Using Security Metrics to improve Cyber-Resilience

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*—Not only critical infrastructure but also everyday interaction in a society relies heavily on secure IT systems. Examples of patients dying due to hospitals unable to admit them because of a ransomware incident indicate a low level of cyber-resilience. To increase cyber-resilience, suggested measures range from anti-malware via backups and redundancy to regular security updates. While following these guidelines, there is an intensive discussion which systems provides the best security. There is no answer – yet. IT security lacks a reliable system to measure security in order to compare systems and make a qualified decision. This paper discusses current research in security metrics and why it is important to provide a security metric to improve cyber-resilience. The authors discuss the advantages, the state of the art and future research needed in order to improve cyber-resillience with security metrics.**

Keywords-Security Metrics; Quality Metrics; Software Security; Software Quality, Cyber-Security; Cyber-Resillience

I. Introduction

In March 2022, the German Federal Office for IT-Security (BSI) issued a security warning regarding the use of Kaspersky Anti-Virus. Since the company is owned by Russians, in light of the Ukraine-Russia war, this would pose a security risk [1]. This was the first warning the BSI has ever released, and it was not based on a software security issue, but on ownership of a software company. This decision meant to increase Germany's cyber-resilience was heavily discussed in and out of court [2].

The Russian war also introduced cyber-attacks by pro as well as contra Russian groups. Those included the usual ransomware incidents, but also attacking railway networks to disturb Russian troop movement [3]. In the wake of these attacks, a cyber-physical security issue in uninterruptible power supplies that could set them and connected devices on fire, was discovered [4]. Whether it was actively exploited is currently unknown, however it demonstrates that even systems meant to increase Information Technology (IT) safety could be attacked. The same holds true for IT security tools with security issues, such as attacks on firewalls [5] or anti-malware [6].

A. The need for security metrics

The vendors of both security and regular software claim their products were secure, or at least as secure as currently possible. However, there is no approved and commonly accepted method to verify this. For most cases even measurable security requirements do not exist.

Based on a broadly understood security metric, also combinations of systems could be evaluated and compared on a security scale. Applying this, IT security professionals could easily identify whether adding e.g., an anti malware solution would actually help security or even reduce security levels.

B. Cyber-Resilience through security metrics

The US government has recently been urged to stop using Microsoft products, amid fears government data could be stolen considering the still ongoing "hacking attack" on Microsoft. To the US Government their software seems to be too vulnerable, posing a too high risk on the US society. If a nations government and thus public infrastructure has such a high risk of being successfully hacked, cyber-resilience of this nation is not satisfactory. Hence, using different software could increase the national cyber-resilience [7].

Whether or not the US government is right, is uncertain - even though the authors share this impression. But rather than belief, anecdotal reports or personal experience, a security metric would support objective judgement. It would immediately enhance cyber-resilience by allow to select appropriate systems.

C. Structure of this paper

The paper is structured as follows: Section II provides a background on security issues (Section II-A), attack vectors, current cyber-security measures (Section II-B), cyber-resilience (Section II-D) and legal requirements for cyber-security (Section II-C). It concludes with how cyber-security issues are currently prevented (Section II-E), how ownership and thereby responsibility could affect cyber-resilience (Section II-G) and how security is managed outside IT (Section II-F).

Section III then discusses software quality and security metrics as well as software quality standards. Additionally it looks at formal verification as a software security measure.

We then continue in Section IV on the impact a security metric would have on cyber-resilience, and approach this from a legal and economic perspective.

Section V provides a conclusion and gives an outlook.

II. Background

To some, security incidents happen like accidents, hardly predictable and are the product of avid hackers. To others security incidents are a result of insufficient software quality assurance in the development process. We asume, strict quality management could prevent the vast majority of them, since security issues follow patterns and abuse flaws introduced during development. This also considers factors such as the legal perspective on system quality and vendor ownership.

A. Typical attack vectors

This dispute may not be fully resolved, however analysing typical attack vectors could provide an indication. The two most prevalent is the abuse of a security flaw in software, be it a publicly known and patched or yet unknown "zero-day", and tricking users into insecure behaviour such as running unknown programs or installing software.

1) Security relevant software issues: While for web applications the OWASP Foundation provides a regularly updated list of their perception of the ten most relevant security issues [8], there is nothing equivalent for regular software, such as operating systems or office software.

On a very high level of abstraction security issues result from the possibility to alter the program's work flow to execute commands in favour of the attacker or to ignore restrictions in accessing data. Examples include shell code injections through buffer overflows or format string vulnerabilities as well as remote shell command injections, which might even be achieved through Structured Query Language (SQL)-injections by "selecting" a series of commands into a local file that is later executed.

Some of these injections are straight forward, such as using a script language's `eval`-function with not sanitised user input, others are a lot more complex, such as the buffer overflows triggered by integer overflows in stage-fright [9].

These two examples not only differ in their exploitation complexity but also in the difficulty to discover them in the source code: While `eval` is detected with automated code review [10], stage-fright was partly due to how a C-compiler handled a type conversion between 32 and 64 bit integers when multiplying two 32-bit values: Without an explicit typecast of at least one of the two factors to 64 bit, despite the result variable being 64 bit, the compiler used a 32 bit multiplication, resulting in an integer overflow. To detect this and the resulting buffer overflow a deep understanding of the compiler and its behaviour are needed [9].

Prevention of the first is as simple as detecting it, while the latter might require more and deeper code analysis.

The complexity of security issues, both in exploiting as well as preventing them might have an impact on the overall security of software. One would assume if a system is vulnerable to simple security issues, it might have an overall lower security level than one that requires more elaborate exploits.

2) Human behaviour: Often security incidents are said to be related to user behaviour, with awareness and training proposed as a remedy [11][12][13][14][15][16]: Clicking on suspicious software or otherwise allowing attackers to install malware is often considered to be a major security issue. However if a simple, unaware user interaction breaks a system's security one might as well argue that this system has a low overall security level and is not resilient to attacks.

Current ransomware attacks serve as example: Installed by the user in reaction to a Trojan like attack, i.e., social engineering, they abuse the ability of a single mal-functioning process to escalate privileges to encrypt the disk.

3) Impact of security issues: Both the complexity to exploit security issues and the impact might vary. Scores like the Common Vulnerability Scoring System (CVSS) [17] only estimate the impact, helping a system administrator to decide on the urgency of a patch or other counter measures. However, they do neither provide a metric for the overall security of software or systems, nor does the score address context.

B. Current counter measures

To prevent security incidents currently system administrators are requested to install additional, potentially vulnerable software, such as malware scanners, firewalls or content filtering proxies [5][6], considered state of the art security measures. Simoultanously, users learn what to click and what not [18], including elaborate spear phishing to show users how vulnerable they are – or how well they detect threads.

This moves the perception of the reason for the security issue far away from the software quality issue, it actually is: It is often considered to a user or human factors problem.

C. Legal requirements on software security

EU regulations such as NIS-2 require software manufacturers to provide better security, the same holds true for concepts such as the BSI introducing a "Software Quality"-seal, based on a self-evaluation by the software manufacturers .The seal also uses a fair bit of their reputation and how fast they would provide security patches [2]. While the speed of fixing issues is a reasonable measure, neither the frequency of the need for fixes nor the severity of the flaws is not taken into account. Nor whether they could have been prevented. Even the manufacturers attitude and interaction towards external security researchers is not considered – there still are software providers threatening to sue those reporting security issues [19], while other embrace them, providing bug bounties. Some manufacturer even try to downplay reports as not security relevant, e.g., because they do not provide a proof-of-concept exploit. For the seal matters, they are better off.

The frequency of security patches might as well give a rough estimate on how efficient quality management is: In other industries, such as cars, if a manufacturer often needs to recall his cars, it immediately affects the quality perception.

Recently, the EU Cyber-Resilliance-Act (CRA) transported concepts such as a Bill of Materials (BoM), well known outside IT, to software. The Software Bill of Materials (SBoM) lists third party software, e.g., external libraries used, with the intention to identify security issues related to external software. The log4shell issue [20] is a good example for the benefits of a SBoM, since many projects relied on Log4J. The SBoM might also help to estimate the quality of software.

D. Cyber-Resilience

Cyber-resilience is the ability to react, respond, adapt or pro actively anticipate incidents on cyber-connected infrastructures [21][22][23][24]. It deploys cyber-security by continuously learning from incidents to adapt to new levels of robustness to fulfil business objectives. Therefore, for any organisation

or government institution in an interconnected cyber-physical world increasing cyber-resilience is considered highly relevant.

Cyber-resilience prevents security incidents through system design. This might include concepts such as backup- and recovery-plans, emergency procedures, but first and foremost requires systems secure by design. This is what security metrics measure.

Software security metrics therefore would also contribute to defence capabilities, incident detection, crisis management while additionally increasing the collective resilience. They would also reduce the effects of cyber-warfare.

E. Preventing these attacks

An effective way to prevent software issues might well be to educate software developers more. Projects with high security levels such as OpenBSD use simple concepts, such as code reviews, clear coding standards, automated software testing etc. [25]. The OWASP SAMM provides a similar guidance for the software development cycle consisting of governance design, implementation and verification measures as well as defined processes if issues would arise despite all the effort put into the project so far [26]. Both aim to avoid typical programming mistakes that could lead to security issues.

These measures preventing attacks need to have an immediate impact on a security score. Currently, they are not accounted for in most software buying decisions. With a security metric, they become visible, affect procurement decission and thus impact cyber-resilience.

F. Measuring security outside IT

Measuring requires to code observations with numbers, which would then create a scale - be it a physical feature like temperature or a non-physical feature like quality [27].

Measuring security and quality of software, an intangible good, is much harder than for tangible ones. Still, some quality measurement concepts could be adapted from other areas.

In automobile industry, security as well as safety issues due to quality problems are of high relevance: They cannot easily be fixed with an update customers install, but requires to transport the vehicles to a workshop, potentially providing customers with a replacement car while the issue is fixed, and thereby generates high costs for the manufacturer. These alone may provide a relevant motivation for higher quality. A recall by national car safety associations, forcing manufacturers to fix all relevant models immediately, is also public.

Therefore car industry implements massive quality checking on items bought by external manufacturers, often with contractual penalties if requirements are not met. This concept only slowly moves over to software industry with the concept of a SBoM by both the CRA and the US Executive Order 14028. Still neither requires any of the library providers to document quality measures.

Car industry invests heavily in crash tests to analyse safety issues, the results of these tests provide a score, which in turn influences development of future cars and allows for comparison – again, regulations such as General Data Protection Regulation (GDPR) [28] and CRA slowly introduce the need for penetration tests, which is an equivalent to a crash test in providing both a security measure and input to a feedback loop affecting the development cycle. However this has not yet been widely adopted.

Those measures do not substitute quality management, they only report on the effects. By contrast, a software security metric measures quality and thereby has an immediate effect on the development process.

G. Responsibility and Ownership

A key challenge to be solved is to determine the indicators to consider as part of the metric. On an operational level the scope and impact of potential attacks appear to be based on security flaws immanent to the software. The (quality) assessment addresses critical aspects by analysing the system behaviour and identifying the effects.

The software in use is usually part of a contract system including parties such as vendors and manufacturers in charge of updates, customising and changes. The BSI warning on replacing security software of a certain vendor (see Section I was purely based on ownership issues. The manufacturer and not the software itself gave reason to alert the nation [1].

A broadened perspective on impact factors could include further indirect aspects as expressed responsibility for security and safety. A “responsibility model” might include an assessment of security compliance rules, e.g., in terms of openness and communication towards security researchers reporting potential bugs (sued or rewarded).

Hence, current challenges impose a need to discuss issues as ownership and responsibility to increase cyber-resilience. Adding company profiles along with a technical software assessment might therefore contribute to a meaningful metric.

III. Research in software quality and security metrics

Despite the importance of measurable software quality and security, currently there are no metrics available. Software quality serves as a proxy for security: Quality means the absence of flaws, each flaw could result in a security issue.

A. Software Quality measurement

Early research on software quality includes [29], proposing to identify parts of a program with a high probability for flaws – the focus was not on security issues, but on functional problems. But this research only analysed the final product without incorporating earlier unit tests into analysis. Other early researchers suffer from different understandings of software quality and security, with no common definition [30][31]. The latter introduces a complex metric with weights for several aspects of quality, however it does not provide how to compute the individual quality measures.

[32] uses a retrospective approach by measuring reported flaws in software. For a security score, this could be replicated using the Common Vulnerabilities and Exposures (CVE)-database. But neither are all issues are assigned a CVE-number

nor are all published, the score is not precise. Also some software is hardly ever scrutinised by security experts, but other under heavy analysis. Thus CVEs would only provide a rough indication rather than a precise score. Additionally the score measures past not current quality.

B. Software Security Metrics

Later work focused more on security metrics, such as [33], providing relevant reasons for measuring security, but no metric, since this would be too complex. in [34] the author suggests to divide complex systems into smaller components for analysis.

[35] encounters the same issue: While the authors consider security metrics as highly relevant, and provide a definition of security, they eventually consider a measured metric as far too complex. As a work-around they suggest to at provide estimation methods, which they then demonstrate outside the IT security world. Unfortunately they do not back-port the results to IT. [36] and [37] come to the same conclusion, when providing accurate computation methods to compose security scores, but are unable to provide the individual scores.

[38] defines a Software Reliabilty Metric, which differs from a security metric, in that it analyses how often software faults occur during a given time frame. The work also discusses several testing methods for this metric, either based on black box testing, which could be adapted to security metrics by a penetration test, or on a software metric, like lines of code, complexity and developer experience, or finally from analysing the software components, called ”architecture based” by the respective authors. The work clearly points out that reliability issues in software are never based on wear, but always on design and usage. But overall, while the work provides good concepts, it does not deliver an usable metric, that could be modified to a software security metric.

More recently [39] as well as [40] started analysis into security metrics, however their focus is on cryptographic protocols. While still hard to apply quantifiable metrics to them, feasibility seems to be higher, since the scope is more focused. Another more accessible area for security metrics is networking [41], since it is less complex than software.

[42] suggests a metric to measure the security of web applications, thereby reducing the complexity to a subset of possible programs. However the metric is still mostly reactive, contains some data that cannot be measured like ”knowledge in security”. The work however indicates there is still a need for research.

Additionally there are some reactive scores, including Time to Patch or Mean Time to Remediate (MTTR), both indicating how long it took for a reported issue to be mitigated, Vulnerability Density, a measure for how many issues exists, Mean Time to Detect (MTTD), giving an idea how long an issues remains undetected. All of these however do neither measure the quality nor the security of the code, they only provide *ex-post* measure.

C. Quality standards

Some ISO standards relate to security and quality, such as ISO 27000 series, ISO 15408 and the derived common criteria as well as IEEE 1028. All of these define processes related to quality, but no measurable quality criteria. By that any certification does not indicate whether a software product is secure. A, e.g., ISO 27001 certified organisation would know how to cope with security issues, but there are no prevention mechanisms for flaws in the code. These however would increase security and thus affect a security metric.

D. Formal Verification

A mathematical approach to software quality is formal verification with, e.g., Hoare logics [43]. Clearly defined pre- and post-conditions for a command or a set of commands identify logic flaws in programs. By enhancing these conditions also issues such as off-by-one or buffer-overflow could be detected and thus be prevented [44]. However formal verification is a tedious and slow process. Current research tries to establish faster and more efficient methods and support schemes [45].

IV. Impact of a security metric

While software security metrics seem hard to achieve and will require more research, there is an immediate need for them, since they would have a multitude of effects, be them legal, economical or by improving cyber-resilience. [33] pointed some of these effects out, a list we add on in this paper.

A. Legal effects

With the GDPR forcing data processors to both evaluate their own IT security as well as that of sub-processors when it comes to processing personal data, a security metric would allow them to speed up this evaluation process while maintaining independence from vendor claims. While larger corporations may be able to do their security assessments from a technical perspective, smaller entrepreneurs can hardly afford the extra costs and time. Often enough they do not have the expertise either. In practice whenever security assessments in a GDPR context are performed, vendors usually require non disclosure agreements, preventing sharing the results with other interested parties, which could streamline the process at least.

As a result most data processors resort to either using software and systems they assume to be secure based on vendors' claims or, even worse, because others use these systems. Ironically those others will also point to other users. The issue is well known with data protection authorities, they therefore tend to be reluctant in punishing those confronted with the impossibility to verify claims. From a legal and a socio-technical perspective this is highly unsatisfactory, since it provides a factual bypass. It is also counter-productive to the concept of cyber-resilience as it is endorsed by the GDPR.

With a security metric in place, both data processors as well as authorities could much easier enforce the use of secure products and sanction the use of insecure, by that creating a market incentive for vendors to (finally) improve their products' security.

Critical infrastructure regulations in different legislations also enforce assessment of IT security and thus adequate levels of security. To do so, they would as well need to be able to assess

any software vendors systems. This is hardly feasible, especially when it comes to closed source software. With a security metric the assessment was simplified, which would allow for more straight-forward compliance with the legal requirements. Currently, this is achieved by claiming "best practices" were used, which often means using the same product as others do, often by simply calling it an industry standard. That "standard" provides then the default security level. As many security incidents demonstrate, this self-reinforcing industry standard obviously is not up to the levels needed.

Without knowing which system is secure and which is not, critical infrastructure providers cannot invest in the most secure solution, which – by definition of critical infrastructure – has a negative effect on the cyber-resilience.

Similar legal requirements are put forward by regulations such as the CRA and NIS-2. Thus the legal system has a strong desire and need for objective software security metrics.

B. Economic effects

As shown above, Investing in software requires by law to buy a secure product, which is hard to identify. The markets lack of transparency results in non-optimal buying-decisions. Market mechanisms as set forward by the free market concept [46] [47] however require buyers to be able to make wise and educated decisions, they should neither be fooled by wrong advertising nor decide under an obvious lack of information.

In most markets for tangible goods there are quality tests and measures in place, almost everywhere in the world there are product testing organisations such as "Consumer Bonds" in the US, "Which?" In the UK or "Stiftung Warentest" in Germany. The same is available for industry grade products. This provides for well-informed buyers and their educated market decisions.

For a free market an objective security metric is therefore an important requirement. Again the car industry could provide an example: 140 years ago, when the "Benz Motorwagen" was the first car, customers did invest in this new means of transportation for various reasons, safety and reliability were not among those. In the 1960s, when Volvo introduced reinforced passenger cells and the three point security belt, as well as Mercedes started with crash testing their vehicles, customers slowly started to understand the need for safety measures and made them a requirement. Since then, wearing a seat belt has become mandatory by law in most countries, passive security has been encompassed by an easy to understand star system in the NCAP series.

As a consequence safety has influenced buyers decision since, and has become so common place that by now, it is taken as granted and customer interest has since moved to other parameters, such as eco-friendliness and usability.

C. Effects on Cyber-Resilience

In a market where purveyors of software have a strong incentive to invest in more secure systems due to customers demand, triggered by an objective software security metric, it is to be expected that cyber-resilience increases – much similar to a constantly decreasing amount of deadly traffic accidents due to increased safety there.

Compared to other industries, the current system of trusting software vendors' security promises and relying on the buying decision of others seems inferior. It undermines cyber-resilience by providing software vendors with the wrong incentive to constantly increase their customer base, potentially consider lock-in-effects in order to keep their customers and thereby hope that their installed base would keep them ahead of competitors, making their products a *de-facto* 'industry standard".

More impact is to be expected once scores for single programs could be combined to provide an overall system score rather than an individual score for just a software product. With this measure, deciding which operating system to run a word processor on becomes feasible, e.g., whether a system providing LibreOffice should run with Linux, MacOS, BSD or Windows, when security is paramount. The combined would also identify the weakest link and to mitigate the risks associated to it.

V. Conclusion and Outlook

The concept of introducing a software security metrics is not new, it has been discussed since decades – first with a focus on quality, later with an explicit focus on security. Research in that field has stalled, current research therefore has not yet provided any practical applicable results. Since not providing immediate results, funding seems to have stopped, furthering the stall.

In this paper we address the current state as well as impact factors to be discussed in the future. Our work demonstrates that there is a strong need for further research in security metrics, as we show security metrics are already required by law. We also show that security metrics would have a massive economical impact and help the software market to move to more secure software. By doing so we show that cyber-resilience would increase. We therefore strongly suggest to increase research into security metrics.

References

[1] Bundesregierung, "Aktuelle Warnung des BSI (translated: Current BSI warnings)," 2022. [Online]. Available: https://www.bundesregierung.de/breg-de/themen/sicherheit-und-verteidigung/cybersicherheitslage-kaspersky-2015970

[2] 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.

[3] A. Roth, "Cyberpartisans hack belarusian railway to disrupt russian buildup," 2022. [Online]. Available: https://www.theguardian.com/world/2022/jan/25/cyberpartisans-hack-belarusian-railway-to-disrupt-russian-buildup

[4] G. Levy, Y. Sarel, B. Seri, and B. Hadad, "Tlsstorm," 2022. [Online]. Available: https://info.armis.com/rs/645-PDC-047/images/Armis-TLStorm-WP%20%281%29.pdf

[5] CISA, "Threat actors exploiting F5 BIG-IP CVE-2022-1388," 2022. [Online]. Available: https://www.cisa.gov/news-events/cybersecurity-advisories/aa22-138a

[6] NIST, "CVE-2023-24934: Microsoft defender security feature bypass vulnerability," 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2023-24934

[7] U. D. of Homeland Security, "Cyber safety review board releases report on Microsoft Online Exchange incident from summer 2023." [Online]. Available: https://www.dhs.gov/news/2024/04/02/cyber-safety-review-board-releases-report-microsoft-online-exchange-incident-summer

[8] OWASP, "OWASP top ten," 2021. [Online]. Available: https://owasp.org/Top10/

[9] J. Drake, "Stagefright: Scary code in the heart of Android," Zimperium, 2015. [Online]. Available: https://www.blackhat.com/docs/us-15/materials/us-15-Drake-Stagefright-Scary-Code-In-The-Heart-Of-Android.pdf

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

[11] Verizon, "Data breach investigations report." [Online]. Available: https://www.verizon.com/business/resources/reports/dbir/

[12] R. Rohan, S. Funilkul, D. Pal, and W. Chutimaskul, "Understanding of human factors in cybersecurity: A systematic literature review," in *2021 International Conference on Computational Performance Evaluation (ComPE)*, Dec 2021, pp. 133–140.

[13] E. Metalidou, C. Marinagi, P. Trivellas, N. Eberhagen, C. Skourlas, and G. Giannakopoulos, "The human factor of information security: Unintentional damage perspective," *Procedia - Social and Behavioral Sciences*, vol. 147, pp. 424–428, 2014, 3rd International Conference on Integrated Information (IC-ININFO). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877042814040440

[14] M. J. Dupuis, R. E. Crossler, and B. Endicott-Popovsky, "Measuring the human factor in information security and privacy," in *Proceedings of the 2016 49th Hawaii International Conference on System Sciences (HICSS)*, ser. HICSS '16. USA: IEEE Computer Society, 2016, p. 3676–3685. [Online]. Available: https://doi.org/10.1109/HICSS.2016.459

[15] J. Hielscher, U. Menges, S. Parkin, A. Kluge, and M. A. Sasse, ""employees who don't accept the time security takes are not aware enough": The CISO view of human-centred security," in *Proceedings of the 32nd USENIX Conference on Security Symposium*, ser. SEC '23. USA: USENIX Association, 2023.

[16] J. Hielscher, A. Kluge, U. Menges, and M. A. Sasse, "Taking out the trash: Why security behavior change requires intentional forgetting," in *Proceedings of the 2021 New Security Paradigms Workshop*, ser. NSPW '21. New York, NY, USA: Association for Computing Machinery, 2022, p. 108–122. [Online]. Available: https://doi.org/10.1145/3498891.3498902

[17] NIST, "Vulnerability metrics," NIST, 2022. [Online]. Available: https://nvd.nist.gov/vuln-metrics/cvss

[18] M. Bada, A. M. Sasse, and J. R. C. Nurse, "Cyber security awareness campaigns: Why do they fail to change behaviour?" *ArXiv*, vol. abs/1901.02672, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:17775217

[19] F. Deusch and T. Eggendorfer, "Strafbarkeit von IT-Sicherheitsforschern und Pentestern (translated: Criminal liability of IT-security researchers and pentesters)," *K&R*, vol. 2023, no. 10, pp. 649–656, 10 20223.

[20] "Log4shell (cve-2021-44228, cve-2021-45046)," 2021. [Online]. Available: https://log4.sh/

[21] NIST, "cyber resiliency." [Online]. Available: https://csrc.nist.gov/glossary/term/cyber_resiliency

[22] K. Hausken, "Cyber resilience in firms, organizations and societies," *Internet of Things*, vol. 11, p. 100204, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660520300408

[23] I. Linkov and A. Kott, *Fundamental Concepts of Cyber Resilience: Introduction and Overview*. Cham: Springer International Publishing, 2019, pp. 1–25. [Online]. Available: https://doi.org/10.1007/978-3-319-77492-3_1

[24] F. Björck, M. Henkel, J. Stirna, and J. Zdravkovic, "Cyber resilience – fundamentals for a definition," in *New Contributions in Information Systems and Technologies*, A. Rocha, A. M. Correia, S. Costanzo, and L. P. Reis, Eds. Cham: Springer International Publishing, 2015, pp. 311–316.

[25] OpenBSD, "OpenBSD security." [Online]. Available: http://www.openbsd.org/security.html

[26] OWASP, "SAMM model overview," OWASP. [Online]. Available: https://owaspsamm.org/model/

[27] R. Böhme and F. C. Freiling, *On Metrics and Measurements*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 7–13. [Online]. Available: https://doi.org/10.1007/978-3-540-68947-8_2

[28] F. Deusch and T. Eggendorfer, "Penetrationstest bei Auftragsverarbeitung (translated: Penetrationtesting sub-processors)," *K&R*, vol. 2018, no. 04, pp. 223–230, 2018.

[29] V. Y. Shen, T.-J. Yu, S. M. Thebaut, and L. R. Paulsen, "Identifying error-prone software an empirical study," *IEEE Trans. Softw. Eng.*, vol. 11, no. 4, p. 317–324, apr 1985. [Online]. Available: https://doi.org/10.1109/TSE.1985.232222

[30] T. Khoshgoftaar, J. Munson, G. Richardson, and B. Bhattacharya, "Predictive modeling techniques of software quality from software measures," *IEEE Transactions on Software Engineering*, vol. 18, no. 11, pp. 979–987, nov 1992.

[31] J. P. Cavano and J. A. McCall, "A framework for the measurement of software quality," in *Proceedings of the Software Quality Assurance Workshop on Functional and Performance Issues*. New York, NY, USA: Association for Computing Machinery, 1978, p. 133–139. [Online]. Available: https://doi.org/10.1145/800283.811113

[32] W. Florac, "Software quality measurement: A framework for counting problems and defects," Carnegie Mellon University, Software Engineering Institute's Digital Library, Carnegie Mellon University, Tech. Rep. CMU/SEI-92-TR-022, Sep 1992. [Online]. Available: https://insights.sei.cmu.edu/library/software-quality-measurement-a-framework-for-counting-problems-and-defects/

[33] R. Savola, "On the feasibility of utilizing security metrics in software-intensive systems," in *IJCSNS International Journal of Computer Science and Network Security*, vol. 10, no. 1, 01 2010.

[34] R. M. Savola, "Strategies for security measurement objective decomposition," in *2012 Information Security for South Africa*, 2012, pp. 1–8.

[35] C. Wang and W. A. Wulf, "Towards a framework for security measurement," in *Proceedings of the 20th NISSC*, 1997. [Online]. Available: https://api.semanticscholar.org/CorpusID:14546880

[36] S. Islam and P. Falcarin, "Measuring security requirements for software security," in *2011 IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS)*, Sep. 2011, pp. 70–75.

[37] G. Hatzivasilis, I. Papaefstathiou, and C. Manifavas, "Software security, privacy, and dependability: Metrics and measurement," *IEEE Software*, vol. 33, no. 4, pp. 46–54, July 2016.

[38] I. Eusgeld, F. Fraikin, M. Rohr, F. Salfner, and U. Wappler, *Software Reliability*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 104–125. [Online]. Available: https://doi.org/10.1007/978-3-540-68947-8_10

[39] J. Müller-Quade, L. Benz, C. F. Fruböse, C. Martin, and J. Mechler, "Quantification of security." [Online]. Available: https://crypto.iti.kit.edu/quantification-of-security.php

[40] Z. Benenson, U. Kühn, and S. Lucks, *Cryptographic Attack Metrics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 133–156. [Online]. Available: https://doi.org/10.1007/978-3-540-68947-8_12

[41] T. Holz, *Security Measurements and Metrics for Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 157–165. [Online]. Available: https://doi.org/10.1007/978-3-540-68947-8_13

[42] 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.

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

[44] G. Klein, J. Andronick, K. Elphinstone, G. Heiser, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood, "SeL4: Formal verification of an operating-system kernel," *Commun. ACM*, vol. 53, no. 6, p. 107–115, jun 2010. [Online]. Available: https://doi.org/10.1145/1743546.1743574

[45] A. für Innovation in der Cybersicherheit, "Tender on övit – ökosystem vertrauenswürdige it. beweisbare cybersicherheit (translation: Ecosystem trustworthy IT. proofable cyber-security)." [Online]. Available: https://www.cyberagentur.de/ausschreibungen/

[46] A. Smith, *An Inquiry into the Nature and Causes of the Wealth of Nations*. n/a, 1776.

[47] F. Daumann, *Die Rolle der Evolution in Hayeks Konzept freiheitssichernder Regeln (Translation: The role of evolution in Hayek's concept of freedom-securing rules*. Berlin, Boston: De Gruyter Oldenbourg, 2001, pp. 83–102. [Online]. Available: https://doi.org/10.1515/9783110506129-007