On the Necessity of Measuring Security in IoT

Tobias Eggendorfer
TH Ingolstadt
Faculty of Computer Science
Ingolstadt, Germany
Email: tobias.eggendorfer@thi.de

Katja Andresen
HWR Berlin
Department of Business and Economics
Berlin, Germany
Email: katja.andresen@hwr-berlin.de

***Abstract*—While the Internet of things has become ubiquitous, it is mostly populated by former embedded devices, whose software was developed by specialists in the respective field. Hence, security researchers often find issues in those, which some consider to be low hanging fruits. Fixing security flaws in deployed embedded devices is sometimes very complex: In automotive or airborne systems, road- or airworthiness tests need to be passed, in these and e.g. medical devices or industrial Internet of things updates cannot interrupt operation. Therefore, in ideal world, Internet of things and embedded systems would be free from security issues. This could be reached with security metrics, a concept the authors are working on.**

Keywords-Internet of Things; IoT; Embedded Systems; Embedded Security; IoT Security; Security Metrics; Cyber Security; Industrial IoT; Legal Aspects

I. Introduction

In 2017, the United Stated Federal Drug Administration (FDA) recalled St. Jude's pacemakers due to a security issue [1], allowing potentially lethal remote tampering – an issue pre-seen by Holt and Holt's thriller "Flimmer" [2]. In Finland, attackers took down a building complex central heating unit, resulting in frozen pipes and massive restoration costs [3]. The industrial Internet of Things (IoT) controller *SIMATIC* by Siemens has 336 Common Vulnerabilities and Exposures (CVE) entries, 33 of these were issued in 2024 alone [4]. Attacking these might disrupt production processes and even challenge critical infrastructures. Cordless screwdrivers, ovens or coffee appliances may now be configured or used over the Internet, and thus have become a part of the Internet of things – as with the now legendary Miele dishwasher directory traversal [5]. Stealing cars is not short circuiting the ignition any more but gaining access to its Controller Area Network (CAN) bus and pretending by software the key was in the lock [6].

A. IoT and embedded security – Fixing issues

While software security issues are common nowadays, those in IoT and embedded systems are especially nasty: patching and updating the devices is often a tedious process. E.g., for a dash cam, firmware needs to be downloaded on the desktop, copied to a freshly formatted SD card, the dash cam rebooted and the card quick enough removed for the next reboot [7]. The complexity results from the device being offline or not providing an online-update feature.

For some IoT, updating is easier, because devices could download their update. However, there are two issues: On several occasions, updates broke the system, e.g., [8] [9]. Also, the update cannot be installed anytime, an example are cars on ferries updating their system without asking their user. With an update taking longer than the ferry trip, the car cannot be started again and thus blocks unloading the ferry [10]. This could happen to, e.g., a fire truck or an ambulance – those need to be available 24/7, at unpredictable times. Updates would need to be carefully planned with replacement vehicles being set up, as it is for regular maintenance. However, over the air software updates might occur more often, since from a manufacturers point of view, they are easily distributed. This puts an additional burden to an already very resource- and cost-intensive process, affecting update speed and thereby security.

Some devices need to run an approved version of their respective software, e.g., devices deployed in planes need an airworthiness approval, sometimes even by several agencies worldwide. In medical devices, similar rules exist. In neither case a mid air update is recommendable.

Some embedded devices are deployed without reasonable physical access, hindering updates, e.g., pacemakers and internal defibrillators. Some are hardly ever serviced, shifting the task of updating to its user, who is unaware of relevant security patches. In some cases, administrators might even be too careless to install security fixes: Both Hafnium and Heartbleed security issues were observed in the wild years after their respective discovery.

B. Lateral movement

In recent attacks, IoT served as the entry point to corporate networks, moving on from there is called lateral movement. There are reports on using a surveillance web cam [11] or the photovoltaic system's web-interface. In a home environment a smart TV could be the vantage point.

It is often suggested to separate networks, moving IoT devices outside the productive environment. This is only a partial solution, e.g., the Mirai bot net abused infected IoT devices for distributed Denial of Service (dDoS) [12].

C. Motivation for IoT and embedded security metrics

Therefore embedded and IoT-devices should be as secure as possible when deployed, thereby reducing the need for patches. But their manufacturers would always claim that security is of paramount importance to them and their devices underwent rigorous testing. Considering the vast amount of CVE numbers assigned to embedded and IoT-devices this is not plausible. While printers are neither embedded nor IoT, *Hewlett-Packard*

provides a very good example for vendor security claims versus reality: On January 29, 2025 they released patches for a CVE issued in 2017 [13] – eight years later.

Due to the absence of security measures in IoT, there is an urgent need for a way to help identify and implement necessary security controls [14].

Search engines like shodan.io support this need. They help identifying IoT-devices and searching for specific devices, software versions or configurations with vulnerabilities.

The authors proposed a security metric [15] to identify a systems level of security as well as to be able to compare devices in order to make an informed procurement decission. This paper discusses how this metric could be applied to IoT and embedded devices in order to increase their security.

D. Structure of this paper

The paper is structured as follows: Section II first defines the concept of embedded systems, IoT and Industrial IoT (IIoT) for the purpose of this paper. It then outlines common security challenges in regular software, how and when they affect IoT systems and how to mitigate them. Based on this, this section provides an overview on software quality management with a focus on security and security metrics outside IoT.

Section III analyses how security metrics could be applied to IoT. Based on this, legal enforcment of the use of security metrics and its potential effects is discussed in Section IV. The last Section V provides a conclusion and gives an outlook on future research.

II. Background

The following section provides definitions and an overview on Information Technology (IT), IT-security, security issues and their respective origin as well as counter measures.

A. Embedded Devices, IoT, firm- and software

For this paper, an embedded device is defined as a computer with a specific purpose that is usually integrated into a cyber physical system, interacting with sensors, actors and / or has a specific user interface, but usually not a full keyboard or screen. Their software is optimized for the use case and not meant to be changed by the user, e.g., dashcams, defibrillators, smart TVs, heating controllers or smart locks, but also engine control systems or a lane assistants in cars.

An IoT device is an embedded system connected to the Internet and could either be configured, monitored or controlled remotely. IIoT is IoT monitoring and / or controlling industrial system, such as assembly lines or automated warehouses.

For this paper, software and firmware are considered to be equivalent: While firmware is usually provided “on a chip”, which might even be a Read-Only Memory (ROM) module, and comes with a device, from a security perspective it is still a program. With the line between embedded systems and regular computers becoming less clear – a Raspberry Pi could be both – the borderline between soft- and firmware also moves. It becomes relevant, when it comes to potential cyber-physical effects of IoT.

B. IT security in general

Often social engineering (SE) and ransomware are seen as the biggest threads to IT security. From an operational perspective, this explains how the attack has been executed [16] and describes its effects. But technically, each SE attack is a security issue, be it bypassing the need to know or least privileges principle, a flaw in the system, which allowing to inject and execute arbitrary code, or other imperfections. Hence, while SE was supportive, the relevant issue was a faulty system.

For the effect, it is not relevant how the attacker exploits the newly gained access, e.g., to deploy ransomware or part taking in industrial espionage. However, the effects are relevant from an economic and political perspective, as well as for attribution and legal measures; however, this falls outside the technical scope. There, it is only a payload to the actual attack.

The difference between payload, the actual attack, e.g., a buffer overflow or format string issue, and the attack vector, describing how the attack was launched, e.g., through Remote Procedure Call (RPC) or SE, is important: Security measures addressing only the attack vector, e.g., by educating users to prevent SE, do not fix the underlying security flaw. It could still be leveraged through other means.

Security measures could mitigate some attack’s effects, e.g., off site backups to prevent ransomware. This reduces the risk, because the damage is lowered. However, the underlying security issue remains unsolved and could still be exploited for activities such as espionage. Hence, the best solution is to fix the cause, the security issue itself, since this is effective for all attack vectors and effects. Only addressing some of them is sometimes called “snake oil” or “security theatre” in the security community, due to their limited effect [17] [18].

C. Typical security issues in IoT and embedded devices

In IoT and embedded devices, there is a huge variety of tasks to perform. While in a Tesla, the systems provide X for the user interface [19] [20], in implanted pace makers a touch screen is not feasible, instead a Bluetooth Low Energy (BLE)-interface could be provided – these having their own security issues, e.g., [21]. Industrial IoT systems often use proprietary protocols to interface with control units, e.g., Supervisory Control and Data Acquisition (SCADA), others have built in web-interfaces, such as the aforementioned dishwasher.

Based on the huge variety of IoT and embedded systems, all security issues in both compiled software as well as those in web applications could occur. For the latter, Open Web Application Security Project (OWASP) provides an overview of potential issues. A regularly updated top ten list indicates the most prevalent, ranging from Cross Site Scripting (XSS) to Lightweight Directory Access Protocol (LDAP)- and Structured Query Language (SQL)-injections, but also including issues like insecure deserialisation of a data stream or authentication bypass [22]. While the list is updated every few years, the issues typically shift positions within the top ten, but rarely disappear entirerly – nor do new issues come up. While at the time of writing, the OWASP Top Ten 2025 list was not yet available, a comparison of the lists from 2017 to

2021 reveals that only three new issues appeared in the top ten. Out of those three that seem to have disappeared, four were merged into two new categories and are still in the top ten, these are: A01:2017 “Injection” and A07:2017 “XSS” merged into A03:2021 “Injection”, A04:2017 “XML External Entities” and A06:2017 “Security Misconfiguration” became A05:2021 “Security misconfiguration” [22]. This indicates bad security practice and a non existent web-security learning curve, applicable as well to IoT-web-interfaces.

In regular programs, security issues could be related to memory access, such as with a buffer overflow, out of bounds read or write, format string issue or off by one. These usually result in a program flow alteration or even code injection. Other issues include integer overflows, which might result in a system with an unstable state [23] or as a vantage point for other attacks, such as a buffer overflow [24]. All of which are applicable to IoT as well. Other issues are resemble web attacks, such as an under protected Application Programming Interface (API) or insecure authentication.

These issues either result from insecure programming practice or design flaws. Examples for the former include using `strcpy` instead of `strncpy`. While `strcpy` does not limit its copying to a maximum amount of bytes, `strncpy` does. This prevents – if no other flaws are present, such as integer overflows or off by ones – some buffer overflows, and was introduced in ANSI C in 1989 [25], long after buffer overflows were first described in 1972 [26]. In 1996, the Phrack Magazine in the famous article “Smashing the Stack for Fun and Profit” [27] explained the security issue due to its prevalence in these days. Still now, buffer overflows are of the most often abused security issue. Good programming practice would enforce the use of safer functions. Again an indication of a very slow learning curve, albeit automatic code analysis tools are able to warn [28]. Which points to them not being (properly [29]) used too often.

Design flaws are harder to identify and come by. Whether one considers Central Processing Unit (CPU) microcode software or hardware, both Spectre and Meltdown [30]–[32] are good examples of design issues: They are a result of pipelining in modern CPU architecture. Heartbleed provides an example of a software security issue due to a design flaw: By providing redundant data and not comparing two values, an out of bounds read could be triggered [33], another example “ZUGFeRD”, a concept for eXtensible Markup Language (XML) and Portable Document Format (PDF) based e-invoices posing a security risk through redundant, not cross checked data [34] [35]. These logical issues are much harder to automatically detect, and are found in IoT as well.

D. Prevention of security issues

In theory preventing security issues based on bad programming style is well known [15] [36] [37] and easily achieved with static code analysis [28], code reviews, peer programming and other simple quality checking measures.

Identifying design flaws is possible through code reviews and peer programming, however this requires a qualified team. If this was easy, faulty standards such as introducing heartbeat functionality in TLS leading to the aforementioned Heartbleed issue, or the ZUGFeRD issues would not have happened.

E. Relation between software quality management and security

All measures undertaken to increase software quality, such as coding standards, static and dynamic code analysis, code reviews and peer programming have an effect on the amount of security relevant issues in a program. OpenBSD is an example: Due to its quality management, the system only had two remote exploitable security issues in its standard installation “in a heck of a long time” [38], i.e., 30 years.

Quality management will find security issues related to bad programming and some logic flaws. At the same time, it will increase the quality of the code created, since programmers know their code will be reviewed, they will obey the coding standards. If these include safe programming practices, they have an effect on security as well.

F. Measuring software quality

Software quality in general is a broad term, providing many perspectives. ISO 25010 provides eight quality dimensions for software, starting from functionality via portability or usability, but also security. However, in this regard security is restricted to functionality. To the authors, software quality serves as a proxy for security: Quality means the absence of flaws, each flaw could result in a security issue [15] [37].

1) Standards: Standards like the ISO 27000 series only provide a management perspective, but offer little in terms of an operational approach to achieve a high level of software quality, and they cannot measure the quality of the code. To some extent, this aligns with how software quality measurement is discussed in research: So far “software quality” itself lacks a common understanding [15].

2) Formal Verification: Other concepts include more formal specification of software and deriving from this specification a verification. One example are Hoare logics [39]. However for most projects this manual process is cumbersome and very intense. There it is probably only used in environments with high quality and especially safety requirements, such as in space missions. But even then, if the conditions are not updated properly to reflect changes elsewhere, issues could still occur, e.g., the Ariane 5 [23]. In IoT, especially when it comes to high cyber-physical risks, projects like seL4 provide formally verified code [40]–[42].

3) Current concepts: Closer to the market are concepts such as the German’s German Federal Office for Information Security (BSI) providing a “software quality seal” based on self assessment of the vendor and easily measurable items such as the time needed to fix issues [43]. The speed of fixing may indicate a learning curve, but is not inherently a quality measure for the software itself. Rather, it serves as an indicator how effective a vendor is in accepting, understanding and resolving reports. This is useful to understand how interested they are in preventing future attacks, it is a measure of the quality of maintenance. But it does not give any indication on whether

the bug could have been identified earlier in the development process, which is where software quality stems from. This "software quality seal" is therefore hardly an indication of what it claims to be.

4) Bill of Materials: Another approach to quality is to identify which third party libraries and software is used within a project, including the respective version. If an update becomes available for one of these, it should be easily possible to identify whether it needs to be installed. This idea has been adopted by European Union (EU) Cyber-Resilliance-Act (CRA) by introducing the Software Bill of Materials (SBoM). However a SBoM does not provide a quality measurement in itself and there is no clear path to derive a quality measure from it: Does a short SBoM indicate a higher level or lower level of quality? There are pros and cons for either way. Aside of this, an SBoM is useful in identifying relevant patches. As per [44] IoT projects use less dependencies.

5) Static measures: Khezemi et al. [36] suggest to measure code quality based on statically comparing code by looking at size, code complexity, cohesion, coupling, code readability, and maintainability. These measures have a different understanding of quality than this paper has, in that it does not address security measures.

6) Conclusion: Quality aspects of embedded systems and IoT should address both, hardware as well as software components [45].

G. Measuring software security

Eggendorfer and Andresen [15] provide a detailed overview on currently available methods to measure software security and concludes that there is no commonly accepted method. This applies even more for IoT. However, to achieve a high level of security that is comparable, it is important to capture aspects beyond already existing standards or initiatives related to software quality.

Other initiatives invest in preventive measure, like educating software developers in order to support secure software development [38] [46] [47]. To assess software a broadly understood security metric would offer orientation and guidance. For web applications [48] developed a concept, which however has some minor flaws [15].

Reckhaus [49] compares the invest into IT security to insurance fees to identify an economic value, i.e., metric, for security investments. While this does not assess the security as a quality measure, it helps evaluating security concepts from a economic perspective.

Wuttig [50] analyses IT security in medical IoT devices, albeit without applying a formal metric. His analysis however gives a good impression of minimum requirements that should be included in a formal security test, that could be part of a security metric. A security issue in patient monitors is an example of the relevance [51].

III. Application to embedded and IoT

IoT and embedded systems have different security implications and requirements than desktop software. The following section discusses effects of different development approaches and how this would affect the respective software quality.

A. Differences in software development

Software development for IoT and embedded systems is different to developing other programmes. Embedded systems are designed for special tasks. For that, embedded systems, e.g., laundromats, cameras, smart home equipment, use firmware (software) that directly accesses the hardware of the device to carry out the intended function. Hence, they do not follow the „one fits all" architecture as the John-von-Neumann approach.

Typical characteristics of these systems have effects on quality, as well as security: Many embedded systems need to process data in real time, i.e., they have an upper limit for computing time. Obvious examples include the Antilock Braking System (ABS) and airbag deployment in cars.

These systems are usually designed for a reliable long term usage – meaning limited maintenance opportunities as well as corrective updates by default.

As they run on special microprocessors and microcontrollers embedded systems use limited resources as computing power and memory – compared with traditional computers. The specific, often restricted hardware might also have an effect on code optimization.

B. Related Work

Corno et al. [44] discuss this in detail for Open Source software, based on an analysis of source code of 30 IoT and 30 non-IoT projects published on github. This is partly because IoT projects are more complex, in that applications need to be fault tolerant, often need to use sensor data, tend to be part of distributed system and require specific domain knowledge [44] [52]–[54]. Also programming languages vary greatly, with memory safe programming languages such as JavaScript being more often used in non-IoT projects, in 18 vs. 4 projects in [44].

Code quality between IoT and non-IoT applications differs massively, software for IoT was "more complex, coupled, larger, less maintainable, and cohesive than non-IoT systems" [36] [55]. This results in some authors suggesting different development processes for IoT [56]–[58].

The unique requirements, specific and open architecture has drawn attention of both malicious attackers and security analysts.

C. Quality assurance

Motoga et al. [59] come to the conclusion that a "diversity of methods" is needed to assure the quality of IoT systems during the development phase. With regard to the lifecycle, "maintainability" is a key goal for the deployment of embedded systems [59], however there is a potential trade off between quality attributes such as security, safety and maintainability [59]. *Müller* [60] also argues that secure IoT development requires its own development process.

Quite interestingly, European Telecommunications Standards Institute (ETSI) issued the standard ETSI EN 303 645 "Cyber

Security for Consumer Internet of Things: Baseline Requirements" representing a collection of rather simple minimum requirements for secure consumer IoT in order to support manufactures to establish security by design. Considering that ETSI is less of an IT but more a telecommunication standardisation body, providing mobile phone network standards such as GSM, EDGE, 3G, 4G and 5G, its IT-security recommendations are not always up to date, ETSI TS 103 523-3 V1.2.1 "CYBER; Middlebox Security Protocol; Part 3: Enterprise Transport Security" even received a CVE number before publication [61]. Not only the authors consider ETSI EN 303 645 problematic, *Morgenstern at al* [62] state the implementation of the standard lacks completeness leading to security risks. Despite this, based on ETSI EN 303 645, the German BSI provides a quality seal for IoT devices, based on voluntary participation [63] [64].

If IoT uses encryption, updating algorithms used for encryption, signing or even hash functions might not be straight forward, since data stored on the device would need to re-encrypted, increasing the complexity of the process. Also interoperability issues with not yet updated systems may occur, if for security aspects downward compatibility has not been implemented.

IV. Suggested legislative support

When it comes to security of embedded systems and IoT, at first, security seems to be one attribute among others. However, security is of high importance due to the massive effects of security incidents, the risks of lateral movement of attackers, the abuse of bot nets built upon infected IoT devices and the dangers of cyber-physical effects. Considering this, legislators might support better security by passing more stringent laws, such as the EU did with Directive 2016/1148 of the European parliament and of the council of 6 July 2016 concerning measures for a high common level of security of network and information systems across the Union (NIS)-2 and CRA recently. These, however, would only become efficient if they required measurable IT security, i.e., an objective and neutral way to assess security.

A. The need of a security metric in a legal context

There is a need for objective and measurable security in a legal context, as a real example demonstrates: With an increasing amount of fraudulently manipulated invoices, where attackers swap the recipients bank account against their own, the Higher Regional Court (Oberlandesgericht (OLG)) in Karlsruhe ruled, that sending the invoice encrypted was not a requirement [65], while the OLG Schleswig-Holstein decided encryption is needed [35]. Both based their decision on the same article of the General Data Protection Regulation (GDPR). As a result, in one case, the invoiced party needed to pay again, in the other case not.

The two courts disagreed on two main issues, one was the applicability of the GDPR to a juridical person (Karlsruhe) versus a natural person (Schleswig-Holstein), which could be disputed, and the other on a technical question. According to the respective rulings neither court heard an expert witness on the matter. Both assumed in their own expertise, encryption was efficient to protect integrity, where a digital signature would actually be needed [34] [35] [65]. Karlsruhe found that encryption is not required based on "real world expectations", Schleswig-Holstein considered it as a requirement.

The decision by the OLG Schleswig-Holstein is currently being discussed mainly in the legal community for it basically enforcing email encryption. Some see it as being disruptive and too strict [66]–[68], while others point to a decade of advocacy towards email encryption [35] [65] [69].

The different rulings are a result of allowing room for interpretation of technical requirements by non technical legislators and a non technical judicative. Objectively measured security by contrast allows for consistent court rulings and also consistent expert hearings. Only with a reliable metric, legal concepts such as GDPR have an effect.

B. Economic impact

Procurement decisions should also be made based on a security metric [15]. By providing a legal requirement, it could easier be incorporated in the buying decision, and thereby enhance cyber resilience [15].

C. Labelling

In other domains like electronic devices or textiles labels, batches and seals are accepted and mandatory. For instance the Conformité Européenne (CE) label for electronic goods guarantees that minimum requirements for safety, health and environment have been addressed and fulfilled, albeit the CE label is based on a self assessment [70].

Self assessment has already proven problematic in IT in general, as most software manufactures declare their products to be of high quality, free of security and other flaws, which hardly ever meets reality. Also with the CE-label, these issues occur. If, however, the self assessment is based on a comparable, standardised metric, "fake" assessments could be identified. Still, a legal framework to prevent the abuse of labels – potentially more strict than the one used for CE-labels [71] – is needed.

D. Mandatory Testing

For products that society and lawmakers consider to carry a higher risk, mandatory tests and specific requirements are set up. Examples include cars, aircrafts or medical devices. In special cases regular re-assessments are a legal requirement as well, e.g., the yearly or bi-annual roadworthiness checks for cars or permanent side effect reporting in medicine. In all these cases, security and safety are a requirement, security issues are (almost) unacceptable. The GDPR introduced security requirements – at least in theory: Some consider penetration tests a GDPR requirement [72], however, reality does not agree.

Therefore a security metric would allow efficient, comparable and reliable security comparisons. It would allow any user to make a decision between a more or less secure device apart from other features. With more and more IoT devices connected, chances increase for exploited vulnerabilities.

E. Life cycle

Quality and security management need to addressed throughout the life cycle of an IoT project. Before allowing a product on the marketing, a formal check such as a roadworthiness checks in cars. Therefore a security metric serves as a foundation to measure criteria or attributes (to define) that need to be passed.

Other than with roadworthiness checks, with IoT not the individual device needs to be reevaluated, but only one exemplary device, as other than with cars mechanical wear and tear is not the primary issue. While reevaluation of all IoT devices would be advisable to prevent risks like lateral movement, in a first step it seems to be most important to both apply regular reevaluation to devices with cyber-physical effects as well as to devices used in critical infrastructure.

V. Conclussion and future work

Security issues are the downside of a technology that surrounds us. This article has shown that embedded systems require special attention to run reliable and secure in an interconnected world. The design and creation of IoT services as well as maintenance tasks along the usage phase appears to be of high complexity.

The authors propose the usage of a metric to measure a software security index that supports the judgement of feasibility for applications in a given context. The metric is therefore addressing the "static" software system with regard to the technical aspects but also considering dynamic aspects. The latter could be part of a penetration test scenario. A metric would allow a judgment in terms of security in IoT environments. Above, a regular validation could be part of a "ready for market" concept. IoT manufacturers would probably have an interest in a better rating leading to quality improvement of the software system.

References

[1] ZDNet. (2017) FDA issues recall of 465,000 St. Jude pacemakers to patch security holes. Accessed 2025.03.09. Online: https://www.zdnet.com/article/fda-forces-st-jude-pacemaker-recall-to-patch-security-vulnerabilities/

[2] A. Holt and E. Holt, *Flimmer*. Piratforlaget, 2010.

[3] M. Komar. (2016) DDoS attack takes down central heating system amidst winter in finland. Accessed 2025.03.09. Online: https://thehackernews.com/2016/11/heating-system-hacked.html

[4] cve.org. (2025) CVE.org search for SIMATIC. Accessed 2025.03.09. Online: https://www.cve.org/CVERecord/SearchResults?query=SIMATIC

[5] ——. (2017) Cve-2017-7240. Accessed 2025.03.09. Online: https://nvd.nist.gov/vuln/detail/CVE-2017-7240

[6] K. Tindell. (2023) CAN injection: keyless car theft. Accessed 2025.03.09. Online: https://kentindell.github.io/2023/04/03/can-injection/

[7] AZDOME. (2021) Method of updating the latest firmware. Accessed 2025.03.09. Online: http://forum.azdome.hk/forum.php?mod=viewthread&tid=930&highlight=M300S

[8] S. Harding. (2025) Firmware update bricks HP printers, makes them unable to use HP cartridges. Accessed 2025.03.11. Online: https://arstechnica.com/gadgets/2025/03/firmware-update-bricks-hp-printers-makes-them-unable-to-use-hp-cartridges/

[9] F. Deusch and T. Eggendorfer, "Zur Kompatibilität beim Updating verbundener Systeme (translated: On compatibility when updating interdependant systems)," *K&R*, vol. 2018, no. 7, pp. 456–464, 2018.

[10] M. Zysset. (2025) Wenn das Software-Update beim Auto für Stau sorgt (translated: If the car software update causes a traffic jam). Accessed 2025.03.09. Online: https://www.tagesanzeiger.ch/bls-autoverlad-wenn-das-software-update-fuer-stau-sorgt-135808833029

[11] G. Hull, C. Trivella, and J. Seland. (2025) Camera off: Akira deploys ransomware via webcam. Accessed 2025.03.11. Online: https://www.s-rminform.com/latest-thinking/camera-off-akira-deploys-ransomware-via-webcam

[12] CISA. (2017) Heightened DDoS threat posed by Mirai and other botnets. Accessed 2025.03.09. Online: https://www.cisa.gov/news-events/alerts/2016/10/14/heightened-ddos-threat-posed-mirai-and-other-botnets

[13] Hewlett-Packard. (2025) HP Universal print driver series (PCL 6 and PostScript) - potential security vulnerabilities. Accessed 2025.03.09. Online: https://support.hp.com/us-en/document/ish_11892982-11893015-16/hpsbpi03995

[14] K. Pawar, C. Ambhika, and C. Murukesh, "IoT hacking: Cyber security point of view," *Asian Journal of Basic Science & Research*, 2021. Online: https://api.semanticscholar.org/CorpusID:235194057

[15] T. Eggendorfer and K. Andresen, "Using security metrics to improve cyber-resilience," in *Proceedings of the IARIA Congress 2024*. Porto, Portugal: IARIA, 2024, pp. 152–157. Online: https://www.thinkmind.org/library/IARIA_CONGRESS/IARIA_Congress_2024/iaria_congress_2024_2_210_50107.html

[16] P. Hengge, *Development of a Rule Set for the Defence Against Cyber Attacks by Analysing and Evaluating Attack Vectors in IT Systems*. Hagen: Masterthesis, FernUniversität in Hagen, 02 2025.

[17] B. Schneier. (2025) Entries tagged security theatre. Accessed 2025.03.09. Online: https://www.schneier.com/tag/security-theater/

[18] P. R. Zimmermann. (1991) Beware of snake oil. Accessed 2025.03.09. Online: http://www.philzimmermann.com/EN/essays/SnakeOil.html

[19] J. Michal, *Tesla Model 3 Control Units Security Analysis*. Prag: Masterthesis, CTU Prag, 01 2021.

[20] DragTimes. (2014) Tesla model s ethernet network explored, possible jailbreak in the future? Accessed 2025.03.09. Online: http://www.dragtimes.com/blog/tesla-model-s-ethernet-network-explored-possible-jailbreak-in-the-future

[21] J. Leyden. (2017) Dildon'ts of bluetooth: Pen test boffins sniff out berlin's smart butt plugs. Accessed 2025.03.09. Online: https://www.theregister.com/2017/09/29/ble_exploits_screwdriving/

[22] OWASP. (2021) OWASP top ten. Accessed 2025.03.09. Online: https://owasp.org/Top10/

[23] J.-L. Lions. (1996) Flight 501 failure. Accessed 2025.03.09. Online: http://sunnyday.mit.edu/accidents/Ariane5accidentreport.html

[24] J. Drake. (2015) Stagefright: Scary code in the heart of Android. Zimperium. Accessed 2025.03.09. Online: https://www.blackhat.com/docs/us-15/materials/us-15-Drake-Stagefright-Scary-Code-In-The-Heart-Of-Android.pdf

[25] ISO. (1989) Iso/iec 9899:tc3. Accessed 2025.03.09. Online: https://www.open-std.org/jtc1/sc22/wg14/www/docs/n1256.pdf

[26] J. P. Anderson. (1972) Computer security technology planning study, volume ii. Accessed 2025.03.09. Online: https://csrc.nist.gov/csrc/media/publications/conference-paper/1998/10/08/proceedings-of-the-21st-nissc-1998/documents/early-cs-papers/ande72.pdf

[27] A. One. (1996) Smashing the stack for fun and profit. Accessed 2025.03.09. Online: https://phrack.org/issues/49/1

[28] T. Eggendorfer, "At the source. static code analysis finds avoidable errors," *Admin Magazine*, vol. 2019, no. 53, 2019. Online: https://www.admin-magazine.com/Archive/2019/53/Static-code-analysis-finds-avoidable-errors

[29] A. Lentz and T. Eggendorfer, "Snakes in the grass," in *NLUUG najaarsconferentie 2024*. Utrecht, Netherlands: NLUUG, 2024.

[30] J. Horn *et al.* (2018) Spectre and meltdown. Accessed 2025.03.09. Online: https://spectreattack.com/

[31] M. Lipp *et al.*, "Meltdown: Reading kernel memory from user space," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018.

[32] P. Kocher *et al.*, "Spectre attacks: Exploiting speculative execution," in *40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019.

[33] Synopsis. (2020) Heartbleed. Accessed 2025.03.09. Online: https://heartbleed.com/

[34] T. Eggendorfer, "Schwer verrechnet (translated: Bad computation)," *Linux Magazin*, vol. 2025, no. 04, pp. 50–53, 2025.

[35] F. Deusch and T. Eggendorfer, "E-Rechnung: Verschlüsseln oder Signieren? (translated: E-invoice: Encrypt or sign?)," *K&R*, vol. 2025, no. 4, 4 2025.

[36] N. Khezemi, S. Ejaz, N. Moha, and Y.-G. Guéhéneuc, "Comparison of code quality and best practices in IoT and non-IoT software," *ArXiv*, vol. abs/2408.02614, 2024. Online: https://api.semanticscholar.org/CorpusID:271709982

[37] D. Mathes, *Qualitätsmetriken für Schutzkonzepte (translated: Quality metrics for security concepts)*. Hagen: Masterthesis, FernUniversität in Hagen, 12 2015.

[38] OpenBSD. OpenBSD security. Online: http://www.openbsd.org/security.html

[39] C. A. R. Hoare, "An axiomatic basis for computer programming," *Commun. ACM*, vol. 12, no. 10, p. 576–580, oct 1969. Online: https://doi.org/10.1145/363235.363259

[40] G. Klein *et al.*, "sel4: formal verification of an operating-system kernel," *Commun. ACM*, vol. 53, no. 6, p. 107–115, Jun. 2010. Online: https://doi.org/10.1145/1743546.1743574

[41] D. B. de Oliveira, T. Cucinotta, and R. S. de Oliveira, "Efficient formal verification for the linux kernel," in *Software Engineering and Formal Methods*, Peter Csaba Ölveczky and G. Salaün, Eds. Cham: Springer International Publishing, 2019, pp. 315–332.

[42] SEL4project. (2025) seL4. Accessed 2025.03.09. Online: https://sel4.systems/

[43] F. Deusch and T. Eggendorfer, "Messbarkeit von IT-Sicherheit (translated: Measuring it-security)," *K&R*, vol. 2023, no. 12, pp. 781–786, 12 2023.

[44] F. Corno, L. D. Russis, and J. P. Sáenz, "How is open source software development different in popular IoT projects?" *IEEE Access*, vol. 8, pp. 28 337–28 348, 2020. Online: https://api.semanticscholar.org/CorpusID:211227160

[45] M. Loghi, T. Margaria, G. Pravadelli, and B. Steffen, "Dynamic and formal verification of embedded systems: A comparative survey," *International Journal of Parallel Programming*, vol. 33, pp. 585–611, 2005. Online: https://api.semanticscholar.org/CorpusID:22718479

[46] T. Kölnberger, *Evaluation of effects of security metrics in the software development life cycle*. Hagen: Masterthesis, FernUniversität in Hagen, April 2025.

[47] OWASP. SAMM model overview. OWASP. Online: https://owaspsamm.org/model/

[48] C. Binder, *Entwurf einer Metrik zur Bewertung des IT-Sicherheitsniveaus am Beispiel von Webanwendungen (translated: Design of a metric to measure the IT security level of web applications)*. Hagen: Masterthesis, FernUniversität in Hagen, February 2024.

[49] S. Reckhaus, *IT-Sicherheit und Kosten-Nutzen Analyse von Cyber-Versicherungen (Translated: IT-security and cost-effect analysis of cyber insurance)*. Hagen: Masterthesis, FernUniversität in Hagen, 02 2016.

[50] D. Wuttig, *IT-Sicherheitsprüfung im Internet of Medical Things (Translated: IT-security testing in the Internet of Medical Things)*. Hagen: Masterthesis, FernUniversität in Hagen, 08 2022.

[51] CISA. (2025) Contec Health CMS8000 patient monitor (Update A). Accessed 2025.03.09. Online: https://www.cisa.gov/news-events/ics-medical-advisories/icsma-25-030-01

[52] A. Taivalsaari and T. Mikkonen, "A roadmap to the programmable world: Software challenges in the IoT era," *IEEE Software*, vol. 34, pp. 72–80, 2017. Online: https://api.semanticscholar.org/CorpusID:6751128

[53] ——, "On the development of IoT systems," *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 13–19, 2018. Online: https://api.semanticscholar.org/CorpusID:44147446

[54] N. Alhirabi, O. F. Rana, and C. Perera, "Security and Privacy Requirements for the Internet of Things," *ACM Transactions on Internet of Things*, vol. 2, pp. 1 – 37, 2021. Online: https://api.semanticscholar.org/CorpusID:231730970

[55] M. Klíma *et al.*, "Selected code-quality characteristics and metrics for Internet of Things systems," *IEEE Access*, vol. PP, pp. 1–1, 2022. Online: https://api.semanticscholar.org/CorpusID:248517759

[56] S. S. Ismail and D. W. Dawoud, "Software development models for IoT," *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0524–0530, 2022. Online: https://api.semanticscholar.org/CorpusID:247230583

[57] J. P. Dias and H. S. Ferreira, "State of the software development life-cycle for the Internet-of-Things," *ArXiv*, vol. abs/1811.04159, 2018. Online: https://api.semanticscholar.org/CorpusID:53282492

[58] X. Larrucea, A. Combelles, J. Favaro, and K. Taneja, "Software engineering for the Internet of Things," *IEEE Software*, vol. 34, no. 1, pp. 24–28, Jan 2017.

[59] S. Motogna, A. Vescan, and C. Şerban, "Empirical investigation in embedded systems: Quality attributes in general, maintainability in particular," *J. Syst. Softw.*, vol. 201, no. C, Jul. 2023. Online: https://doi.org/10.1016/j.jss.2023.111678

[60] M.-C. Müller, *Konzeption eines ganzheitlichen, die IT- und funktionale Sicherheit unter berücksichtigenden IoT-Entwicklungsansatzes (Translated: Concept for a holistic IoT development approach including IT and functional security)*. Hagen: Masterthesis, FernUniversität in Hagen, 06 2018.

[61] CVE.org. (2019) Cve-2019-9191. Accessed 2025.03.09. Online: https://www.cve.org/CVERecord?id=CVE-2019-9191

[62] M. Morgenstern, O. Pursche, and E. Clausing, "Die Sicherheitslage im IoT-Umfeld: Steigende Gefahrenlage und Sicherheit durch Tests (translated: The state of security in IoT: Increased risks and security by testing," *Datenschutz und Datensicherheit - DuD*, vol. 45, pp. 102–106, 02 2021.

[63] BSI. (2022) Consumer IoT. Accessed 2025.03.09. Online: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Consumer-IoT/Consumer-IoT.html

[64] F. Deusch and T. Eggendorfer, "Update IT-Sicherheitsrecht 2021/2022 (translated: Update IT-security-law 2021/2022)," *K&R*, vol. 2022, no. 12, pp. 794–803, 2022.

[65] ——, "Update IT-Sicherheitsrecht 2022/2023 – Teil 2 (translated: Update IT-security-law 2022/2023 part 2)," *K&R*, vol. 2024, no. 4, pp. 242–248, 4 2024.

[66] R. Petrlic. (2025) Remarks on the OLG Schleswig-Decission (Annoucement). Accessed 2025.03.09. Online: https://www.linkedin.com/posts/petrlic_datenschutz-dsgvo-cybersicherheit-activity-7294652352156897280-QMjx/

[67] R. Petrlic and J. Zwerschke, "OLG Schleswig: Ende-zu-Ende-Verschlüsselung ist nach der DSGVO im Geschäftsverkehr regelmäßig erforderlich (translated: OLG Schleswig: End-to-end encryption is a requirement in business communications according to gdpr)."

[68] N. Härting. (2025) Gefährdungshaftung aus Art. 32 DSGVO – OLG Schleswig verirrt sich in die DSGVO und bezeichnet Papier als "Mittel der Wahl" (translated: Liability based on Art. 32 GDPR - OLG Schleswig getting lost in the GDPR and suggesting paper as means of choice). Accessed 2025.03.09. Online: https://www.pingdigital.de/blog/2025/03/03/gefaehrdungshaftung-aus-art-32-dsgvo-olg-schleswig-verirrt-sich-in-die-dsgvo-und-bezeichnet-papier-als-mittel-der-wahl/2528

[69] F. Deusch and T. Eggendorfer, "Verschlüsselte Kommunikation im Unternehmensalltag: Nicetohave oder notwendige Compliance? (translated: Encrypted communications in day-to-day business: Nice-to-have or required compliance)," *K&R*, vol. 2018, no. 04, pp. 223–230, 04 2018.

[70] EU. (1993) Council Directive 93/68/EEC of 22 July 1993. Accessed 2025.03.09. Online: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:01993L0068-19980812

[71] F. Gronkvis. (2023) What is fake CE marking? Accessed 2025.03.09. Online: https://www.compliancegate.com/fake-ce-marking/

[72] F. Deusch and T. Eggendorfer, "Penetrationstest bei Auftragsverarbeitung (translated: Penetration test with processors)," *K&R*, vol. 2018, no. 04, pp. 223–230, 04 2018.

ACKNOWLEDGEMENTS

The authors like to thank the reviewers for their valuable and useful comments that were helpful in improving and clarifying the paper.