in the secure data storage 612 of the secured software agent for further reference of other permission requests for that application (Figures 5 and 6, paragraphs [0057] and [0058]). In the embodiments of Figures 7 and 9, which also deal with permissions attributed to the (now installed) applications, these permissions are verified in a similar way upon receiving the relevant request, using the enhanced permission container manifest stored in the secured data storage (paragraphs [0066] and [0073]). Thus, it is clear that the SELinux functionality is not sufficient to qualify as a validation in accordance with feature 1.2.3.

59. The meaning of the term 'sole access' in feature 1.2.4 does not require clarification in all respects. Given its function of protecting the contents of the secured data storage against tampering, the feature 1.2.4 at least requires, in any event, that only the secured software agent be able to write to the secured data storage.

4.3 Validity

4.3.1 Added matter

4.3.1.1 *Legal framework*

60. Pursuant to Article 65(2) UPCA and Article 138(1)(c) EPC, a European patent may be revoked if its subject-matter extends beyond the content of the application as filed.

61. In order to assess whether there is inadmissible added matter, the Court must first ascertain, what the skilled person would derive directly and unambiguously using his common general knowledge and see objectively and relative to the date of filing, from the whole of the application as filed, whereby implicitly disclosed subject-matter, i.e. matter that is a clear and unambiguous consequence of what is explicitly mentioned, shall also be considered as part of its content (Court of Appeal, decision of 25 November 2025, UPC_CoA_528/2024, para. 54 – Amgen v. Sanofi; order of 14 February 2025, UPC_CoA_382/2024, para. 52 – Abbott v Sibio). Accordingly, literal support is not required to comply with Art. 138(1)(c) EPC. Neither is it required that all features of the claim can be found in one paragraph or one example of the application (Court of Appeal, decision of 25 November 2025, UPC_CoA_528/2024, para. 90 – Amgen v Sanofi). The assessment of whether there is added matter is a question of law to be decided on the basis of the facts brought forward by the parties. The facts are the relevant claims and the application as filed. Since the test is whether the relevant claims have basis in the application as a whole, the Court is allowed to look at the entire document (Court of Appeal, ibidem, headnote no. 4).

4.3.1.2 *No added matter in the case at hand*

62. The Defendants argue inter alia that the combination of (the sole) features 1.2.1, 1.2.2 and 1.2.3 without requiring that this mechanism exclusively concerns the loading of Loadable Kernel Module (LKM) and specifically the validation of LKM by using integrity verification amounts to an intermediate generalization which goes beyond the disclosure of the patent application as filed. They also argue that paragraphs [0047] and [0048] of the application as filed (paragraphs [0035] and [0036] in EP'787) do not disclose that the secure agent would always, with respect to any request and/or access control decision, use the hook

mechanism and, in turn, validate the request based on information located in a secure data storage.

63. However, the Court agrees with the Claimant that paragraphs [0047] and [0048] of the patent application disclose features 1.2.1, 1.2.2 and 1.2.3, i.e. the combination of the upcalls to the agent and a secured data storage used by the agent as claimed, without an inextricable link to the loading of an LKM. Particularly, integrity verification is mentioned in paragraph [0048] only as one example for the agent's performance. In this context, paragraph [0049] of the application deals with kernel resources and data access control performed by the agent. It discloses that the agent's performance with regard to validation may occur when an application attempts to access OS kernel resources, i.e. to access a kernel object, which is done via hooks as explained in the preceding paragraph [0047] of the application.

64. This is further confirmed by Figure 3, where the agent performs integrity monitoring (with regard to the exemplary components mentioned in paragraph [0060] of the application) when the native application¹ 320 attempts to access system resources through the system call I/F 338. Moreover, this is also reflected in the following figures of the application. The fact that some figures do not depict the secure data storage is not decisive because the basic structure is shown in Figures 2A-C and Figure 3. Against this backdrop, the person skilled in the art understands that the embodiment in relation to LKM and integrity verification is merely a specific example of a general teaching that is reflected in feature group 1.2. Furthermore, claim 1 does not necessitate that a validation is only carried out by the secured software agent using the hook mechanism. It is sufficient for compliance with claim 1 if a method provides the mechanism required by feature group 1.2.

65. Thus, the claimed subject-matter is directly and unambiguously disclosed in the patent application as filed.

4.3.2 Sufficiency of disclosure

4.3.2.1 *Legal framework*

66. Pursuant to Article 65(2) UPCA and Article 138(1)(b) EPC, a European patent may be revoked if the patent does not disclose the invention in a manner sufficiently clear and complete for it to be carried out by a person skilled in the art.

67. Sufficiency shall be examined based on the patent as a whole, i.e. the claims, description and drawings, from the perspective of the person skilled in the art with his/her common general knowledge at the filing or priority date. The test to be applied is whether the person skilled in the art is able to reproduce the claimed subject-matter on the basis of the patent without any inventive effort and without undue burden. An invention is sufficiently disclosed if the patent specification shows the skilled person at least one way – and in case of functional features: one technical concept – of performing the claimed invention. Where a claim contains one or more functional features, it is not required that the disclosure includes specific instructions as to how each and every conceivable embodiment within the functional definition(s) should be obtained. A fair protection requires that variants of specifically disclosed embodiments that are equally suitable to achieve the same effect, which could not have been envisaged without the invention, should also be protected by the claim. Consequently, any non-availability of some embodiments of a functionally defined claim is immaterial to sufficiency, as long as the skilled person through the disclosure is able to obtain suitable embodiments within the scope of the claim. The burden of presentation and proof lies with the party invoking invalidity of the patent (cf. Court of Appeal, decision of 25 November 2025, UPC_CoA_528/2024, paras. 105 et seqq. – Amgen v. Sanofi).

4.3.2.2 *Sufficiency in the case at hand*

68. The Defendants argue inter alia that even though the description makes use of the known LSM framework (see e.g. paragraph [0035]), the description does not provide explanations to the skilled person on how to implement the claimed invention using this framework. In this context, they refer inter alia to that the skilled person has no information as to whether the storage where the LSM policies are stored shall be used or not by the secured agent and identifies certain questions that might arise if assuming that the claimed secured data storage would differ from the LSM storage. Furthermore, the Defendants assert that, if the patent-in-suit were inventive in view of the running previous Linux version and the Linux source tree published on 28 March 2012 that contains the source code of the device-mapper verity (dm-verity) functionality, the skilled person would equally not be able to

implement dm-verity into a running previous Linux version without inventive effort and undue burden in order to arrive at the teaching of claim 1.

69. These arguments are not convincing. In any case, the Defendants' argument relates to the LSM framework and dm-verity but claim 1 is not limited to using a LSM framework and dm-verity, let alone to a Linux operating system. Consequently, it is sufficient for sufficient disclosure that a skilled person in general when designing an operating system from the scratch and wishing to implement the invention is able to do so without involving any inventive effort and without undue burden. Defendants did not demonstrate that this is not the case.

70. Thus, the patent discloses the invention in a manner sufficiently clear and complete for it to be carried out by a person skilled in the art.

4.3.3 Novelty

4.3.3.1 Legal framework

71. Pursuant to Article 65(2) UPCA and Article 138(1)(a) EPC, a European patent may be revoked if the subject-matter of the European patent is not patentable under Articles 52 to 57 EPC. To be patentable, the invention must, inter alia, be new (Article 52 EPC). Art. 54 EPC specifies that an invention shall be considered to be new if it does not form part of the state of the art, which shall be held to comprise everything made available to the public by means of a written or oral description, by use, or in any other way, before the date of filing of the European patent application.

72. The assessment of novelty requires the determination of the whole content of the prior art publication. It is decisive whether the subject matter of the claim with all its features is directly and unambiguously disclosed in the prior art citation (see Court of Appeal, Order of 25 September 2024, UPC_CoA_182/2024, Mammut v. Ortovox, para. 123). In addition, the disclosure also may be implicit when a person skilled in the art would objectively consider such disclosure as necessarily implied in the explicit content of a prior-art document or would arrive, inevitably and without any reasonable doubt, at the result falling within the scope of the claim by applying the teaching of the prior-art document (see e.g. LD Helsinki, 29 April 2026, UPC_CFI_403/2025, AIM Sport Development v. TGI Sport).

4.3.3.2 *Novelty of claim 1 in view of the Prior Linux version*

73. The Defendants allege that claims 1, 2, 3, 4, 5, 10, 11, 12 and 14 are not novel in view of the Prior Linux version. The Prior Linux Version that serves as the basis for this novelty attack is actually the Linux kernel source tree publicly available online on 28 March 2012, i.e. two days before the filing date of the patent-in-suit. It is undisputed that this Linux kernel source tree is a set of source code files comprising SELinux and dm-verity.
74. SELinux is a mandatory access control (MAC) system implemented by using Linux Security Modules (LSM) in the Linux kernel. The LSM framework identifies various kernel objects, and sensitive actions performed on them. At the point at which each of these actions would be performed, an LSM hook function is called to determine whether or not the action should be allowed based on the information for it stored in an opaque security object. In SELinux, MAC policies are centrally defined and enforced by the system. SELinux confines user programs and system servers to the minimum privilege they require to do their jobs, thereby reducing the potential damage from a compromised application. Thus, SELinux uses predefined policies that describe which interactions between processes, files, and system resources are permitted. Anything not explicitly allowed by this rule is automatically blocked, limiting the impact of security breaches.
75. dm-verity is a Linux kernel feature that provides transparent integrity checking of block devices, using cryptographic hash trees to ensure that data has not been tampered with. Unlike SELinux, dm-verity is not mounted on LSM. The mechanism builds a Merkle hash tree over the entire block device until a single root hash is computed. When the kernel reads a block through the mapped device, which occurs at least at boot-time, the target code ensures the integrity of the block by verifying its hash against the precomputed Merkle tree. If this check fails, the block is rejected (usually causing the I/O to fail with an error code).
76. However, the Linux kernel source tree is not a usable Linux kernel as such, but rather a long list of building blocks that are not always compatible. There is no indication that a running combination of SELinux and dm-verity was available before filing date of the patent-in-suit, or that a specific implementation containing both SELinux and dm-verity was directly and unambiguously disclosed in the available source tree. In particular, the Defendants have not demonstrated that building a Linux kernel from the source tree inevitably would result in a system in which dm-verity is implemented in such a way that it performs checks during

runtime, or that the validation by dm-verity in such a system would be a reaction to a request sent by the kernel to SELinux (being considered, together with dm-verity, as secured software agent) via a hook, i.e. an upcall within the meaning of features 1.2.1 and 1.2.2.

77. Accordingly, the novelty attack on claim 1 based on Prior Linux Version is not successful.

4.3.3.3 *Novelty of claim 1 in view of D1 (US 2009/0271844 A1)*

78. D1 is a US patent application (US 2009/0271844 A1) that relates to techniques for controlling access to accessible components of computing environments. The techniques can be used to provide Mandatory Access Control (MAC) mechanisms for mobile and embedded systems. One or more accessible resources which a component may attempt to access are determined so that one or more access permissions can be stored in a manner that they can be obtained if the component attempts to access the one or more accessible resources, thereby allowing access to the one or more accessible resources to be determined based on access permissions that are readily available. The access permissions can be stored in anticipation of need.

79. The object of the invention in D1 is to provide a safe (e.g. a trusted) access controlling (or monitoring) system which can control access to resources of a computing environment, in particular based on the permissions discussed supra. This is achieved by an access monitoring component provided in a secure and trusted operating environment utilizing MAC capabilities of a secure operating system like SELinux OS. Further aspects are in particular disclosed in and described with regard to Figure 3A.

80. It is undisputed that D1 discloses features 1. to 1.2.2 apart from feature 1.1.2 with regard to the location of the secured software agent. It is also undisputed that D1 discloses an integrity verification where the verification is split between Integrity Measurement Service 322 in the kernel and Integrity Verification Services 320 outside the kernel in the user space (see e.g. D1 Figure 3A).

81. D1 does not directly and unambiguously disclose a secured software agent that is embedded in an abstraction layer, as required by feature 1.1.2 of claim 1. When discussing claim construction (supra), the Court has concluded that, according to the patent, the secured software agent in its entirety, or at least the functionality responsible for validation, must reside in the same (identical) abstraction layer as the kernel. According to

D1, it is the Integrity Verification Service 320 that may verify integrity values measured by the Integrity Measurement Service 322, by checking them against a known value. Furthermore, the Integrity Verification Service 320 is the entity that, based on the result of the verification, decides to grant or deny (requested) access and that may generate inputs to the reference monitor (cf. D1, paragraphs [0077] and [0104]). Since the Integrity Verification Service 320 resides in user space, it is not directly and unambiguously disclosed that the responsibility for the validation and thus the secured software agent resides in the same layer in which the OS kernel resides (feature 1.1.2). Only the Integrity Measurement Service 322, which measures the current integrity of the client application, i.e. the value to be checked by the Integrity Verification Service 320 against the known value, resides in the kernel space and therefore in the same layer as the kernel.

82. Furthermore, D1 does not directly and unambiguously disclose that the Integrity Verification Service 320 has sole access to a secured data storage (feature 1.2.4, cf. paragraphs [0104] and [0077]). The fact that the Integrity Verification Services 320 may be in communication with an external entity, e.g. a trusted verification server (paragraph [0077]), does not disclose sole access to this external entity, even if it were to be considered as a secured data storage. Paragraph [0066], which contains a reference to sole access, relates to storage of access permissions used by the trusted access monitoring manager 302a and not to information used by the integrity service 320. In other words, it is not disclosed that the information used for integrity verification is stored in the secured storage 302b, at least not with sole access.

83. The fact that D1 discloses a trusted access monitoring manager 302a in the kernel that can determine whether access to an accessible resource should be granted or denied based on access permissions stored in a secure storage 302b in the kernel (see e.g. Figure 3A and paragraph [0071]) is not sufficient, since (only) such granting or denying of access without further validation by an element in the abstraction layer does not qualify as a validation in accordance with claim 1 (supra, mn. 58).

84. Accordingly, the novelty attack on claim 1 based on D1 is not successful.

4.3.3.4 *Novelty of claim 1 in view of D6 (“Linux Security Modules: General Security Support for the Linux Kernel”)*

85. D6 is an article from a conference held in 2002 discussing General Security Support for the Linux Kernel. It deals with strong security solutions to the access control mechanisms of existing mainstream OS focussing on SELinux as a flexible access control architecture in the Linux kernel. The Linux Security Module (LSM) is described as allowing modules to mediate access to kernel objects by placing hooks in the kernel code just ahead of the access. Before the kernel would have accessed an internal object, a hook makes a call to a function that the LSM module must provide. The module can either let the access occur, or deny access, forcing an error code return. LSM provides only the mechanism to enforce enhanced access control policies. It is the LSM modules that implement a specific policy and are critical in proving the functionality of the framework.

86. D6 is mentioned at the first page of the patent-in-suit and the Court has already concluded that SELinux does not qualify as a secure software agent that validates a request in accordance with feature 1.2.3 of claim 1. Again, merely granting access based on permission is not sufficient. In addition, D6 does not unambiguously disclose that SELinux has sole access to a secured data storage. Defendants argue that, according to Claimant’s infringement action, in particular Claimant’s exhibits HE 20 (concerning SELinux concepts) and HE13 (containing the White Paper of the drones), the LSM policy rules are stored in an opaque security object (cf. HE 20, p. 1) and, with regard to the attacked embodiments, in the root file system, protected by secure boot (cf. HE 13, p. 15 third paragraph). However, this does not necessarily reflect the situation at the relevant filing date of the patent-in-suit. Moreover, it is not always clear whether Defendants base their arguments on the disclosure of D6 or on the existing LSM framework including SELinux as such at the date of D6. Apart from that, the asserted circumstances do not directly and unambiguously imply that the secured software agent has sole access to storage.

87. Thus, claim 1 is novel in view of D6.

4.3.3.5 *Novelty of claims 2 to 14*

88. Since claim 1 is novel, the same applies for system claim 14, which is defined as carrying out the method of at least claim 1. All other claims are also new by virtue of their dependence on claim 1.

4.3.4 Inventive step

4.3.4.1 *Legal standard*

89. Pursuant to Article 65(2) UPCA and Article 138(1)(a) EPC, a European patent may be revoked if the subject-matter of the European patent is not patentable under Articles 52 to 57 EPC. To be patentable, the invention must, inter alia, involve an inventive step (Article 56 EPC). Article 56 EPC specifies that an invention shall be considered as involving an inventive step if, having regard to the state of the art, it is not obvious to a person skilled in the art.

90. Based on decisions by the Court of Appeal of 25 November 2025, in cases CoA_21/2025 et al. and CoA_528/2024 et al., Meril v. Edwards Lifesciences and Amgen v. Sanofi, the UPC approach to inventive step is as follows.

- The burden and presentation of proof with regard to the facts from which the lack of validity of the patent is derived and other circumstances favorable to the invalidity or revocation lies with the claimant in a revocation action (Art. 54 and 65(1) UPCA, R. 44(e)-(g), 25.1(b)-(d) RoP). Even though proof of certain facts, if contested, may be required, the assessment of the relevant facts and circumstances is a question of law.
- It first has to be established what the object of the invention is, i.e. the objective problem. This must be assessed from the perspective of the person skilled in the art, with their common general knowledge, as at the application or priority date (also referred to as the effective date) of the patent. This must be done by establishing what the invention adds to the state of the art, not by looking at the individual features of the claim, but by comparing the claim as a whole in the context of the specification and the drawings, thus also considering the inventive concept underlying the invention (the technical teaching), which must be based on the technical effect(s) that the person skilled in the art, on the basis of the application, understands is (are) achieved with the claimed invention.
- In order to avoid hindsight, the objective problem should not contain pointers to the claimed solution.
- The claimed solution is obvious when at the effective date the person skilled in the art, starting from a realistic starting point in the state of the art in the relevant field of

technology and wishing to solve the objective problem, would (and not only “could”) have arrived at the claimed solution.

- The relevant field of technology is the specific field relevant to the objective problem to be solved as well as any field in which the same or similar problem arises and of which the person skilled in the art of the specific field must be expected to be aware.
- A starting point is realistic if the teaching thereof would have been of interest to a person skilled in the art who, at the effective date, wishes to solve the objective problem. This may for instance be the case if the relevant piece of prior art already discloses several features similar to those relevant to the invention as claimed and/or addresses the same or a similar underlying problem as that of the claimed invention. There can be more than one realistic starting point, and the claimed invention must be inventive starting from each of them.
- The person skilled in the art has no inventive skills and no imagination and requires a pointer or motivation that, starting from a realistic starting point, directs them to implement a next step in the direction of the claimed invention. As a general rule, a claimed solution must be considered not inventive/obvious when the person skilled in the art would take the next step, prompted by the pointer or as a matter of routine, and arrive at the claimed invention.
- For an inventive step to be present, it is not necessary to show improvement of the technical teaching as defined by the patent claims over the prior art. Inventive step may also be found if the patent claims disclose a non-obvious alternative to solutions known in the prior art.

4.3.4.2 The objective technical problem

91. The objective technical problem of the invention, which happens to be in accordance with paragraph [0025] of the patent-in-suit, is to provide an improved security mechanism for preventing unauthorized use of a device and digital assets on that device that overcomes some of the problems mentioned in paras. [0011] to [0024] of the patent-in-suit (cf. supra chapter 4.2.1) and the limitations of static platform security.

4.3.4.3 *The Prior Linux version*

92. The Defendants initially argued lack of inventive step starting from the Linux kernel source tree that already has been discussed in relation to novelty (chapter 4.3.3.2). When the Claimant objected that this was not a usable Linux kernel that can actually be executed on a computer, and that the actual Linux version that were available at that time (Linux 3.3) did not contain SELinux and dm-verity in combination, the Defendants added inter alia arguments on lack of inventive step based on Linux 3.3 as a starting point. It is, as will be seen below, irrelevant which of these “Prior Linux versions” is used as the starting point for assessing inventive step.
93. The Linux kernel source tree is, as mentioned above, not a usable Linux kernel as such, but rather a long list of building blocks that are not always compatible. However, these building blocks (source code files) comprise SELinux and dm-verity, which are described supra (chapter 4.3.3.2).
94. Linux 3.3 is a usable Linux kernel version released on 18 March 2012, i.e. a few days before filing date of the patent-in-suit. It is an open-source, Unix-like operating system (OS) kernel as such being the core component of that operating system. It comprises SELinux activated as a mandatory access control system and implemented as an LSM, but it does not include dm-verity.
95. The Defendants argue that should the Court find that the Linux kernel source tree does not disclose all features of claims 1-5, 10-12 and 14, then the asserted claims would at least lack an inventive step in view of the source tree and/or Linux 3.3. In particular, the difference between the subject-matter of claim 1 and Linux 3.3 is the validation of the request (features 1.2.3 and 1.2.4). The person skilled in the art, faced with the objective technical problem (even if it is narrowed to providing an improved security mechanism for Linux kernel preventing unauthorized access to it) and starting from Linux 3.3 and/or the Linux kernel source tree, would – according to the Defendants – have been prompted to build a Linux kernel comprising SELinux (natively) and dm-verity, e.g. by adding dm-verity to an existing and usable Linux kernel that included SELinux, using his/her common general knowledge.

96. However, even if we assume that the skilled person would have built a Linux kernel comprising SELinux and dm-verity, the Defendants have at least not demonstrated that the skilled person when being confronted with the objective problem would have implemented dm-verity in a way that it operates also during runtime (cf. mn. 57). Nor have they shown that the skilled person would have arrived at a system where any validation by dm-verity would be a reaction to a request sent by the kernel to the secured software agent via an upcall during runtime, or that the skilled person would have an incentive to do so.

97. Thus, there is no evidence that the possible combination of SELinux and dm-verity would have resulted in the method described in claim 1. Claim 1 is therefore inventive in view of the Prior Linux version(s).

4.3.4.4 *D1 (US 2009/0271844 A1), alone or in combination with D2 (US 2005/0005101 A1) or D3 (US 2011/0047542 A1) or D4 (US 2009/0007233 A1)*

98. D1 is a US patent application (US 2009/0271844 A1) that relates to techniques for controlling access to accessible components of computing environments. It has been described above when discussing novelty. On novelty, the Court concluded that D1 does not disclose that the elements considered as secured software agent are embedded in an abstraction layer, as required by feature 1.1.2 of claim 1, or that the secured software agent has sole access to a secured data storage where the information used for integrity verification is stored (feature 1.2.4).

99. The Defendants have not presented persuasive arguments that the person skilled in the art, starting from D1 and faced with the objective problem of providing an improved security mechanism for preventing unauthorized use of a device and digital assets on that device without the limitations of static platform security, would have been motivated to modify D1 by moving the Integrity Verification Services (320) from the user space to the abstraction layer that includes the kernel (see D1, Figure 3A) and performing integrity verification (being a form of validation) there as claimed in the patent-in-suit (feature 1.1.2). Moreover, it is not apparent why the skilled person, starting from D1, would foresee that the Integrity Verification Service 320 of D1 has sole access to a secured data storage used for verification (feature 1.2.4). In particular, the cryptographic context of D1 does not prompt the person skilled in the art to implement a secured data storage of that kind.

100. D2 (US 2005/0005101 A1) is a US patent application that relates to a system for verifying signature information of kernel modules as these are loaded into the kernel. D2 discloses a kernel module signature verification system (KMSVM) 350, which is included in the kernel memory level 340 and which automatically monitors kernel module signature path and extracts and verifies the signature information provided by each module attempting to load to the kernel. To this end, the KMSVM 350 may comprise a kernel cryptographic framework (KCF) and kernel cryptographic framework daemon (KCFD) for processing kernel cryptographic module signatures to authenticate and verify the kernel module's ability to be loaded into the kernel memory 340. The signature information retrieved for the kernel module path is provided by the KCFD to verify the signature information provided by the KCF when the same kernel module attempts to register the signature information with the KCF (cf. D2, paragraphs [0035] et seq.). In this context, the KCFD helps in retrieving signature verification information from a storage device in the computer system every time a cryptographic module attempts to request to be loaded into the kernel 340. The KCFD may also do module verification for kernel cryptographic modules (cf. D2 paragraphs [0038] and [0039]).

101. D2 at least does not disclose the use of upcalls/hooks (feature 1.2.1) and that the KCFD has sole access to the secured data storage (feature 1.2.4) used for verification (cf. D2 paragraphs [0038] and [0039]). In particular, the cryptographic context of D2 does not directly and unambiguously disclose to the skilled person in an implicit way that the KCFD has sole access to the aforementioned storage device in the computer system.

102. The integrity verification in D1 is performed at runtime while it in D2 is performed at boot time (cf. D2, paragraphs [0041] and [0045], Figure 6). The Defendants have not demonstrated why a person skilled in the art would combine D1 and D2, despite this difference and the further difference that D2 does not disclose the use of upcalls/hooks in this context. The fact that the Integrity Measurement Service of D1 may additionally perform measurements during boot time yields no other result. These measurements concern Mandatory Access Control (MAC) policies (cf. D1, paragraph [0103]) and have nothing to do with D1's integrity verification process with regard to application programs during runtime upon a request for access to a target object as discussed supra (chapter 4.3.3.3) on novelty. Moreover, as discussed supra, D2 does not disclose at least feature 1.2.4 of claim 1 in the patent-in-suit, i.e. that the storage device is only accessed by the

kernel cryptographic framework daemon. The cryptographic context of D2 does not prompt the person skilled in the art to restrict access to the storage to the KCFD or the KCFD and any other component of the KMSVM involved in the verification of the signature. Consequently, a combination of D1 and D2 still would lack feature 1.2.4 with regard to sole access by a secured software agent to a secure data storage used for verification.

103. D3 (US 2011/0047542 A1) is a US patent application that describes a method for enforcing security policies in a virtual environment and the use of a virtualized OS together with client OS. It requires a hypervisor on top of running an OS to achieve security. The hypervisor, i.e. the basic abstraction layer of software that sits directly on the hardware below operating systems, abstracts the hardware for the virtual machines and controls the execution of virtual machines as they share the common processing environment (D3 paragraph [0023]). A security layer within the kernel space may provide a memory element (e.g., a database) as inventory 24 (cf. D3 Figure 1) that keeps information about authorized binaries and (loadable) kernel modules which got pushed there from a user space inventory 36 (cf. D3 paragraph [0024]). In particular, this inventory 24 can be implemented as checksum-based inventory (LKM) 52 (cf. D3 Figure 3, paragraph [0034]). The kernel space also comprises a system call table 28 that is configured to intercept loading of modules and binaries in a given computer, and a loading routines module 30 that provides functionality for hooked routines, which can perform a check on each kernel module and binary (D3 Figures 1 and 3, paragraph [0024]). Both instances of inventory 24 in kernel space and 36 in user space may be updated, and/or suitably modified based on systematic configurations, or events that trigger a change to either of these inventories. A control module 34 offers control for the security layer in user space (cf. D3 paragraph [0024], Figures 1 and 3). For example, when a request to run a binary or (loadable) kernel module (LKM) is intercepted, it is checked to see whether it is authorized. To this end, the security layer or security agent (e.g., a piece of software running in Domain 0, i.e. in the inner ring with highest privilege that hosts the virtual machine monitor/hypervisor, cf. Figure 2) computes checksums and compares the result against the values stored in the inventory 52 in order to validate whether a given element (i.e., a binary or a kernel module) is authorized for the particular system, and to discern whether to allow these elements to be executed (cf. D3 paragraphs [0025], [0027] and [0043]).

104. It can be left open whether the use case of D3, i.e. enhancing security in a virtual machine environment by an instance residing in the privileged domain of the hypervisor, is too far away from the one of D1, so that D3 would not be considered by a person skilled in the art for solving the objective problem already mentioned. D3 does not directly and unambiguously disclose that the security agent or the security layer (in particular the loading routines module 30) has sole access to the inventory 24 (or its checksum based special embodiment 52) that stores information the validation/verification is based on (cf. D3, paragraph [0024], [0034] and [0043]). D3 even explicitly states that the general inventory 24 that keeps information about authorized binaries and kernel modules can be modified (paragraph [0024]). Against this backdrop, it is not clear that a combination of D1 (which does not disclose this feature either) and D3 would result in the secured software agent (Integrity Verification Services (320) of D1 and security agent/layer of D3, respectively) having sole access to a secured data storage, as required by feature 1.2.4 in claim 1 of the patent-in-suit (cf. paragraph [0024] in D3). As with D1, the cryptographic context of D3 does not prompt the person skilled in the art to restrict access to the inventories to the secure agent.

105. D4 (US 2009/0007233 A1) is a US patent application that describes a defence against root (DAR) system and method for securing sensitive material on a computer system comprising a network of computers from unauthorized access by a root level user of the computer system. Referring to Figures 1 and 2, the DAR system 100 modifies operating system 105/204 of each computer 104 to provide a screening layer 206 to screen out dangerous system functions directed at sensitive material 201, while the operating system 105 interacts with both protected sensitive material 201 and unprotected material 202 through primitive (e.g. kernel level) system functions 204. Critical system functions that are potential security holes include, e.g., Linux functions such as ptrace (debugging), insmod and remmod (inserting and removing kernel modules), reboot (restarting the machine), etc. Accordingly, in various embodiments of the DAR system 100, screen 206 operates to deny access to these functions to root (or other users) on a given computer 104 while a sensitive material is present (e.g. when a protected application is running) (cf. D4 paragraphs [0052] and [0053], Figure 2).

106. D4 at least does not disclose a secured data storage with sole access by the screening layer 206 (feature 1.2.4) and that the screening layer being considered as secured software agent resides in the same abstraction layer as the OS kernel (feature 1.1.2). Defendants did not show a secured data storage in D4, let alone with sole access by the screening layer 206. Furthermore, Defendants did not demonstrate that feature 1.1.2 is directly and unambiguously disclosed. The modification of the OS addressed in D4 (cf. e.g. paragraphs [0052] and [0057]) relates to installing a screening layer 206 between the OS and the access to sensitive material in order to screen out dangerous system functions 204 directed at sensitive material 201 (as depicted in Figure 2 and described in paragraph [0053] of D4) and, as the case may be, to modifying the system functions 204 accordingly. However, Defendants did not show that modifying the OS and the critical system functions 204 such as debugging also includes locating the screening layer 206 in its entirety, or at least in its essential functions needed to determine access (see mn. 50), directly in an abstraction layer that comprises the OS kernel. Rather, while paragraph [0052] seems to address the system functions 204 also as operating system (that is to be modified), Figure 2 gives the impression that the screening layer resides in an interface layer that is located outside the layer of OS 15 and contains the system (call) functions 204. The fact that, instead of directly modifying the kernels of the network's computers, the DAR system may also use (loadable) kernel modules to modify the system functions 204 by adding security functionalities (cf. D4 paragraph [0067]) does not yield another result. It is still not directly and unambiguously disclosed that the loadable kernel modules comprise the screening layer in its entirety or at least in its essential functions needed to determine access.

107. The Defendants rely on the combination of D1 and D4 for the embodiments described by a combination of claim 1 with dependent claims 10–12 of the patent-in-suit, i.e. debugging. However, the Defendants have failed to show that D4 (or D1, see mn. 98) would prompt the person skilled in the art to have a secured software agent in an abstraction layer that includes the kernel that performs integrity verification in this layer blocking a debug utility request based on information stored in a secured data storage the secured software agent has sole access to.

108. Thus, claim 1 is inventive in view of the attacks starting from D1.

4.3.4.5 D5 (US 7,644,271 B1) in combination with D1

109. D5 (US 7,644,271 B1) is a US patent that presents a method for providing enforcement of security policies when some kernel modules are to be loaded. The paths to attain shared library executable files by user processes are cached. When a request to load a Loadable Kernel Module (LKM, referred to as kernel loadable module (KLM) in D5) is received, a previously cached path for said LKM is retrieved, said path mapping a location of an executable file from which said LKM was produced. A security policy is applied to said path, wherein when said path triggers a security policy rule then an action associated with a triggered rule is taken. In this context, D5 discloses a Rule Engine 18 that resides in kernel space (D5 Figure 1) and performs the determination of the file path and applies the security policy to said file path (D5 Col. 5 lines 29 to 61). It is further disclosed that the implementation can also be extended with hash/signature-matching. To this end, the Rule Engine 18 can request the user-level agent process (also referred to as the Local Event Manager) to calculate and send the hash of the module executable file and compare it with the hash of the memory buffer in the create module system call (D5 Col. 5 lines 62 to 67). The Local Event Manager 22 resides in user space (D5 Figure 1).

110. It is undisputed that D5 does not disclose the “sole access to the secured data storage” in feature 1.2.4 of the patent-in-suit. Given the claim construction described above, i.e. that features 1.1.1 and 1.1.2 require that the secured software agent in its entirety, or at least the functionality responsible for validation, resides in the same (identical) abstraction layer that includes the kernel, the Defendants have also failed to demonstrate that D5 disclose these features. Rather, the calculation of the hash value of the module executable file which must be compared with the good value is performed by the Local Event Manager (D5 Col. 5 lines 63 to 67). Even though the comparison might be carried out by the Rule Engine 18, the calculation of the hash value constitutes an essential function in the context of integrity verification (see mn. 51). Since the Local Event Manager is a user-level process and resides in user space (D 5 Col. 5 lines 63 to 64, Figure 1), it is not directly and unambiguously disclosed that the Local Event Manager resides in the same abstraction layer as the OS kernel. Therefore, D5 does not disclose a secured software agent embedded in the abstraction layer (feature 1.1.2).

111. Defendants did not demonstrate that, by virtue of D5, the person skilled in the art, when starting from D5, would be prompted to foresee a sole access of the Rule Engine 18 to a secured data storage containing the good hash value and to move the calculation of the hash value of the LKM to be loaded into the same abstraction layer in which the OS kernel resides.

112. The Court has already concluded that the same applies to D1. A combination of D5 and D1 therefore still would lack features 1.2.4 and 1.1.2.

113. Thus, claim 1 is inventive in view of the combination of D5 and D1.

4.3.4.6 D6 ("Linux Security Modules: General Security Support for the Linux Kernel") alone or in combination with D2 (US 2005/0005101 A1) or D3 (US 2011/0047542 A1) or D4 (US 2009/0007233 A1) or D5 (US 7,644,271 B1)

114. D6 is the article from a conference held in 2002 discussing General Security Support for the Linux Kernel. It has been described above when discussing novelty. On novelty, the Court concluded that D6 does at least not disclose a secured data storage, or any sole access to such a storage by the secured software agent. The Court has also, when discussing the inventive step attacks starting from D1 and D5 respectively, concluded that these missing features cannot be found in and are not suggested by D2, D3, D4 or D5. Defendants did not demonstrate that D6 would suggest a secured data storage with sole access by the secured software agent.

115. Thus, also the inventive step attacks based on D6 alone or in combination with D2, D3, D4 or D5 must fail. Claim 1 involves an inventive step in view of these combinations.

4.3.4.7 Inventive step of claims 2 to 14

116. Since the attacks against an inventive step regarding claim 1 are not successful, the same applies for system claim 14. Claim 14 is defined as carrying out the method of at least claim 1. All other claims also involve an inventive step by virtue of their dependence on claim 1. Apart from that, Defendants did not put forward any argument regarding lack of inventive step in relation to claims 6 to 9 and 13.

4.3.5 Conclusion on validity

117. Since the patent-in-suit is valid, the counterclaim for revocation shall be dismissed.

4.4 Infringement

118. The Claimant asserts that the drones as well as the remote controllers make use of the method defined in claims 1-5 and 10-12 of the patent-in-suit and that the drones as well as the remote controllers are systems as defined in claim 14 of the patent-in-suit, as follows:

- The independent method claim 1 in combination with dependent claims 3 and 4 is asserted together with the dependent claims 2 and 5 by means of the link 'in particular'.
- Furthermore, independent system claim 14 with a reference to claims 1, 3 and 4 is asserted together with the dependent claims 2 and 5 by means of the link 'in particular'.
- Furthermore, the independent method claim 1 in combination with dependent claim 10 is asserted together with the dependent claims 11 and 12 by means of the link 'in particular'.
- Furthermore, independent system claim 14 with a reference to claims 1 and 10 is asserted together with the dependent claims 11 and 12 by means of the link 'in particular'.

119. It follows from the Court's claim construction that a prerequisite for the attacked embodiments to infringe the patent-in-suit is that they have a secured software agent, that is able

- to receive from the operating system kernel via an upcall during runtime, a request to an internal operating system kernel object, and,
- as a reaction to this request, to determine whether this request is valid based on information located in a secured data storage it has sole access to, and
- to limiting access to the abstraction layer based on the aforementioned determining step.

120. It is undisputed that dm-verity is not mounted on the LSM through hooks/upcalls like SELinux, and that any validation by dm-verity instead takes place when data protected by dm-verity is read from disk.

121. Even more important, the Claimant have failed to demonstrate that dm-verity, as implemented in the drones and their controllers, performs any validation during runtime, which is required by the patent-in-suit and disputed by the Defendants. In this respect, Claimant's specific statements on dm-verity in its Statement of claim are strictly limited to booting (sections 6.1.6 and 6.1.7 on claim 1, in particular para. 156, section 6.1.9 on claim 3, in particular para. 177, section 6.1.10 on claim 4, in particular paras. 184, 185). Apart from that, the Claimant relies on vague statements (cf. SoC para. 154, Reply paras. 315, 322) and some alleged evidence on how dm-verity normally works in Android. At the same time, the Drone Security White Paper from DJI, which the Claimant relies on, explicitly states (cf. exhibit HE13, p. 15):

DM-VERITY

DM-Verity is the technology used to protect the partition integrity in Android, employing a hash tree structure to map the data of the entire system partition. The hash tree will be signed. During the boot process, the public key is used to verify the integrity of the hash tree. Next, the applications and the software library are read from the system partition and compared against the hash tree. Any failure during verification will result in the termination of the boot process.

The system partition contains many system-level applications and software libraries, providing basic functions such as flight, navigation, image transmission, and data storage for the drone. Tampering of the system partition indicates malicious damage of the drone's basic functions, which affects the system security and data security of the drone. Therefore, DM-Verity plays a major role in ensuring the security of users' property and data. See the Appendix for models that support DM-Verity.

122. Thus, the White Paper describes dm-verity as a protection used at boot time and there is no indication in the White Paper that dm-verity has been configured to perform any validation also at runtime. Nor are there any other sufficient fact and evidence for concluding that this is the case. Rather, it seems very natural that, in the attacked embodiments, the system-level applications and software libraries that are contained in the system partition protected by dm-verity and that, according to the above quotation from the White Paper, provide the device-specific basic functions of a drone are already loaded during boot time, simply because they provide basic drone functions required by the attacked embodiments, even though these functions may not form basic functions in other android-based operating systems, e.g. in a smartphone, and may therefore form in these other operating systems loadable kernel modules that are loaded only on demand upon runtime. For this reason, it is not decisive for the case at hand that, according to the documentation on kernel modules for an android operating system in general (exhibit HE 23), loadable kernel modules may be loaded during runtime, and may therefore be checked by dm-verity at runtime when their storage location in the system partition protected by

dm-verity is accessed to load them to the kernel upon demand. According to exhibit HE 23, there are different types of loadable kernel modules that, depending on the type and the configuration, are loaded during boot time or at run time. In particular, there are hardware-specific kernel modules that might have to be already loaded during booting (cf. exhibit HE 23 p. 3 last paragraph). In the present case, Claimant has not stated any specific fact that a specific system-level application or software library that is located in the partition protected by dm-verity in the attacked embodiments is loaded during runtime. On the contrary, the loadable kernel modules listed in para. 178 of the Statement of claim are, according to Claimant, contained in the folder “/lib/modules” of the exemplary Mavic Pro 3 drone (cf. SoC para. 178). This folder belongs to the vendor-boot partition that is already loaded during boot time (cf. exhibit HE 21, p. 1 first paragraph on the page and first paragraph under “Module location”, and exhibit HE 23, p. 3 last paragraph in section “Vendor modules”), as explicitly acknowledged by Claimant at the end of paragraph 177 of its Statement of claim. There is no single fact put forward or otherwise apparent that, in the attacked embodiments, dm-verity operates (be it in the partition or in other memory) on a system-level application or software library protected by it after they have been successfully loaded during booting.

123. Nothing else follows from the expert opinion submitted by the Claimant (exhibit HE 28). The written opinion deals with SELinux and dm-verity only in a general manner and does not relate to the specific implementation in the attacked embodiments.

124. Consequently, the Claimant has failed to substantiate, let alone to prove that the drones and/or their controllers make use of any of the asserted methods or systems. Therefore, the infringement action shall be dismissed.

4.5 Legal costs and other expenses

125. According to Article 69 UPCA, reasonable and proportionate legal costs and other expenses incurred by the successful party shall, as a general rule, be borne by the unsuccessful party, unless equity requires otherwise, up to a ceiling set in accordance with the Rules of Procedure. Where a party succeeds only in part or in exceptional circumstances, the Court may order that costs be apportioned equitably or that the parties bear their own costs. A party should bear any unnecessary costs it has caused the Court or another party.

126. According to Rule 118.5 RoP, the Court shall – in the decision on the merits – decide in principle on the obligation to bear legal costs in accordance with Article 69 UPCA. This cost decision may, according to Rule 150 RoP et seqq., be the subject of separate proceedings following a decision on the merits and, if applicable, a decision for the determination of damages.

127. In this case, the parties have agreed to a "flat amount" as reimbursable costs of the total proceedings (infringement and counterclaim). It is the parties' common understanding that such "flat amount" arrangement means that the court would build one percentage quota based on the win/loss ratio in the infringement and counterclaim for revocation proceedings. The "flat amount" agreed between the parties will then be split in accordance with this quota and unless it is a 50/50 decision, the difference will be paid by one party to the other.

128. The outcome of these proceedings is that the Claimant is the unsuccessful party in the infringement action, while the Defendants are the unsuccessful parties in the counterclaim. The value of the counterclaim has been set higher than the value of the infringement action, but this is mainly because the patent also has a value outside these proceedings (e.g. in relation to the still ongoing proceedings against former Defendant 1). This higher value of the counterclaim does not automatically lead to the conclusion that the party who was successful in those proceedings won more than 50 percent of the total proceedings.

129. In this case, the Court finds it appropriate to order that each party shall bear their own costs. Consequently, there is no reason for granting an interim award on costs.

5. DECISION

- I) The infringement action is dismissed.
- II) The counterclaim for revocation is dismissed.
- III) Each party shall bear their own costs,

Delivered in Mannheim on 12 June 2026

NAMES AND SIGNATURES

Böttcher Legally qualified judge	Dirk Andreas Böttcher Digital unterschrieben von Dirk Andreas Böttcher Datum: 2026.06.11 12:37:22 +02'00'
Johansson Legally qualified judge	Stefan Erik Johansson Digitally signed by Stefan Erik Johansson Date: 2026.06.11 11:08:03 +02'00'
Sender Legally qualified judge	Tobias Sender Digital unterschrieben von Tobias Sender Datum: 2026.06.11 11:59:51 +02'00'
Jauregui Urbahn Technically qualified judge	Signé numériquement par: Cristian JAUREGUI URBANH Organisation: VAAEY Unité d'identification: 0002 552030967 Limites d'utilisation: Exempt Text ETSI EN 319 411-2 compliant certificates for natural persons (QCP-N-QSCD) : Signature and Authentication (Exempt Text ETSI EN 319 411-2 compliant certificates for natural persons (QCP-N-QSCD) : Authentication and Signature Date: 11/06/2026 11:43:40
For the Sub-Registrar: Kranz, Clerk LD Mannheim	Digitally signed Kranz Andreas Michael 2026-06-11 16:40:02 +0200 

Information about appeal

An appeal against the present Decision may be lodged at the Court of Appeal, by any party which has been unsuccessful, in whole or in part, in its submissions, within two months of the date of its notification (Art. 73(1) UPCA, R. 220.1(a), 224.1(a) RoP).

Information about enforcement

This decision has no enforceable effect.