Special Issue

Dominik Brodowski*

(Ir-)responsible disclosure of software vulnerabilities and the risk of criminal liability

DOI 10.1515/itit-2015-0014
Received April 1, 2015; accepted September 20, 2015

Abstract: Whenever security researchers find an exploitable vulnerability in software, they face several options: Common examples are contacting the author, maintainer or vendor of the software in private (limited disclosure), publishing information on the vulnerability – possibly with a proof-of-concept of how to exploit it – (full disclosure), a combination of both (responsible disclosure), or even selling information on the vulnerability to third parties. In this article, I will discuss the legal obligations and the legal limitations to the various, typical options available to IT security researchers, with a specific focus on how they may comply with German and European criminal law.

Keywords: Full disclosure, responsible disclosure, handling of software vulnerabilities, liability for software exploitation.

ACM CCS: Software and its engineering → Software creation and management, Security and privacy → Software and application security, Legal aspects of computing

1 Introduction

1.1 Software vulnerabilities as a threat and a gift

During forensic analysis of computer systems, it is all but rare that malware implementing new forms of exploits is discovered [4]: Let us assume that during a forensic analysis of such malware, a previously unknown *vulnerability* in a computer application was found. A vulnerability is a flaw in the software that may be utilized (exploited) to circumvent checks and safeguards against unauthorized access [33] and may allow for a *privilege escalation*.

***Corresponding author: Dominik Brodowski,** Chair for Criminal Law (Prof. Burchard), Frankfurt (Main), e-mail: law@dominikbrodowski.net

This knowledge gained by the forensic expert is both a threat and a gift: It is a threat in the hands of criminals, who may use this knowledge to attack those information systems which run vulnerable versions of the software. It is a gift in the hands of the developers of the software, who may resolve the issue in new versions of the software. And it is a gift in the hands of the forensic expert, who may, depending on the situation, receive a bounty payment, or be in the position to demand a payment in exchange for disclosing his knowledge. Recipients of the information, who may be willing to pay for it, may be the developers of the software, criminals (on *black markets*) or even state officials who seek information on previously unknown vulnerabilities (so-called *zero-days*) [29].

1.2 Common disclosure strategies

But what *must*, what *may* the forensic expert or – more generally: an IT security specialist – do from a legal perspective? Before looking at these questions, it is important to distinguish the common types of behavior found in practice.

1.2.1 Full disclosure

Full disclosure means that the researcher publishes his or her knowledge about the vulnerability without informing anyone in advance. It is a practice propagated by those who fear that this vulnerability may already be exploited in practice, calling for swift actions of the software authors and maintainers (by fixing the vulnerability) *and* the users of the affected software (by mitigating the vulnerability, by determining past attacks or by temporarily disabling the software) [31].

1.2.2 Limited disclosure

Limited disclosure (or *no disclosure*) means informing only the authors or maintainers of the affected software. It is then upon them to decide whether to notify the users of the discovered vulnerability, e. g. as soon as an updated version of the software is available; and it relies on the software vendor to have incentives to fix the issue and to release an updated version to its clients.

1.2.3 Responsible disclosure

Responsible or *coordinated* disclosure combines both strategies: As a first step, only the developers, maintainers or publishers of the affected software (and, possibly, distinguished users) are informed of the vulnerability, so that they have the chance to create updated versions of the software and coordinate the release of bugfixes and of informing the public. Then – and at latest after a lapse of time determined by the IT security researcher (typically 60 days) – the knowledge of the vulnerability is disclosed to the public [14].

1.2.4 Black-market disclosure

Finally, black-market disclosure shall mean that the vulnerability is sold on a black market, i. e. through channels where it is obvious that the vulnerability will later on be exploited by third parties.

1.3 Research questions and limitations

To analyze the legal implications of responsible disclosure of software vulnerabilities, I will first discuss whether IT security researchers have a *duty* to disclose and/or to fix security-related vulnerabilities in software (Section 2). Then I will discuss the risks of liability stemming from the information being (mis-)used by third parties, with a specific focus on criminal liability (Section 3) and the risks of liability for disclosing intellectual property (Section 4). In combination with a discussion of the necessity to disclose the vulnerability – which may offer a defense to any liability – (Section 5), this will allow me to reach conclusions on the legal risks of the aforementioned disclosure strategies (Section 6).

I will exclude questions on breaking the law during or preceding the software analysis. While my analysis will use German legal standards as a starting point, the relevant laws and provisions are strikingly similar in many jurisdictions and largely influenced by European [20, 21] and international [15] provisions. In order to allow for a more in-depth analysis of the fundamental questions at stake, I will need to exclude discussing any specialities of other legal orders; for U. S. law, see [17]. Moreover, I will exclude questions on moral obligations, on the ethics of hacking and on what disclosure model serves whom best, i. e. on the economic analysis of the disclosure options [1, 12, 31, 32]. Last but not least, I want to emphasize that this article can only provide a broad overview over the risks and opportunities, but cannot replace legal advice which takes the specific situation of a particular case into account.

2 On a duty to disclose and/or to fix vulnerabilities in software

One core differentiation in law is between acts and omissions. While a person is generally responsible – and potentially criminally liable – for his or her acts, a responsibility – and criminal liability – for omissions is the exception. Grounds for a *duty to act* may be found, in very general terms, in some statutes (such as in duty-to-rescue laws, e. g. § 323c StGB[1]) and in contractual obligations. Furthermore and according to German criminal law doctrine, a duty to act exists if someone created or contributed to a risk of damage to a third party (*Ingerenzverantwortlichkeit*) [27].

What does this mean relating to software vulnerabilities – is there a duty to disclose and/or to fix such issues? To answer these questions, several common situations need to be distinguished.

2.1 Example case 1: Random find of a security-related vulnerability

Let us assume that freelance IT security researcher R takes a look at a certain Open-Source Software in his or her spare time and discovers a vulnerability. In such a case and under typical Open-Source Software licenses such as the GPL [22], R has no contractual obligation to share this knowledge with anyone. R did not create or contribute to a risk beforehand, and there are no statutes obliging R to report security vulnerabilities to authorities or to software developers or vendors.

1 German Criminal Code – Strafgesetzbuch. An official English translation is available at [9].

2.2 Example case 2: Duties and obligations of Free- and Open-Source Software maintainers

Let us assume similar conditions as in case 1, but R developed and published the software which contains the vulnerability under a typical Free- and Open-Source Software license such as the GPL [22]. As R does not want to be embarrassed, R declines to update the software once he or she finds the vulnerability, even though R is certain that criminals will utilize the vulnerability to gain unauthorized access to data – a criminal offense under § 202a I StGB. Criminal C has discovered the vulnerability on his or her own and gains unauthorized access to several computer systems.

2.2.1 Criminal-law aspects

Within the criminal-law literature, some authors argue that the mere creation of the risk, such as the publication of a vulnerable software by R, leads to a duty to act [27, and others]. However, others [2, § 15 at 65 ff., and others] and especially jurisprudence [7, and elsewhere] argue that such a duty can only arise when the creation of the risk was by itself contrary to R's duties – which is not the case if the software was developed and published according to common software development practices.

In the end, however, it is clear that R has no duty to act based on *criminal* law, as R cannot be held criminally liable for the unauthorized access of computer systems by C: This unauthorized access – an intentional crime – was committed by C, without any connection to R. Therefore, this crime cannot be attributed [25, at 253, and others] to R.

2.2.2 Contract and tort-law aspects

As R only licenses his code, but does not sell it, responsibilities stemming from the sharing of the software are marginal. Relating to tort law, the result is less clear: While the author of a software – especially, but not limited to a commercial context – may be obliged to monitor whether the software causes any risks or damages when being used (*Produktbeobachtungspflicht*, § 823 I BGB,[2] [34, at 564]), the author of a software has a duty to avert the commission of intentional crimes by third parties only in exceptional cases [28, at 928].

2.2.3 Conclusion

Therefore and in general terms, the result is the same as in case 1: R has no contractual or criminal-law obligation to disclose or to fix the vulnerability.

2.3 Example case 3: Duties and obligations in a commercial context

Let us assume that R developed and sold software to customers in exchange for a fee of 100 € each. Later on, R discovers a vulnerability in the source code of this software, but declines to amend the situation or to inform his or her customers about the vulnerability, even though he or she foresees that the vulnerability is utilized by criminals. Criminal C has found out about the vulnerability on his or her own and gains unauthorized access to several computer systems.

In common software licensing contracts, there are no contractual obligations regarding duties to resolve vulnerabilities. Exceptions to this rule may be found in some software contracts relating to customized software, or in contracts concerning the development of such software for a specific customer. Apart from contractual obligations or the lack thereof, the situation is similar to case 2: As the criminal offense of unauthorized access was committed intentionally by a third party and cannot be attributed to R, R has no criminal or civil duty to prevent such a crime.

2.4 Summary

Apart from exceptional cases, there are no *criminal-law* requirements to disclose or to fix vulnerabilities in software. Contractual or tort-law requirements may exist – depending on the specific situation – for developers and vendors of software, but also for IT security researchers who have entered a contract with another party to specifically analyze software for security-related vulnerabilities.

3 Information as a weapon

Let us now assume that IT security researcher R voluntarily disclosed the information about the vulnerability to a third party. In general terms, he or she may do so in

2 German Civil Code – Bürgerliches Gesetzbuch. An official English translation is available at [9].

three ways: first, by a verbal description of the problem; second, by a bug-fix (patch) which rectifies the situation; third, by a *proof-of-concept*, i. e. a small example program which shows how and that the vulnerability may be exploited [17].

All types of information – from the verbal description to the proof-of-concept – may enable others to exploit the vulnerable software; the proof-of-concept makes it most easy for third parties. Therefore, any form of disclosure of the information can be a *causation* for any later exploit of the vulnerability – e. g. an unauthorized access to a computer system by criminal C. Can R be held liable for the crime committed by C?

3.1 Cooperation, conspiracy and inchoate crimes

3.1.1 Clear criminal intent

A first and evident basis for criminal liability of R exists if he or she shares the information on the vulnerability with other persons in order to facilitate the joint commission of computer crimes, or to facilitate the commission of computer crimes by others. In such cases, R may be held criminally liable, either as co-perpetrator (§ 25 I StGB), as abetter to the crime committed by someone else (§ 27 StGB), or – under exceptional cases – as conspirator (§ 30 II StGB).

3.1.2 Awareness of the risk

The situation becomes much less clear, however, once the commission of a crime was not R's *aim* when he or she shared the information on the vulnerability – but R was nonetheless aware of the risk that this information could and would be misused by someone else to commit a crime. Can R be held responsible for aiding to this crime committed by someone else?

Criminal-law aspects. To be held criminally liable for the inchoate crime under § 27 StGB, the subjective elements are surprisingly low: The person aiding to a crime neither needs to know exactly about the plans of the main perpetrator, nor does he or she need to know the main perpetrator at all. Instead, jurisprudence considers it sufficient that a person hands over a tool regularly utilized for crimes, and knowingly accepts the risk that someone else utilizes this tool for such a crime [6, at 138]. However, there is general consensus that neutral professional activities – such as selling a knife in a kitchen store, even if it is bought by a potential murderer – should not lead that easily to a criminal liability. The details are heavily disputed [3]: Jurisprudence distinguishes based on whether or not the perpetrator actually sought to facilitate a crime [8]. In legal literature, some call for taking professional standards and regulations into account [24, 30].

Considering that researchers most often aim at encouraging software developers and vendors to make their products more secure and resilient to attacks, they should not need to fear criminal liability, as long as they follow either full or responsible disclosure procedures – which both seem to be accepted professional standards (see Section 1.2 above). Nevertheless, let me add a word of caution: These grounds for exclusion have not yet been tested in court in relation to the disclosure of security vulnerabilities. Moreover, there is a distinct risk that prosecutors and courts consider the aim of (full) disclosure to include harming the software vendor and downplaying the risks associated with the disclosure. This could, therefore, lead to criminal liability in case the disclosed vulnerability – and especially a proof-of-concept – was (mis-)used by third parties to commit crimes.

Tort liability. As to tort liability: Such a liability exists – very generally speaking – whenever someone is also criminally liable for a crime (§ 823 II BGB). In addition, such a liability exists for acts committed by third parties according to § 823 I BGB only insofar as the disclosure violated professional standards which specifically aim at preventing crimes by third parties [28, at 928].

Conclusion. To summarize: As long as the aim of the disclosure is to encourage software developers and vendors to correct the vulnerability, and as long as professional standards and regulations are taken into account when disclosing the vulnerability, the risks of criminal and tort liability are limited.

3.2 Perparatory offenses and the risks of dealing with harmful software

Criminal justice systems show a general interest in prohibiting dealing with dangerous tools. For example, there are wide-ranging criminal prohibitions on creating, buying, selling, importing, exporting, or owning weapons, drugs and medicine, and tools to forge money. As some software may also be utilized to gain unauthorized access to information systems or for similar criminal means, criminal-law provisions in Germany and abroad [15, 21] also prohibit any dealing in "computer programs the purpose of which is to commit [a computer-related crime]"

(§ 263a III StGB, see also §§ 202c I, 303a III, 303b V StGB); similar provisions exist in copyright law.

What does constitute such a "*computer program*"? It is clear that a factual description of the vulnerability or a patch to the source code of the software in order to fix the vulnerability do not constitute computer programs by themselves, as they do not contain sufficient instructions for immediate execution by the information system, but only more or less explicit hints of how to update these instructions. A proof-of-concept is, however, a "computer program", even if the source code first has to be compiled into machine code to run on the target system.

Is a proof-of-concept also a program "*the purpose of which is to commit [a computer-related crime]*"? It does not matter whether additional programs are required to commit a crime. However, most consider defining the purpose of a computer program to be difficult at least, if not impossible (programs provide functions, but do not pursue a purpose [35]). The German Federal Constitutional Court calls for a three-step approach [10]: First, the program must be able to further a crime – which is evidently the case for a proof-of-concept for exploiting a vulnerability. Second, the intent of the author (or modificator) of the program has to be determined – did he intend to provide a tool for criminal utilities, or for other, legitimate purposes? Third, this intent of the author must be observable either in the program itself or in the distribution of the program.

Last but not least, these preparatory offenses in German criminal law contain a subjective requirement that the creation, acquisition, etc. of such a program must be done to prepare the commission of a specific compter-related. According to jurisprudence, this means that the person dealing with such software is aware of the possibility that this program will be misused by someone he or she shares it with, but accepts this risk [10]. According to others, the crime to be committed later on and its details (who committs it when and against whom) must at least be known in broad terms to the person dealing with the software [26, at 28] – in line with the (stricter) standard of European and international law [15, 21].

4 Vulnerabilities and the protection of intellectual property

4.1 Free- and Open-Source Software

An additional issue one needs to be aware of when disclosing a vulnerability is the criminal and civil-law protection of intellectual property. This does not pose a problem if the source code in question is already available to the public, such as is the case for Free- and Open-Source Software. For such software, it is perfectly legal to analyze the code and to report on its functionality and on vulnerabilities therein. The sharing of code and the submission of patches to correct the vulnerability, however, requires a justification in light of copyright law; this can – most often – be found in the license under which the software has been obtained, such as the GPL [22].

4.2 Unlawful analysis of software or unlawful means for analysing software

More difficulties arise in situations where the software in question is closed-source, and/or where the analysis of software by IT security researchers was unlawful by itself, because they did not have the right to reverse engineer or to disassemble the code, because they circumvented security mechanisms themselves, or because the means they utilized for the analysis were of unlawful or dubious origin. I will not discuss the criminal-law implications of the software analysis by itself here. However, the dubious prerequesites of the discovery cause implications for the further steps of limited, responsible or full disclosure:

Coypright protection. First of all, software enjoys copyright protection. This means that any sharing of software or parts thereof in a disclosure of a vulnerability requires a justification. This may be the agreement of the copyright holder. Otherwise, such a justification may only be found in *fair use* (e. g. in US copyright law) or in the *right to quotation* (§ 51 UrhG[3]). As long as only small parts of the code are shared and proper attribution is given, this risk of liability may at least be reduced.

Trade-secrets law. The decisive difference in terms of disclosure, though, rests in trade-secrets law: Communicating trade and industrial secrets to third parties constitutes a criminal offense and may lead to civil liability, if serving "the purposes of competition, for personal gain, for the benefit of a third party, or with the intent of causing damage to the owner of the business" (§ 17 II Nr. 2 UWG[4]).

3 Act on Copyright and Related Rights – Gesetz über Urheberrecht und verwandte Schutzrechte. An official English translation is available at [9].

4 German Act Against Unfair Competition – Gesetz gegen den unlauteren Wettbewerb. An official English translation is available at [9].

The German provision is interpreted in such a way that "trade and industrial secrets" are all non-obvious aspects relating to the trade and business which its owner legitimately wants to keep secret [5]. In relation to closed-source software, such an interest of the software author regarding the functioning of the program may easily be presumed: Disclosing the non-obvious fact that this software contains a security-related vulnerability may well cause detriments to the trade or business. Moreover, detailed information on the operation of a program – which may be necessary aspects of a disclosure in order to explain the vulnerability – may be considered to be a "trade and industrial secret" within the meaning of this provision.

Furthermore, the subjective requirements of the law are actually quite easy to meet: Personal gain includes at least financial gain, but – according to some – also immaterial advantages [16] such as job offers. And the intent to cause damage may easily be assumed if the disclosure to the public ridicules the software author.

In a similar vein, trade-secrets law has been the reason that certain IT security researchers were prohibited to disclose their findings on vulnerabilities [32, at 183 f.]. For example, a UK court issued an injunction against British and Dutch researchers to forbid them publishing on a vulnerability in an automobile immobiliser system, as the court assumed that the software utilized to find the vulnerability had been of unlawful origin, and although the manufacturer of the immobiliser system was informed nine months in advance of the publication [11, 19].

5 On the necessity to disclose vulnerabilities

What can be brought to the rescue of the IT security researcher? Germany and several other jurisdictions do not know a generic *freedom of speech* exception, but only a *freedom of sharing one's opinion* (Art. 5 GG[5]) which offers far lesser protection regarding the sharing of facts, especially if they are protected by a general law [23] – such as is the case for trade and industrial secrets. By itself, this is therefore a weak justification for the disclosure of information contrary to criminal or civil laws.

The risks of criminal and tort liability may be mitigated insofar as the IT security researcher can refer to a *defense*, and especially to the *defense of necessity*. This defense to liability, which is more or less accepted worldwide, requires that the chosen action is the best means available in order to avert an imminent danger to a person and his legal interests. Furthermore, the action chosen must outweigh the risk associated with not taking action (§ 34 StGB) [18].

Regarding the disclosure of a software vulnerability and depending on the specific situation, an imminent danger may be present to information systems used by many, because it might be (further) exploited unless the vulnerability is fixed or other mitigating steps are taken immediately. The options available to the IT security researcher to mitigate the danger are manifold. Informing the public – and thereby informing all potentially affected users – may be a useful action to mitigate the situation, e. g. if the users can take swift action to disable the affected software or change the software configuration. However, such a disclosure may also *increase* the risk that the vulnerability is exploited, and it may require the public sharing of trade or business secrets of the software authors. These aspects must be balanced against each other, depending on the specific circumstances of the particular case.

6 Full, responsible or limited disclosure?

Based on these considerations, I will now turn to some conclusions on the typical modes of disclosure (limited, responsible and full as well as black-market disclosure).

6.1 Black-market disclosure

It is evident that the disclosure of vulnerability on a black market poses a severe risk of criminal and tort liability to *all* actors: A wide range of grounds for liability are available to the authorities, rangig from inchoate offenses to preparatory offenses, as the intent of the actor is evidently malicious. Even selling the information to authorities only – e. g. to an intelligence agency which seeks information on "zero-days" to remotely infiltrate information systems – may not be of help, especially in cross-border situations.

6.2 Limited disclosure

Informing the authors, distributors and/or vendors of software about a vulnerability hardly poses a risk of criminal, contract or tort liability. This counts true even if it includes

5 Basic Law for the Federal Republic of Germany – Grundgesetz für die Bundesrepublik Deutschland. An official English translation is available at [9].

sharing a proof-of-concept of an exploit, as such a program clearly follows the aim of demonstrating the vulnerability and is shared with a limited circle of trustful persons. Furthermore, the person sharing the information may legitimately trust that this information is not passed on to criminals, but is only used to remedy the situation. Last but not least, limited disclosure causes less risks in terms of trade-secret protection.

Pitfals rest, however, when linking the disclosure with threats (such as to selling the vulnerability to criminals) and/or demands (such as requesting 10 000 € in exchange for the limited disclosure of the exploit). For the US jurisdiction, some recommend to avoid making any demands at all, as they could lead to criminal or civil liability [17]. In my opinion and regarding German jurisdiction, though, not all demands and/or threats linked to the disclosure are illegal per se: A link between a threat and a demand is only unlawful if either the threat or the demand are unlawful by themselves, or if the link between the threat and the demand are unlawful [36, at 11 and 34 ff.]. In general, the IT specialist has no duty to share his knowledge with the software author. Hence, linking the information-sharing with a resonable demand for money is not a matter for criminal courts, but general business practice.

6.3 Full disclosure

In case of full disclosure, sharing a proof-of-concept of the exploit poses a severe risk of criminal liability because of the preparatory offenses in §§ 202c I, 263a III, 303a III, 303b V StGB and similar offenses in other criminal justice systems. This risk may be lessened, though, if the purpose of the proof-of-concept is clearly limited to demonstrate the severity of the vulnerability, and if it serves as a test-case whether the vulnerability (still) exists in the software. Then, the software evidently (also) include legitimate functions, which are manifest in the software itself. This legitimate functionality should be made clear in the source code and on any webpage or mailing list message that links to this program. Furthermore, all such messages should be explicitly directed at remediating the situation (and formulated in such a manner).

The risk of liability for full disclosure may be further mitigated if the defense of necessity is available. Factors which may make full disclosure necessary include ongoing widespread exploits (which call for swift, mitigating actions by the users), the inability to contact the software author directly or, depending on the specific circumstances, the software author declining to take action by him- or herself after having been informed previously in private.

6.4 Responsible disclosure

The first step of responsible disclosure – informing the authors, etc. – is equivalent to limited disclosure; therefore, this does not pose severe risks of liability by itself. The second step of responsible disclosure – releasing information on the vulnerability to the public – can be evaluated similarily to the case of full disclosure, as discussed previously. Risks of liability are present, however, in the link between these two steps.

If the IT security researcher states that he will release the information to the public *unless* the software developer/vendor releases an updated version within a certain timeframe (and/or unless he is paid a certain amount of money), he faces the risk of criminal liability because of coercion or blackmail (§§ 240, 253 StGB): The publication of a software vulnerability may easily constitute a serious harm to the software author/vendor, and this is exactly what the IT security researcher threatens with, unless his or her demands are met.

Again, such a link between a threat and a demand is only unlawful if either the threat or the demand are unlawful by themselves – which is not the case, as there are legitimate reasons for full disclosure –, or if the link between the threat and the demand is unlawful [36, at 11 and 34 ff.]. In my opinion, however, at least the link to requiring a bugfix is legitimate: The risk that someone else finds the exact same vulnerability and exploits it for criminal purposes increases with each day. If the software authors, however, do not fix the vulnerability within a reasonable time frame (e. g. 60 days), they have demonstrated that they themselves behave irresponsibly. Therefore, with each day that passes by, the reasons for disclosure to the public gain importance: Other users of the software have a strong need to be informed about the irresponsible behavior of the software authors and of ways to mitigate any immenent exploit of the vulnerability. Because of these reasons, the grounds for a defense of necessity increase day by day.

7 Conclusion

The disclosure of vulnerabilities is closely linked with severe risks of criminal and tort liability in Germany and in other affected jurisdictions. A careful design of the disclosure – in terms of *when* to disclose *what* to *whom* – allows to mitigate these risks down to an acceptable level. Risks are lowest as long as Free- and Open-Source software is involved, no detailed proof-of-concept code is shared, the maintainers of the software are contacted in private first and an agreement is reached with them on how to proceed. If the maintainers are uncooperative, however, and fail to correct the vulnerability within a reasonable timeframe (such as 60 days for software vulnerabilities), full or responsible disclosure will be justified in most cases. Disclosing vulnerabilities in black markets, or writing proof-of-concepts which do not manifestly aim at explaining the problem and determining whether the vulnerability is securely resolved, cause severe risks of criminal and tort liability.

References

1. A. Arora, R. Telang and H. Xu. Optimal Policy for Software Vulnerability Disclosure. In: *Management Science 54*, pp. 642–656, 2008.
2. J. Baumann, W. Mitsch and U. Weber. Strafrecht. Allgemeiner Teil. 11th ed. 2003.
3. K. Beckemper. Strafbare Beihilfe durch alltägliche Geschäftsvorgänge. In: *Jura*, pp. 163–169, 2001.
4. B. Bencsáth, G. Pék, L. Buttyán and Márk Félegyházi. The Cousins of Stuxnet: Duqu, Flame, and Gauss. In: *Future Internet 4(4)*, pp. 971-1003, 2012.
5. Bundesgerichtshof [German Federal Supreme Court]. Judgment of 1995-05-10 – 1 StR 764/94. In: *BGHSt 41*, pp. 140–144, 1996.
6. Bundesgerichtshof [German Federal Supreme Court]. Judgment of 1996-04-18 – 1 StR 14/96. In: *BGHSt 42*, pp. 135–139, 1997.
7. Bundesgerichtshof [German Federal Supreme Court]. Judgment of 1997-12-19 – 5 StR 569/96. In: *BGHSt 43*, pp. 381–407, 1998.
8. Bundesgerichtshof [German Federal Supreme Court]. Decision of 1999-09-20 – 5 StR 729/98. In: *Neue Zeitschrift für Strafrecht*, pp. 34–36, 2000.
9. Bundesministerium der Justiz und für Verbraucherschutz [German Ministry of Justice and for Consumer Protection]. Translations of Statutes/Ordiances [website]. Retrieved from http://www.gesetze-im-internet.de/Teilliste_translations.html.
10. Bundesverfassungsgericht [German Federal Constitutional Court]. Decision of 2009-05-18 – 2 BvR 2233/07, 2 BvR 1151/08, 2 BvR 1524/08. In: *BVerfGK 15*, pp. 491–509, 2009.
11. R. Carolina and K. G. Paterson. Megamos Crypto, Responsible Disclosure, and the Chilling Effect of Volkswagen Aktiengesellschaft vs Garcia, et al. [Comment of 2013-08-28]. Retrieved from http://www.isg.rhul.ac.uk/~kp/Carolina-Paterson-Megamos-comment-20130828.pdf.
12. H. Cavusoglu, H. Cavusoglu and S. Raghunathan. Emerging Issues in Responsible Vulnerability Disclosure. In: *Proceedings of the Fourth Annual Workshop on Economics of Information Security (WEIS 2005)*. Retrieved from http://www.infosecon.net/workshop/pdf/65.pdf.
13. Chaosradio. CR193. Responsible Disclosure and Full Disclosure [Podcast]. Retrieved from http://chaosradio.ccc.de/cr193.html.
14. S. Christey. Responsible Vulnerability Disclosure Process. Draft IETF RFC of February 2002. Retrieved from https://tools.ietf.org/html/draft-christey-wysopal-vuln-disclosure-00.
15. Council of Europe. Convention on Cybercrime [signed in Budapest on 2001-11-23]. Retrieved from http://www.conventions.coe.int/Treaty/en/Treaties/Html/185.htm.
16. H. Diemer. § 17 UWG. In: *G. Erbs (founder) and M. Kohlhaas (continuator). Strafrechtliche Nebengesetze. 177th suppl.*, 2009.
17. Electronic Frontier Foundation. Coders' Rights Project Vulnerability Reporting FAQ [website]. Retrieved from https://www.eff.org/de/issues/coders/vulnerability-reporting-faq.
18. V. Erb. Der rechtfertigende Notstand. In: *Juristische Schulung*, pp. 17–22, 2010.
19. England and Wales High Court (Chancery Division). Judgment of 2013-06-25 – [2013] EWHC 1832 (Ch) – Volkswagen Aktiengesellschaft ./. Garcia et al. Retrieved from http://www.bailii.org/ew/cases/EWHC/Ch/2013/1832.html.
20. European Union. Council Framework Decision 2005/222/JHA of 24 February 2005 on attacks against information systems. In: *Official Journal of the European Union, L 2005/69*, pp. 67–71, 2005.
21. European Union. Directive 2013/40/EU of the European Parliament and of the Council of 12 August 2013 on attacks against information systems and replacing Council Framework Decision 2005/222/JHA. In: *Official Journal of the European Union, L 2013/218*, pp. 8–14, 2013.
22. Free Software Foundation. GNU General Public License. Version 3, 29 June 2007. Retrieved from https://www.gnu.org/copyleft/gpl.html.
23. W. Frenz. Die Meinungs- und Medienfreiheit. In: *Jura*, pp. 198–202, 2012.
24. W. Hassemer. Professionelle Adäquanz. In: *wistra*, pp. 81–87, 1995.
25. B. Heinrich. Strafrecht. Allgemeiner Teil. 3rd ed. 2012.
26. E. Hilgendorf. § 202c StGB. In: *H. W. Laufhütte, R. Rissing-van Saan and K. Tiedemann (eds.): Strafgesetzbuch. Leipziger Kommentar. 12th ed.*, 2010.
27. K. Kühl. Das Unterlassungsdelikt. In: *Juristische Arbeitsblätter*, pp. 507–512, 2014.
28. D. Looschelders. Schuldrecht. Allgemeiner Teil. 12th ed. 2014.
29. N. Perlroth and D. E. Sanger. Nations Buying as Hackers Sell Flaws in Computer Code. In: *New York Times*, pp. A1, 2013-07-14.
30. M. Rabe von Kühlewein. Strafrechtliche Haftung bei vorsätzlichen Straftaten anderer. In: *Juristenzeitung*, pp. 1139–1146, 2002.
31. B. Schneier. Full Disclosure of Security Vulnerabilities a 'Damned Good Idea' [Blog post, January 2007]. Retrieved from https://www.schneier.com/essays/archives/2007/01/schneier_full_disclo.html.
32. M. Schwalb. Exploit Derivatives & National Security. In: *Yale J. L. & Tech. 9*, pp. 162–192, 2006–2007.
33. R. Shirey. Internet Security Glossary, Version 2. IETF RFC 4949.
34. G. Spindler. § 823 BGB. In: *H. G. Bamberger and H. Roth (eds.): Beck'scher Online-Kommentar BGB. 34th ed.*, 2015.
35. B. Valerius. Anwendungsbereich des und Zweck eines Computerprogramms i. S. d. § 202c StGB. In: *Juristische Rundschau*, pp. 84–86, 2010.
36. J. Vogel. § 253 StGB. In: *H. W. Laufhütte, R. Rissing-van Saan and K. Tiedemann (eds.): Strafgesetzbuch. Leipziger Kommentar. 12th ed.*, 2010.

Bionotes

Dr. iur. Dominik Brodowski
Chair for Criminal Law et al.
(Prof. Burchard), HPF EXC 15,
60629 Frankfurt am Main, Germany,
Tel.: +49-69-798-31476
law@dominikbrodowski.net

Dominik Brodowski is a senior researcher at the University of Frankfurt (Main) and a lecturer (*Lehrbeauftragter*) at Albstadt-Sigmaringen University in a master's course on digital forensics. As a graduate of the University of Tübingen and University of Pennsylvania Law School, his professional activities focus on European criminal law, criminal procedure and its interaction with the realities of technology.