A mixed methods probe into the direct disclosure of software vulnerabilities

Ruohonen Jukka, Hyrynsalmi Sami, Leppänen Ville

DOI: 10.1016/j.chb.2019.09.028

A Mixed Methods Probe into the Direct Disclosure of Software Vulnerabilities

Jukka Ruohonen[a,*], Sami Hyrynsalmi[a,b], Ville Leppänen[a]

[a] *Department of Future Technologies, University of Turku, FI-20014 Turun yliopisto, Finland*
[b] *Pori Department, Tampere University of Technology, P.O. Box 300, FI-28101 Pori, Finland*

Abstract

Software vulnerabilities are security-related software bugs. Direct disclosure refers to a practice that is widely used for communicating the confidential information about vulnerabilities between two parties, vulnerability discoverers and software producers. Building on software vulnerability life cycle analysis, this empirical paper observes the qualitative and quantitative characteristics of direct disclosure practices, focusing particularly on the historical problem related to producers' reluctance to participate in the practices. According to the results, the problem was still present in the 2000s and early 2010s—and likely is still present today. By presenting this empirical result about the underresearched phenomenon of direct disclosure of software vulnerabilities, the paper contributes to the research domain of vulnerability life cycle modeling in general and the subdomain of empirical vulnerability disclosure research in particular.

Keywords: full disclosure, public disclosure, responsible disclosure, coordinated disclosure, grace period, proof-of-concept exploit, vendor, life cycle, mixed methods

1. Introduction

A software *vulnerability* is an abstraction for a software bug that has consequences for security. Software vulnerabilities expose weaknesses in software systems. An *exploit* utilizes these weaknesses to compromise or otherwise harm the intended behavior of a software system. The concept of software vulnerability *disclosure*, in turn, relates to the behavioral processes via which confidential and sensitive information about discovered vulnerabilities are communicated by the initial *discoverer* to other actors, including particularly the associated *vendor*, the producer of the software system affected. When a disclosure process occurs privately between a discoverer and a vendor, the process can be classified as a *direct disclosure* practice. This empirical paper analyzes these direct disclosure processes by quantitative and qualitative means, using a data source that combines both vulnerabilities and exploits.

The paper contributes to the software vulnerability disclosure research tradition (Arora et al., 2010; Hahn and Govindarasu, 2012; Kranenbarg et al., 2018; McQueen et al., 2011; Mitra and Ransbotham, 2015; Ozment, 2007; Ruohonen and Allodi, 2018), continuing particularly the recent empirical work on exploit development and publication (Ablon and Bogart, 2017; Almukaynizi et al., 2018; Bullough et al., 2017; Hafiz and Fang, 2016; Sabottke et al., 2015; Sen and Heim, 2016). The contribution is straightforward: there is no existing empirical research on direct disclosure. This gap in the literature is suboptimal and unfortunate because the process has historically been the most common way to disclose vulnerabilities (Cavusoglu et al., 2007; Schneier, 2007). This type of disclosure is still important today—if not still the most prevalent way to disclose software vulnerabilities. A better understanding about direct disclosure is important because many difficult problems in software vulnerability disclosure have specifically manifested themselves through direct disclosure. With the empirical results presented, the paper also provides a few insights about the historical and contemporary reasons hindering the (direct) disclosure of software vulnerabilities.

The classical alternative to direct disclosure has been *public disclosure*, which refers to releasing of sufficient information to the public Internet from which vendors and other actors can then pick up the details. If information is released right after the discovery, a term *immediate disclosure* is also sometimes used (Sen and Heim, 2016). The same publication process is further known as a *full disclosure* in case also sensitive technical information is released (Schneier, 2007), including *proof-of-concept* (POC) code for exploitation. The ideological rationale behind full disclosure is usually simple: the public has the right to know. The ideology of *no disclosure* (Householder et al., 2017; Choi et al., 2010; Hahn and Govindarasu, 2012) builds on the logical opposite: the public does not need to know because the release of vulnerability information and POCs supposedly increases the likelihood of attacks and therefore reduces the overall well-being in the Internet. The same rationale works underneath *limited disclosure* via which only partial information is released to the public in order to

*Corresponding author.
Email address: juanruo@utu.fi (Jukka Ruohonen)

limit the exposure (Householder et al., 2017). Although many theoretical models have been proposed over the years for comparing these and other vulnerability disclosure types (Cavusoglu et al., 2007; Choi et al., 2010), the fundamental questions involved are most of all normative. In other words, vulnerability disclosure is a so-called *wicked problem* (Householder et al., 2017) that cannot be addressed with a single optimal model.

Reflecting the wickedness, there are also different *hybrid disclosure* (Cavusoglu et al., 2007) practices via which vulnerabilities are processed through *vulnerability disclosure institutions* (Ozment, 2007) and commercial *disclosure companies* acting as brokers (Algarni and Malaiya, 2014). In addition, many companies nowadays orchestrate vulnerability finding campaigns and so-called *bug bounties* either directly or via crowd-sourcing platforms (Böhme, 2006; Finifter et al., 2013; Ruohonen and Allodi, 2018). In contrast to direct, immediate, and full disclosure, these hybrid models usually operate under formal policies for processing the sensitive vulnerability information between the actors involved in the disclosure processes.

A particularly important policy aspect is the provision of a *grace period*, which refers to a safety period during which vendors have time to provide patches and advisories before the eventual public disclosure. If these grace periods are used, it is often said that *responsible disclosure* is followed. When the institutions and coordination practices embedded to hybrid disclosure are combined with responsible disclosure, the further concept of *coordinated disclosure* is often used. While grace periods have thus been institutionalized into the hybrid disclosure models (McQueen et al., 2011; Mitra and Ransbotham, 2015; Ozment, 2005), these are also a prevalent norm in direct disclosure. Norms are norms, not policies; such informal codes of conduct are not slavishly followed during direct disclosure, however.

A key factor in responsible disclosure is the time a vendor takes to respond to queries about the vulnerabilities discovered (Finifter et al., 2013). If a vendor takes a long time to respond, discoverers are likely to prefer immediate public or full disclosure instead of responsible disclosure. In fact, discoverers often use the timing of public disclosure as a coercive weapon against vendors who would otherwise refuse to correct the discovered software bugs that the disclosed vulnerabilities expose (Freeman, 2007). The same applies to disclosure institutions and companies (Arora et al., 2010; McQueen et al., 2011). There are well-known reasons for such coercive tactics. Historically vendors have been slow to correct security-related bugs in their software products (McQueen et al., 2011), often refusing to communicate with ethical discoverers (Mitra and Ransbotham, 2015), or even threatening them with legal action (Householder et al., 2017; Schneier, 2007). Henceforth, this well-known issue bundle is referred to as *the problem of reluctant vendors*. The main research question of this paper is to investigate whether the problem was still present in the 2000s and early 2010s. For studying the problem, the paper relies on both quantitative and qualitative inquiry. These research approaches and the empirical dataset are described in the opening Section 2. Results and discussion follow in Sections 3 and 4, respectively.

2. Materials and Methods

The paper relies on a single but comprehensive data warehouse for POC exploits. Qualitative and quantitative methods are mixed for analyzing a sample that contains information about direct disclosure of vulnerabilities. Both the materials and the methods require a brief elaboration.

2.1. Sample

The empirical dataset is based on a sample from Exploit Database (EDB), which presently provides the most comprehensive collection of publicly known exploits (EDB, 2016). There were 36,191 exploits in the database during the data collection in early October 2016. This large amount was first reduced by transforming the raw textual entries into lower-case letters, and then including the cases that contained either the character string *disclose* or the string *contact*. This regular expression search reduced the amount to 4,867 unique POC exploits. These were processed manually in the second step of simultaneous quantitative pre-processing and qualitative analysis.

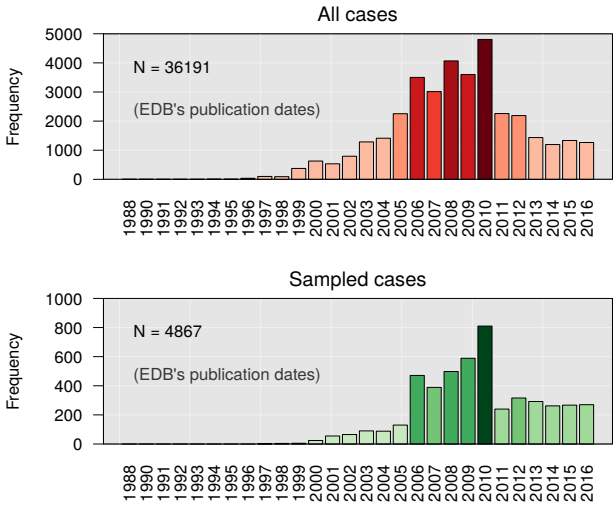

Figure 1: Years of Publication

The sample does not contain obvious biases with respect to all POC exploits that were archived in the database at the time of data collection. In particular, the dates of publication according to EDB's evaluation (which approximate the dates of public disclosure) are comparable (see Fig. 1). Most of the cases observed were published in the late 2000s and early 2010s. Therefore, the sample analyzed seems reasonable also for deducing about contemporary direct disclosure practices and processes, which, like most socio-technical phenomena, are unlikely to change rapidly.

The context of exploits adds further weight to the observed direct disclosure: the disclosed cases have not been limited to mere reporting of vulnerabilities, but these have also included the public release of POC solutions for exploitation. Although the POCs archived in EDB have not been written with actual attacking in mind (Allodi and Massacci, 2013), studying the proof-of-concepts may provide valuable and sufficient information for engineering robust exploits that can be used to reliably compromise or otherwise disturb software systems. There exists also some empirical evidence supporting this reasoning (Almukaynizi et al., 2018). Against this backdrop, POCs can be seen as a first step in a longer exploit development life cycle (Ablon and Bogart, 2017; Sabottke et al., 2015). For these same reasons, POCs used for disclosure purposes should act as a sufficient deterrent to vendors' reluctance.

2.2. Qualitative Setup

A so-called *mixed methods* approach is used for the analysis. For the present purposes, the term method is simply understood as a technique for empirical inquiry. The analysis mixes qualitative and quantitative methods for analyzing the same dataset; hence, the second term mixed.

Following an *inductive logic* (Sarker et al., 2013), a *thematic approach* (Cruzes et al., 2015; Guest et al., 2006) was adopted for the qualitative inquiry: in parallel to quantification, each sampled case was read carefully in order to locate the key thematic constructs that underline the phenomenon observed. The research process could be further classified as a *sequential exploratory* design (Carayon et al., 2015; Hesse-Biber, 2010). The key point behind the design is that the manual quantification and the qualitative analysis were done in parallel, but the actual quantitative analysis was carried out only after the qualitative analysis was already completed. Therefore, the quantitative analysis provides a reality check for the qualitative results, which are always partially reliant on subjective interpretation.

To further increase the robustness of the qualitative analysis, *the principle of transparency* (Sarker et al., 2013) is embraced by referencing the noteworthy thematic observations with cases that support the observations. The unique EDB identifiers are used for this purpose. These identifiers can be easily queried from the database's online interface for replication checks. As the numerical identifiers form a monotonically increasing sequence, these can be also used to assert that the qualitative observations are longitudinally consistent. If the historical problem of reluctant vendors was present during the whole period observed, there should be references to identifiers carrying small (old) and large (new) numerical values.

The direct disclosure cases analyzed exhibit traits of *naturally occurring data* (cf. Silverman, 2006). In other words, the quantitative and qualitative characteristics have both emerged naturally when discoverers have documented their involvement in direct disclosure processes either during the processes or in retrospect. This type of data allows to connect the thematic approach further to (digital) ethnography; the goal is to construct thick analytical descriptions by focusing on the discoverers' own point of view of the ordinary detail of direct disclosure (cf. Sharp et al., 2016). While many cases indeed provide their own intriguing narratives, it can be remarked that the raw textual data mostly contains vulnerability and security advisories, structural logs and other bookkeeping material, severity assessments, technical summaries, and POC programming code. Like in any good documents, these often follow each other logically in separate sections. These structural characteristics eased the finding of key themes with repeated patterns of meaning. For additional guarding against the misidentification of key themes, the qualitative analysis was further framed against the existing disclosure literature, and by maintaining the necessary bridges to the quantitative side of the methodological mixture. This side should be also briefly elaborated.

2.3. Quantitative Setup

The quantitative analysis builds on vulnerability life cycle modeling. While the contextual terminology is often slightly different, which is unfortunately typical for the research domain (Ozment, 2007), the manually coded life cycle variables build upon a sizable amount of existing research (Finifter et al., 2013; Garcia et al., 2014; Hahn and Govindarasu, 2012; McQueen et al., 2011; Nappa et al., 2015; Sen and Heim, 2016). Namely, manual coding was used for quantifying the following five life cycle variables:

- *Discovery* refers to the dates on which vulnerabilities were initially discovered by discoverers.
- *Disclosure* records the dates on which discoverers first contacted the vendor(s) affected.
- *Response* stores the dates on which vendors first responded to the initial contacts made by discoverers.
- *Publication* points to the dates on which raw entries were originally published in the wild.
- *Patching* keeps track of the dates on which vendors potentially released patches for the vulnerabilities.

All variables (1) are allowed to contain missing values in case no sufficient information was available in the raw textual data. The same assumption is made in many vulnerability and exploit databases (Bozorgi et al., 2010; Garcia et al., 2014). Two small shortcuts were made to increase the sample sizes: (2) in case the date of public disclosure was missing, the publication date provided in EDB was used as a proxy, and either (3) vendors' advisory release dates or commits to version control systems were used in case patch release dates were not available. In contrast to previous definitions (Ozment, 2007), (4) incorrect or partial patches were not recorded. Because the interest is to observe direct disclosure, (5) no quantitative records were made for those cases in which discoverers contacted third-parties who in turn contacted vendors. For cases reporting

multiple vulnerabilities that may affect multiple vendors, (6) all five life cycle variables refer to the latest dates provided. Finally, (7) all dates were manually coded from the *self-reported* values given by the particular discoverers.

The five variables capture the basic life cycle information related to direct disclosure. The corresponding dates are usually provided by discoverers in a forthright listings. Such listings made the manual quantification relatively easy. Although each unique discoverer uses a custom semi-structured format, the following cut excerpt (from EDB-18220) illustrates a typical albeit minimalist format for real-world vulnerability life cycle bookkeeping:

```
### Vendor logs:
# 10/10/2011 - Bug found
# 10/11/2011 - Vendor contacted
# 10/11/2011 - Vendor replied and requested POC
# 10/11/2011 - POC sent to vendor
# 10/31/2011 - Vendor said the POC will be researched
# 10/27/2011 - Submitted to CERT
# 11/09/2011 - CyberLink updated the product
# 11/09/2011 - POC still works on the latest version
# 12/09/2011 - No response from vendor, POC release.
```

Table 1: Interpretation of the Life Cycle Metrics

x	Meanings	Range $(-\infty, 0)$	Range $[0, \infty)$
a	The overall length of direct disclosure, including potential grace periods.	Publication before contacting a vendor; immediate-style disclosure.	Responsible disclosure; the larger the value, the longer the grace period.
b	How long a discoverer kept a zero-day vulnerability as a secret before contacting a vendor?	Negative values should be logically impossible; a contact mandates a discovery.	Large values indicate questionable practices; a secret was kept long with unknown motives.
c	An indicator for communication tardiness; how long was the communication handshaking?	Negative values are logically implausible; vendors cannot respond without being queried.	Large values indicate communication problems; reluctant vendors take long to respond.
d	An indicator for overall security risks; how long it took for vendors to release patches?	Public or full disclosure before patches; a violation of responsible disclosure ideals.	Small values are desirable; because POCs are released, vendors should patch rapidly.

The operationalization of the observed life cycle variables follows a familiar arithmetic path (Ruohonen et al., 2017; Sen and Heim, 2016). That is, the quantitative analysis operates with the following four subtractions:

$$x \in \begin{cases} a = Publication - Disclosure, \\ b = Disclosure - Discovery, \\ c = Response - Disclosure, \\ d = Publication - Patching. \end{cases}$$

Each resulting metric is restricted to be finite, although this is not necessarily true in practice. For instance, the operationalization for c is mathematically undefined in case a reluctant vendor never responds to an initial contact. Consequently, all metrics are used only in subsets that have quantitative data for both variables in the subtractions. For cases with no missing values, the optimum is $a = b = c = d = 0$, meaning that the whole process from discovery to patching was completed during the same day.

Table 2: Independent Metrics

Variable	Description / operationalization
CATEGORY^n	The four meta-data categories provided in EDB: DoS (denial-of-service), Web, Local, and Remote.
PLATFORM^n	The six most frequent platforms in the sample (Linux, Windows, Hardware, Multiple, PHP, and ASP) together with a re-coded catch-all group (Others) for all other platforms provided in EDB.
APPLICATION^d	Value one if the vulnerable application is available for download from EDB and a value zero otherwise.
VERIFIED^d	Value one if the EDB community has verified the POC exploit and a value zero otherwise.
SCREENSHOT^d	Value one if a screenshot is available in EDB for demonstrating the functionality; zero otherwise.
TOPDEV^d	Given EDB's author records, a value one in case the author is among the top-10 most productive POC authors in the sample; a value zero otherwise.
SEVERITY^i	The mean of NVD's CVSS base scores for all CVEs referenced in EDB for a given POC exploit; a value zero for exploits with no CVE references.
DECADE^d	Given EDB's publication dates, a value one for exploits published in the 2010s; a value zero otherwise.
MONTH^n	Given EDB's publication dates, the calendar time month in which the POC exploit was published.
OUTLIER^d	A value one for life cycle values exceeding the 99.5:th percentile and zero for all other values.

Scale: ^{d} for dichotomous, ^{n} for nominal, and ^{i} for interval.

As summarized in Table 1, the four metrics each answer to specific life cycle questions. Descriptive statistics are used as the primary quantitative mean for seeking answers to these questions. In addition, *exploratory* regression analysis is used for examining whether the life cycle metrics vary systematically across a few structural factors. The adjective *structural* is used to emphasize that none of the independent metrics considered have been explicitly designed to measure disclosure and related processes (Ruohonen et al., 2017). Rather, the set of independent metrics enumerated in Table 2 refers to the meta-data categories provided in EDB and, in the case of severity proxied through the Common Vulnerability Scoring System (CVSS), the National Vulnerability Database (NVD). As usual, identifiers for the Common Vulnerabilities and Exposures (CVEs) are used to link the information between the two databases.

3. Results

The following presentation of the empirical results follows the thematic qualitative approach. The dissemination proceeds from the most notable repeatedly occurring themes to a few important detours worth discussing. To improve readability, only those themes are explicitly referenced that are backed by three or more cases. The symbol S is used for the referencing; the corresponding EDB identifiers are listed in Appendix (from Table A.1 to Table A.5). Likewise, direct quotations are referenced with the symbol Q, and the EDB identifiers for these are listed in Appendix Table A.6. The brief quantitative analysis ends the presentation and ties some of the loose ends.

3.1. Ideal Cases

In many cases direct disclosure processes have followed a linear life cycle pattern: a vendor was contacted, the vendor delivered a response and subsequently made patches available, after which information was publicly disclosed. The overall turnaround cycle was sometimes rapid.

The clear majority of the direct disclosure cases observed relied on electronic mail for communication. Only a few outlying cases indicated more personal communication, such as phone calls, video conferences, and face-to-face meetings. Even though there were also a few cases involving Internet relay chat (IRC), vendor-specific web applications, and social media (particularly Twitter but in a few cases also Facebook) communication (S_1), these related mainly to discoverers' attempts to find suitable initial contact persons (S_2). Email communication was also fluent in the ideal cases. Discoverers were sometimes even surprised about how fast vendors responded (S_3). When communication was good, good things often followed.

The information exchanged was not necessarily limited to the initial notifications and vendors' short responses. For instance, a discoverer prepared a security advisory, which was further delivered to the corresponding vendor for evaluation, while the vendor in turn delivered further technical information to the discoverer for research purposes. Some actively participating vendors sent status updates to keep discoverers updated (S_4). One responsive vendor even notified a discoverer that they had accidentally disclosed unnecessary information to the public during patch development, and consequently requested a more coordinated approach. Another vendor went further by granting access to an internal software development system in order for the discoverer to participate in patch development. Some vendors proactively delivered the patched versions for the discovers to test (S_5). Thus, in the ideal cases it was typical that discoverers and vendors worked together for developing patches. In some cases direct disclosure did work almost as described in a guideline book.

In particularly illuminating, almost stylized examples, communication started in good spirit with an exchange of pretty-good-privacy (PGP) keys, proceeded to discussion about reproducibility, included a few test rounds with confirmations, and finally ended to vendors' requests about proper acknowledgements for the discoverer in the vendors' own security advisories (S_6). Some vendors also reviewed discoverers' prepared advisories for technical accuracy. When communication worked well, the discoverers were often explicit about future public disclosure and the exact dates on which sensitive information was to be made public (S_7). This explicit but slightly coercive strategy may have improved communication, which, however, was generally a visible bottleneck in the historical direct disclosure cases observed. Ideal cases were ideal cases.

3.2. General Communication Problems

Different communication problems were prevalent throughout the period examined. While many vendors preferred encrypted (PGP) communication (S_8), some vendors seemed to have difficulties in understanding why PGP was—and still is—preferable for confidential information exchanges. More generally, it was typical that a good initial start for a communication did not last long.

Often vendors' initial replies did not imply subsequent replies (S_9). In a couple of illuminating cases discoverers and vendors exchanged various messages via electronic mail, but at some point the vendors no longer responded, which forced the discoverers to make phone calls in order to salvage the situations. Another good example would be the few cases that involved forwarding of the emails to the associated development teams—who never replied. Likewise, many vendors requested further clarifications, which were delivered, but which tended to end the communication from the vendors' side (S_{10}). As these cases exemplify, communication often ended either abruptly or gradually during the direct disclosure processes.

There were also many other manifestations of communication problems. For instance, communication was supposedly difficult with a vendor when one individual answered to initial queries, but another individual stepped

in to provide updates. Another individual hid the confidential information in a development system for security reasons, which made an unaware individual to complain about improper procedures to a discoverer. After a lengthy email exchange, another discoverer nailed down an important issue by recommending "*that developers actually in charge of these issues are copied in the e-mail loop, or that access to internal issue-tracking tools be given to them to actively participate in the discussions and the patching process*" (Q_1). In other words, it was difficult to disclose information to apparently already somewhat dysfunctional organizations. This point likely applies also today.

3.3. Reluctant Vendors

Many vendors did not bother answering to queries. Many vendors refused to release patches, either explicitly or implicitly. Patches never arrived even after information had already been published. In general, it was thus common that there were neither responses nor patches (S_{11}). Nor did reminders help, regardless whether these were done via emails or phone calls (S_{12}). In many cases responses were missing even after multiple different attempts that involved video demonstrations, grace periods measured in months, numerous emails, contact attempts via online forms, and public or private Twitter messages.

Also the initial communication handshaking often required multiple contact attempts, but even established communication channels seemed to require constant reminders. An email after an email was required for knowing whether a reluctant vendor was even receiving the messages (S_{13}). When no responses were received for mostly unknown reasons, there was often no option but to proceed with public disclosure. "*Advisory released, unable to contact the vendor*" (Q_2), as one discoverer summarized the issue. The rationales for public disclosure included increasing the incentives for patching, sharing information with other discoverers, and informing the public. Not once did the discoverers agree to comply with vendors' requests for removing POCs and other details. In one case, for instance, a vendor first requested a contact via phone, which was declined, and then wanted that the exploit code should not be released with the discoverer's advisory, which was again declined because "*it gives the users a tool to assess the risks they are running and the effectiveness of possible countermeasures and workarounds*" (Q_3). These general ideals of public information and open data can be generalized to all discoverers present in the sample. After all, the information would not have otherwise ended in EDB.

Often, vendors patched vulnerabilities disclosed to them silently with no additional communication or coordination (S_{14}). Often, moreover, vendors released patches that did not actually correct or otherwise properly address the vulnerabilities reported (S_{15}). Unfortunately, the sample does not contain sufficient information about the reasons for such silent and inadequate patching. Perhaps vendors may have had difficulties to fathom the information communicated, but this explanation does not seem that likely. Because POCs were often delivered to vendors—if sometimes only upon request (S_{16}), concrete software testing should have been possible, which generally signals that technical characteristics unlikely explain the reluctance. Although not always having had time, many discoverers were generally willing and even keen for explaining technical details, including remediation instructions for vendors and users alike. Some discovers even offered to help in software testing and patch development (S_{17}). However, it was common that vendors neither responded nor utilized such instructions and other forms of voluntary help. Thus, to rephrase, vendors' plain incompetency offers a possible but not a plausible explanation for their reluctance.

When vendors were contacted and communication worked, the sample reveals a few noteworthy explanations for vendors' refusal to provide patches. First, in many cases vendors responded that no patches were provided because the products affected had already reached their *end-of-life* (EOL) software life cycle states (S_{18}). A product's looming EOL state was sometimes also used as a rationale for public disclosure, and exploits were sometimes also written for old software that is no longer supported (S_{19}). These observations indicate that software life cycles were an important element characterizing direct disclosure and the reluctance of vendors.

Second, some issues were supposedly too trivial for vendors and their users; an advice such as "*do a dirty patch*" (Q_4) seemed appropriate for some vulnerabilities. Vendors were also sometimes rather carefree with low-impact vulnerabilities, giving an early "*go ahead to publicly disclose*" (Q_5). In some other cases the software was so poorly engineered that direct disclosure itself was unlikely to make much of a difference. Some software products "*contain all sorts of obfuscated junk all the time*" (Q_6), as one discoverer bluntly but veraciously stated.

Third, some vendors disagreed about whether some issues reported to them were in fact vulnerabilities (S_{20}). Even though vulnerability disclosure institutions may have acted as arbiters in some of these disagreements, it was more common that vendors flat out refused to acknowledge the security implications. To give one example: upon reporting multiple security issues, a vendor's spokesperson noted that one of the issues was a "*feature*", while "*I do not consider this a vulnerability*" was a response to another issue (Q_7). Another vendor said similarly that the vulnerabilities reported "*were deliberate design decisions*" wanted by customers (Q_8). Regardless whether these examples are credible responses to security issues, it can be concluded that also disagreements shed some light on the problem of reluctant vendors.

Last but not least, scarce resources and tight schedules explained the reluctance in some cases; a "*vendor confirmed the vulnerabilities, but has no time to fix them*" (Q_9). Some vendors were too "*tightly scheduled on other priorities*" to properly fix vulnerabilities disclosed to them (Q_{10}). Other vendors were "*not going to rush out a release to fix*" less significant vulnerabilities (Q_{11}).

These four partial explanations notwithstanding, it remains largely a mystery why some truly reluctant vendors neither replied to inquiries nor patched their products.

3.4. Responsible Disclosure

Many discoverers had their own norms for grace periods. The range varied from short 7 day periods through 15, 30, and 40–45 days to longer 90 day grace periods (S_{22}). When asked, however, discoverers usually accepted delay requests (S_{23}). For instance, one vendor requested a "*90 day restraint on vulnerability release to give clients time to patch*" (Q_{12}). Another vendor requested more time so that a patch could be delivered alongside a major update. These examples are not outliers; the grace periods used during direct disclosure were often flexible and negotiable.

Many of the ideal cases show no visible life cycle problems regarding the time lines that both parties deemed as appropriate. When communication worked, the two parties were usually able to also negotiate each other's schedules (S_{24}). More generally, coordinated disclosure and remediation was often possible (S_{25}). These negotiations also dealt with vendors' specific policies, such as the release of patches on specific days. When these ideal cases are excluded, however, the sample reveals also many cases indicating that reluctant vendors seldom answered to proposals for responsible disclosure. Therefore, these qualitative observations do not mean that different life cycle issues would not have been present.

For instance, a minority of discoverers preferred rather narrow waiting times for responses to initial queries. "*Public disclosure and simultaneously initial vendor contact*" (Q_{13}) was a process preferred by some discoverers. These cases exemplify the rationale behind the concept of immediate (public) disclosure. In less extreme cases, less than a week was provided for a vendor, which was considered as sufficient for releasing the POC code developed (S_{21}). Given the slowness of email communication—and the general slowness of communication in large organizations, such grace periods seem rather harsh. These short periods are not necessarily unreasonable when communication works smoothly, which may explain the preference of short grace periods. If things have worked previously well with other vendors, even a small delay may seem too long when dealing with a more tardy vendor. In fact, bad experiences with responsible disclosure had turned some discoverers to prefer immediate public disclosure with some reluctant vendors and their problematic products. To conclude: also vulnerability life cycle characteristics have caused some problems in direct disclosure.

3.5. Third Parties

Different third-parties were often involved also in direct disclosure processes. Sometimes vendors delivered their responses via third-party security companies. For instance, in one case a vendor had contacted a global security company, which subsequently stated that the issue reported was not in fact a vulnerability. These outsourcing patterns caused also some problems. In another case, a vendor noted that the disclosed vulnerability belonged to an Internet service provider, which, unsurprisingly, did not respond to further queries. According to the qualitative material, it was relatively common that multiple vendors were affected, or that one or more third-party vendors were responsible for vulnerabilities that affected another vendor's products. The web application domain stands out in this regard. Open source software development is another visible case. Like today, coordination was often required between the so-called upstream (developers) and downstream (distributors) open source communities (S_{26}). Also discoverers themselves contacted different third-parties.

Many vulnerabilities are nowadays coordinated through bug bounties and their online platforms. However, the sample reveals that bug bounties and related arrangements were commonly used long before these gained mainstream traction (S_{27}). There are also a few interesting cases in the sample about different incentives and problems related to bug bounties. For instance, some discoverers pursued direct disclosure only after third-party disclosure companies had refused to handle (or pay for) the vulnerabilities.

Somewhat analogous observations apply with respect to vulnerability institutions and governmental agencies: many discoverers contacted such bodies even though they still relied on direct disclosure for the communication with vendors. In particular, many discoverers contacted computer emergency response teams (CERTs), which subsequently coordinated the initial communication with the associated vendors, helping to complete the communication handshaking, among other things (S_{28}). However, many of these cases did not imply that a CERT would have been the actual coordinator. Nevertheless, response teams indeed were one way for addressing the problem of reluctance, although even these teams did not always receive adequate replies from reluctant vendors (S_{29}). Ironically, also common vulnerability institutions sometimes exhibited typical forms of reluctance; in some cases they never bothered answering to queries, failed at keeping promises, and so forth (S_{30}). Multiple vendors, bug bounties, and disclosure institutions were not the only third-parties involved in the direct disclosure processes, however.

For gaining publicity, mailing lists, security media outlets, blogs, and data warehouses, including EDB itself (Hafiz and Fang, 2016), were used for publication, advertisement, and vulnerability tracking. These cases included also requests for CVE identifiers during the direct disclosure processes (S_{31}). Another aspect relates to vulnerability discoveries made in conferences and different hacking gatherings (S_{32}). A further important point is that some discoverers operated under contracts. In some cases these contracts required that a plan for disclosure was first coordinated with the customer contracted, after which the actual, vulnerable, vendor was contacted (S_{33}). All in all, these observations suggest that the hybrid disclosure models are not as clear-cut as often presented in the literature.

3.6. Maliciousness

The qualitative sample contains a few cases of full disclosure with no grace periods, meaning that a previously unknown zero-day vulnerability was disclosed to the public Internet before the associated vendor, or anyone else, had time to react. Although opinions vary and borderline cases have been common, it can be argued that maliciousness, as such, does not manifest itself by disclosing a few lines of code required for a cross-site scripting vulnerability. Likewise, when a discoverer rallied for a particular disclosure ideology, there was arguably nothing malicious in stating that "*full disclosure rocks!*" (Q_{14}). In contrast, some outlying cases have exhibited rather deliberate maliciousness, which seems to have been targeted either toward security companies, other discoverers, some particular vendors and software projects, or different underground groups. These few outlying cases are also accompanied with profane language, which is absent from most of the cases sampled.

But it is often difficult to say who is malicious and who is not. There are also many reckless software vendors who explicitly include questionable techniques into their software products, but refuse to take any real responsibility from their actions. An illuminating case starts with a few acknowledgements about the original disclosure in a security conference, proceeding to remind the Internet that a company for network switches had included a backdoor for one of its products, and the issue had already been unresolved for a year. Sadly, this case is not a sole representative in the sample about vendors' backdoors (S_{34}). It is often difficult to say what is ethical and what is unethical, but most would place these cases of public disclosure toward the ethical end of the continuum.

3.7. Quantitative Observations

Quantitative analysis is a good way for ending the empirical exposition. Before continuing to the results, it is important to emphasize that the quantitative data cannot be used for answering to questions regarding the truly reluctant vendors who never responded to any queries. In other words, all four life cycle metrics (see Section 2.3) reflect more the ideal cases than the particularly problematic ones trapped by the qualitative analysis. While keeping this important point in mind, the manually quantified life cycle metrics are illustrated in Fig. 2. By using Table 1 as an interpretation guide, the results can be summarized as follows, moving from *a* to *d* consecutively.

- The clear majority of the direct disclosure cases observed have used grace periods with varying lengths. Full disclosure before notifying vendors was relatively rare, as can be concluded by the small amount of negative values in the upper-left plot. When the negative values are removed, it can be concluded that the average grace period lengths varied from about 34 (median) days to 70 days (mean). This range supports the qualitative observations.

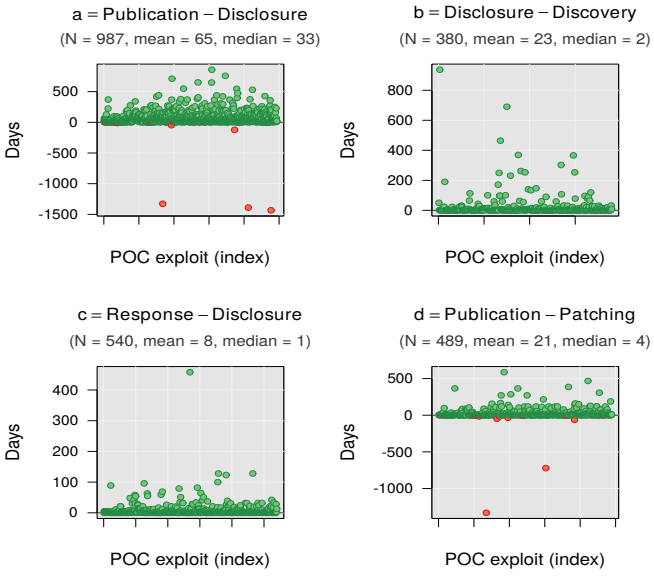

Figure 2: Life Cycle Metrics

- Most discoverers were quick to contact vendors after discovering vulnerabilities; on average, the time lag was about two (median) to 23 days (mean). That said, there were also some alarming lags between the initial discoveries and the subsequent disclosures to vendors. Although a couple of extreme outliers are present in the sample (which may have also resulted from discoverers' typing errors), there were also a few cases of discoverers holding to their secrets even up to six months. Proof-of-concept exploit development takes time like any other type of development, of course, yet these delays still seem a little odd when considering the motto of public disclosure.

- Provided that vendors responded to inquiries, they were relatively fast. On average, about a week was required for receiving the first replies. Even through the truly reluctant vendors are not present in the quantitative sample, tardiness still manifests itself through the few vendors whose initial replies have taken a month or more. In one extreme case a vendor took even over a year to respond to a query.

- The sample contains only a few cases whereby discoverers published POCs before vendors had their patches, advisories, or commits ready. This observation is again reflected in the small amount of negative values in the lower-right plot in Fig. 2. Furthermore, in the subset with sufficient data (N = 489), about 15% of the cases saw a publication during the same day as vendors released their patches or other remediation solutions. This detail again supports the earlier qualitative observations; schedules were often coordinated or even synchronized.

A few further observations can be drawn by regress-

ing the four life cycle metrics against the structural factors in Table 2. Because the amount of negative values is small for a and d (and, as expected, absent for b and c), it seems reasonable to focus on the subsets of non-negative values. This choice leads to a further decision on the regression methodology to use. Initial modeling indicates decent enough OLS estimates when the non-negative values are further passed through a $\ln(x + 1)$ transformation, as in previous vulnerability disclosure research (Arora et al., 2010; Ruohonen et al., 2018). While this transformation makes the residuals from the regression models roughly normal (see Fig. 3), the residuals still show patterns of non-constant variances. For this reason, the results reported in Table 3 were computed with a heteroskedasticity-consistent covariance matrix estimator.

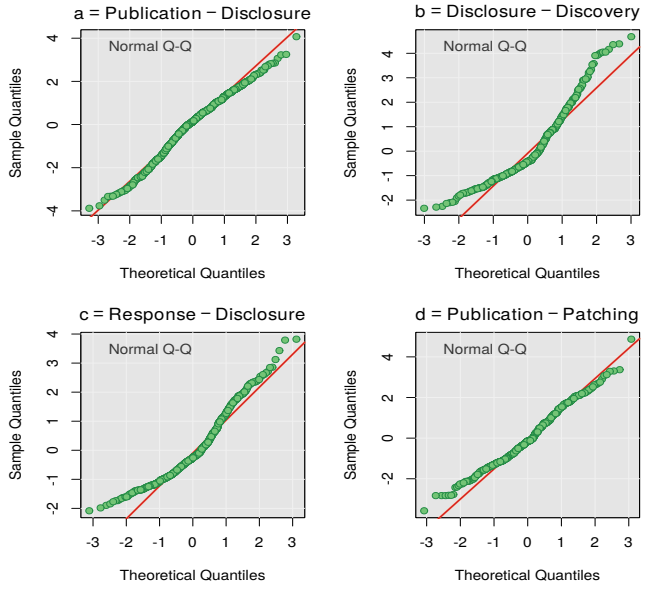

Figure 3: Residual Normality

The estimates indicate that there exists some systematic variation in the four timelines with respect to the EDB's meta-data category and platform. When compared to web vulnerabilities and POC exploits thereto, denial-of-service and remote vulnerabilities seem to have decreased vendors' reply times. Remote vulnerabilities tended to also increase the time vendors took to patch. These observations seem logical in the sense that web vulnerabilities are often less severe and less complex. For these reasons, discoverers may have preferred short grace periods for web vulnerabilities and vendors may have fixed these more quickly.

However, the regression results do not allow to explicitly infer about the relationship between the timelines and the severity of the vulnerabilities reported. Neither the average CVSS base scores nor the squared scores attain statistically significant coefficients, but this is related to the operationalization of the SEVERITY metric (see Table 2). Most of the long timelines specifically refer to vulnerabilities with an average base CVSS score of zero, meaning that these interesting cases do not have CVE references required for NVD-based CVSS information. Due to this data limitation, it would be equally unwarranted to use the individual CVSS metrics in the present context.

The three metrics related to EDB's quality assurance (APPLICATION, VERIFIED, and SCREENSHOT) are not easy to interpret, but it can be noted that verified cases seem to have decreased the values of all four metrics. Also the metric TOPDEV attains a negative coefficient for d. Taken together, these observations hint that vendors may have patched more quickly when they dealt with seasoned discoverers and robust POCs.

The estimates for DECADE indicate that slightly longer grace periods were used in the 2010s compared to the 2000s and 1990s. Although the details are omitted from Table 3 for the sake of brevity, also some of the monthly effects are statistically significant with relatively high coefficient magnitudes. Finally, the adjusted coefficients of determination indicate only modest overall statistical performance. These are familiar observations from earlier studies (Arora et al., 2010; Ruohonen et al., 2018). It is probable that direct disclosure timelines have contained systematic statistical variations, but the meta-data information available from existing databases seems to capture only a small portion of such variations. The quantitative results are also exposed to some validity concerns soon discussed.

4. Discussion

The following last rites summarize the key results, discuss limitations, and point out further research potential.

4.1. Key Results

The foremost key result can be summarized as follows:

- *The problem of reluctant vendors was still present in the 2000s and early 2010s. Many many vendors were reluctant, slow, and generally unwilling to participate in direct disclosure of software vulnerabilities.*

This answer supports industry surveys according to which some vendors are still hesitant to participate or even hostile toward discoverers (Ring, 2015; Uchil, 2016). Though, every cloud has a silver lining. When reporting vulnerabilities to a unfamiliar vendor, there are five basic possibilities: the vendor "could (1) respond gratefully and patch the vulnerability as soon as possible, (2) ignore it, (3) deny it, or (4) report to the police" (Kranenbarg et al., 2018, p. 3), or (5) threaten with legal action. The close to five thousand cases observed do not fortunately indicate any hints about the fourth and fifth response types. Despite of the many problems reported, also direct disclosure processes have improved in this regard from the 1990s. Nevertheless, the prevalence of the second type has serious consequences for both research and practice. In terms of the latter, the overall price to pay from the reluctance

Table 3: Regression Results

			Dependent Metric			
Metric	Reference	Variable	a	b	c	d
	–	Constant	3.918***	1.386*	0.944***	2.522***
CATEGORY	Web	DoS	0.140	-0.240	-0.518**	-0.394
		Local	-0.166	-0.341	-0.154	0.379
		Remote	0.164	-0.105	-0.424*	0.596*
PLATFORM	Others	ASP	-0.684	0.607	-0.128	-0.847
		Hardware	0.042	0.200	0.537**	0.184
		Linux	-0.252	0.510	-0.347	-0.044
		Multiple	-0.204	1.045*	0.102	0.072
		PHP	-0.933***	0.185	0.018	-0.618*
		Windows	-0.327	0.938*	0.193	-0.436
APPLICATION	Unavailable	Available	-0.559***	-0.446*	-0.108	-0.134
VERIFIED	Unverified	Verified	-0.358***	-0.125	-0.205	-0.805***
SCREENSHOT	Unavailable	Available	-0.471	-0.900**	0.067	-0.211
TOPDEV	Other authors	Top-10 authors	-0.096	-0.494	0.216	-0.859***
SEVERITY	–	Mean CVSS	-0.028	0.118	-0.092	0.066
	–	(Mean CVSS)^2	0.002	-0.018	0.014	-0.012
DECADE	1990s and 2000s	2010s	0.573***	0.517*	0.149	0.070
OUTLIER	No	Yes	3.316***	5.663***	4.049***	3.503***
MONTH	Eleven dummy-variables included for monthly effects but not shown for brevity.					
N			976	380	540	474
Adjusted R^2			0.205	0.147	0.117	0.246
Breusch-Pagan (p-value)			0.013	0.572	0.044	0.022

Notes: non-negative values with a $\ln(x+1)$ transformation for all $x \in \{a, b, c, d\}$; OLS regression with standard errors based on White's (1980) covariance matrix estimator; the last row probes the homoskedasticity of the residuals (H_0) according to the test of Breusch and Pagan (1979); *** for $p < 0.001$, ** for $p < 0.01$, and * for $p < 0.05$.

largely falls into the hands of consumers and users who continue to use software products known to be vulnerable.

In terms of research, the literature on vulnerability disclosure has traditionally placed a great deal of emphasis on the dates on which patches and exploits were released to the public. These dates have frequently been used to define different exposure periods (Arbaugh et al., 2000; Bilge and Dumitras, 2012; Nappa et al., 2015). The quantity d in Table 1 is a typical example in this regard. However, observations made in recent research allow to question how reliable and valid such exposure periods are in reality.

While reliability is threatened by data quality issues, some of the validity problems are accompanied with more interesting contextual observations. For instance, recent research has revealed time lags in CVE allocation, non-synchronized release of patches by regional subsidiaries, as well as supplementary patches for patching unintended omissions in earlier patches (Li and Paxson, 2017; Nakajima et al., 2019; Park et al., 2017; Ruohonen et al., 2018). The qualitative results presented augment these observations: vendors have frequently released also invalid patches that do not actually fix the vulnerabilities disclosed to them. More strikingly, the problem of reluctant vendors continued to further imply that patches never arrived for many products that were known to have been vulnerable. If there are no patches for many vulnerabilities and incorrect patches for some, how valid can the different exposure periods be? As was noted in Subsection 2.3, time differences cannot even be used in this non-patching context because the other quantity in a subtraction is undefined.

The reluctance of vendors should not be exaggerated, however. Sometimes direct disclosure worked well or even smoothly. The quantitative results indicate that direct disclosure was relatively fast when the initial communication obstacles had been solved: the mean turnaround times were 23 and 8 days for discoverers to contact vendors after their discoveries and vendors to respond to the discoverers' initial queries, respectively. The median values were even lower. These quantitative observations can be accompanied with the qualitative observations about secure communication of confidential vulnerability information. While keeping in mind that the quantitative results do not apply to vendors who neither respond to queries nor patch their products, the few but noteworthy ideal cases signify another key result. It can be stated as follows:

- *When direct disclosure worked particularly well, it was relatively fast and tended to exhibit distinct forms of coordinated software engineering, including collabo-*

rative patch development, sharing of technical information, multi-party software testing, synchronized release of security advisories, and related activities.

In other words, the problem of reluctant vendors was present, but the problem cannot be generalized to all vendors. Partially due to the nature of the sample and the ethnographic focus adopted (see Subsection 2.2), only a few answers were present in the data for the more fundamental question of *why* the problem was still present. The notable explanations culminate in four facets: (a) release engineering, software life cycles, and EOL states; (b) poor software quality and low-impact vulnerabilities; (c) disagreements between discoverers and vendors; (d) and vendors' limited time and scarce resources for patching. The first facet has also been a familiar theme in media (Corfield, 2019), and the last facet has been visible in industry surveys (Cimpanu, 2018). The importance of software life cycles is also noteworthy because the amount of vulnerabilities discovered has been observed to slow down as software products age (Ruohonen et al., 2015). Although not visible in the sample, it should be emphasized that the reluctance was presumably also explained by business and related factors. This presumption is backed by observations that security issues can have a negative impact upon quality perceptions (Licorish et al., 2015), and even affect firms' stock prices (Spanos and Angelis, 2016). The following key result points out more practical problems:

- *Direct disclosure was highly vulnerable to different communication delays, handshaking difficulties, breakages, and related communication problems, likely partially due to the reliance on electronic mail.*

Thus, communication is still a visible bottleneck. It is almost like emails are caught by spam filters or just otherwise end up in void. The medium used for communication likely contributes to the problems. For instance, feedback delays typically increase the ambiguity in email communication (Byron, 2008), and postponing replies to emails is a typical way to intentionally delay decision-making (McKeown and Zhang, 2015; Shirren and Phillips, 2011). Furthermore, the results provide only weak support for the existing observations about increasing use of social media for disseminating vulnerability and other security information (Bullough et al., 2017; Le et al., 2019; Sabottke et al., 2015; Syed et al., 2018). According to the qualitative observations, social media is mainly used for finding contact persons. The observation is sensible in the sense that the disclosure of complex technical information can hardly be done in a tweet. Therefore, it is also probable that the fundamental characteristics of email communication continue to hinder also the contemporary direct disclosure practices.

Responsible disclosure has been a common norm also in direct disclosure. When communication worked, the parties involved also managed to often coordinate each other's responses. This said, the quantitative results show some signs of vendors' tardiness on one hand, and discovers' short grace periods on the other. The average grace period used was about 30–70 days. Although some current practitioners disagree (Ring, 2015), this range may be too short particularly for more complex vulnerabilities (McQueen et al., 2011). Furthermore, it is again important to stress that this average length reflects the ideal cases rather than the problematic ones. By relying on both the quantitative results and the qualitative results, the following key result provides a more balanced viewpoint on the vulnerability life cycle characteristics:

- *When communication worked, direct disclosure was relatively fast, but when communication was problematic, discoverers often preferred public disclosure over delayed responsible disclosure with reluctant vendors, suggesting that direct disclosure practices tended to operate with relatively short grace periods.*

This key result can be balanced with a corollary:

- *Although the ideology of public disclosure—and, to a lesser extent, the ideology of full disclosure—is universally shared in the empirical sample analyzed, the direct disclosure processes observed only very seldom exhibited malicious motives.*

While disclosure ideologies still played a strong role in the direct disclosure practices, many of the discoverers observed were employed security professionals specialized in vulnerability discovery. There was still, of course, a blend of professionals and hobbyists, but particularly the newer cases point toward a professionalized field. Given that also criminology research has recently shown interest in vulnerability disclosure (Kranenbarg et al., 2018), a couple of reservations can be nevertheless mentioned. First, nothing can be concluded about potential use of the information possessed prior to the dates of disclosure. The quantitative results indicate that some discoverers have held some vulnerabilities suspiciously long as private secrets. Malicious motivates may be present in some cases, but human mistakes and lack of commonly agreed disclosure procedures may explain other cases. "*Sorry, I was not sure how to handle this and forgot about it for a long time*" (Q_{15}), as one discoverer noted. Second, it is impossible to evaluate whether and how the criminal undergrounds are connected to direct disclosure and the publication of POCs. As was elaborated, these and other difficult questions do not apply only to discoverers. There are also suspicious vendors.

To some extent, the observed reluctance issues also signify the business model that the specialized disclosure companies launched in the early 2000s. There is still a market demand for a commercial hybrid disclosure model, which may well be a better and more efficient practice also in terms of actual security attacks and exploitation. The same point applies to bug bounties and public sector involvement via CERTs and related institutions. Reflecting the wickedness of the problem, however, the qualitative results also indicate that even prominent national CERTs do not always have practical means to counter the reluctance.

The role of an intermediate actor can be further reflected against the communication patterns during direct disclosure processes. For businesses, business-to-business or business-to-public-sector relationships are presumably easier and more familiar than communication with supposedly often unknown actors who possess confidential security information. Trust is necessary for any business. Consequently, even the ideas of business-to-hacker communication may be scary for many vendors. Although the analysis did not try to explicitly identify the discoverers, there are both qualitative and quantitative signs that seasoned discoverers with commercial affiliations were able to handle disclosure better than individual hackers. Liability and juridical reasons may play their roles, but some of these cases provide good targets for communication practices. A cavalier but frank style may work also for hackers.

The hybrid disclosure models are relevant also from a research perspective, as summarized in the final key result:

- *Different third-parties, including, but not limited to, vulnerability disclosure institutions, other vendors, CERTs, and disclosure companies, were often present also in direct disclosure processes, suggesting that the theoretical hybrid and direct disclosure types may intervene in practice.*

In other words, direct disclosure often contained different hybrid behavioral patterns, which imply that the theoretical demarcations are not as sharp as often presumed in the disclosure literature (cf. Ransbotham et al., 2012, Fig. 1). These observations also open a window of opportunity for further research. Before considering future research directions, some limitations must be acknowledged.

4.2. Threats to Validity

A few potential threats to validity should be discussed. Although the validity of qualitative results cannot be evaluated in quantitative terms (Williams and Morrow, 2009), the concepts of external, construct, and internal validity can be still used by loosening the definitions for these. In what follows, the results are thus evaluated in terms of the soundness of the measures, concepts, and abstractions (construct validity), the generalizability or transferability of the empirical results (external validity), and the consistency of the general empirical reasoning (internal validity).

4.2.1. Construct Validity

A notable construct validity threat relates to the timestamp quantities used to compute the four life cycle metrics. In addition to the subjective element involved in enforcing the coding criteria (see Section 2.3), the use of self-reported values is problematic. However, it should be emphasized that also other studies have relied on self-reported bookkeeping material (Ablon and Bogart, 2017), and particularly that similar manual coding is typically done by vulnerability and exploit database maintainers.

There are also more theoretical concerns. For instance, the date of discovery does not rule out the possibility that someone else would not have found (and perhaps even disclosed) the same vulnerability earlier (McQueen et al., 2011), which would entail the question about *vulnerability rediscovery* (Ozment, 2005). Another point worth noting is that the dates characterizing exploit development were not quantified; therefore, the relationship between POCs and direct disclosure remains implicit in the quantitative analysis because it is impossible to deduce whether a discoverer had a POC ready upon contacting a vendor. Despite of these issues, the fine-grained definitions (Garcia et al., 2014; Hahn and Govindarasu, 2012; Li and Paxson, 2017) used in the present work are useful in the sense that many studies tend to strictly equate the date of disclosure to the date on which information was published in a vulnerability database (Massacci and Nguyen, 2014; Nappa et al., 2015; Syed et al., 2018). Given that CVE identifiers are often requested while the disclosure processes are still active (S_{31}), it is clear that the events of disclosure have occurred already much earlier. Furthermore, vulnerability severity is one important element that is being assessed and negotiated during direct disclosure (Householder et al., 2017). By implication, also CVSS metrics are theoretically problematic for the quantitative analysis: if the information has not yet been available, it cannot have influenced the disclosure processes. Analogous points apply to most variables listed in Table 2. When a discoverer first contacts a vendor, there is seldom information that has already been stored into databases. These potential construct validity issues apply only on the quantitative side, however.

4.2.2. External Validity

There are four ways to consider generalizability threats. The first is to consider whether the sample from EDB generalizes to all cases archived in the database. However, it is not easy to evaluate such generalizability because sampling from the database cannot be done randomly; only a minority of the cases archived contain information about the direct disclosure phenomenon. That said, it is worth to remark that the sample size (about 13% of all archived cases at the time of the data collection) is still larger than what has been used (3.5%) in comparable settings (Holm and Afridi, 2015). The second way is to contemplate about EDB's generalizability toward all vulnerabilities publicly disclosed and archived. The database contains only a small subset of cases archived in NVD (Allodi and Massacci, 2013), and NVD contains only a subset of archival material in other databases (Ablon and Bogart, 2017). Nor does EDB cover all POC exploits (Sabottke et al., 2015). Thus, it seems reasonable to assume generalizability toward the cases archived in EDB but not toward other databases.

The third way is to consider whether the historical sample observed generalizes to the present day. It seems fair to assume this kind of longitudinal generalizability. As was remarked in Subsection 2.1, a socio-technical phenomenon is unlikely to change rapidly. By implication, particularly

the sample's many observations between 2010 and 2016 should reasonably generalize to 2019. Threats to longitudinal generalizability are also abated because the paper's scope was explicitly framed to direct disclosure without attempting to compare it to other types of vulnerability disclosure. While it is true that bug bounties have changed vulnerability disclosure practices and processes (Ruohonen and Allodi, 2018), direct disclosure is still widely practiced, and there is no particular reason to assume that the problems described would no longer exist.

The fourth and final way relates to generalizability on the qualitative side. In this regard, one important point is that EDB attracts quite a specific community (Hafiz and Fang, 2016). Although this point might be used for arguing that the qualitative results are too context-specific, it should be emphasized that EDB has also sought to systematically gather historical archival material from mailing lists and other sources. Because a thematic approach was used, generalizability of the qualitative results can be also understood to relate to a *saturation point* after which no important themes emerge in the qualitative analysis (Cruzes et al., 2015; Guest et al., 2006). Given these four ways, it is fair to conclude that external validity may be problematic, but the problem is more pressing on the quantitative side. The qualitative results are robust and saturated enough for supporting the primary conclusion: the problem of reluctant vendors was still present in the 2000s and early 2010s—and likely is still present today.

4.2.3. Internal Validity

A few points can be remarked about the internal validity problems affecting the whole domain of empirical vulnerability disclosure research. When disclosure occurs privately between two actors, it is possible that no traces are left for scholars to examine. Some vulnerabilities never attain their own distinct digital object identifiers; some vulnerabilities never live through the publication life cycle state. Unfortunately, it is impossible to evaluate how many vulnerabilities are disclosed to vendors without informing the public. Another question relates to the amount of vulnerabilities that are never disclosed. This question is an important part of the problem of reluctant vendors. For instance, many discoverers still fear legal repercussions from reporting software vulnerabilities (Ring, 2015; Uchil, 2016; Whittaker, 2018). Regardless whether there are rational and justified reasons for such fears, these lead to suspect that a sizable amount of known vulnerabilities may remain undisclosed. An analogous point relates to vulnerabilities discovered and silently patched by vendors themselves. These fundamental issues must be acknowledged as limitations also in this paper.

Finally, it is necessary to briefly return to the nature of the dataset and the ethnographic tenet associated with it. As was noted in Subsection 2.2, the paper's focus was explicitly framed to the discoverers' point of view. Although this framing is sufficient for examining whether the problem of reluctant vendors was still present during the period observed, it may cause biases in further research regarding the why-question. Although the four simple explanations presented are unlikely threatened, it should be emphasized that discoverers are likely to exaggerate the problems and vendors to downplay these. It takes two to tango.

4.3. Further Research

Direct disclosure of software vulnerabilities still contained plenty of problems throughout the 2000s and early 2010s. Reflecting the philosophical balance between great harm and great good (Freeman, 2007), there are seldom bad things without glimpses of good things; sometimes direct disclosure worked even surprisingly well as a coordinated engineering activity. This activity was often outsourced to third-parties, including vulnerability disclosure institutions, disclosure companies, and crowd-sourced bug bounty programs. Although many scholars have rallied for the associated hybrid disclosure models (Arora et al., 2010; Cavusoglu et al., 2007; Mitra and Ransbotham, 2015), including bug bounties (Choi et al., 2010), none of the disclosure practices are likely to disappear in the near future.

This presumption is an important point for further research. For vulnerability disclosure research, (1) it is important to consider means for improvement by studying particular types rather than comparing rival disclosure types. The results presented also suggest that different disclosure patterns are not as clear as previously presented; there are many borderline cases that exhibit both direct and hybrid behavioral patterns. Therefore, also (2) further theoretical work is required. Bug bounties are a good topic in this regard. To give an intriguing anecdotal example, a discoverer associated with the Google's bug bounty program experienced considerable difficulties when pursuing direct disclosure of a high-profile vulnerability to a multinational company—which had outsourced its disclosure practices to another bug bounty (Silvanovich, 2018). This example is illuminating in the sense that essentially similar hybrid disclosure practices may cause problems due to multiple different disclosure institutions, and due to the fact that direct disclosure is sometimes preferable despite of the availability of these supporting institutions.

By increasing the existing knowledge about the under-researched direct disclosure practice, (3) the paper also taps for further research regarding trust and communication of confidential security information, including the importance of studying leaks from coordinated disclosure practices (cf. Sabottke et al., 2015). Given the persistence of the problem of reluctant vendors, (4) further research should also examine whether vulnerability disclosure practices are poorly understood in the industry. Even though there is an international standard for software vulnerability disclosure (ISO/IEC, 2014), it seems that many vendors are either unaware of the standard or repel complying with it. A related important theme that warrants further research is (5) the already enacted or the still emerging regulations and legislations for vulnerability disclosure.

These have been actively pursued in recent years particularly in the European Union (Ķinis, 2017; Pupillo et al., 2018; Silfversten et al., 2018). As was briefly noted in Subsection 3.5, vulnerability disclosure often involves also job contracts and other legal arrangements that make a comparative juridical analysis challenging. By implication, however, the already interdisciplinary nature of vulnerability disclosure research can be further strengthened; political scientists, criminologists, and legal scholars may bring important new insights to the research domain.

Finally, (6) further software engineering research is required for better understanding the aspects of multi-party coordination and synchronization of security engineering activities. Better understanding requires also further empirical vulnerability life cycle modeling with more nuanced and combined datasets. Because software life cycles are an important element also in direct disclosure, an interesting research direction would open by merging software and vulnerability life cycles into a unified framework.

References

Ablon, L. and Bogart, A. (2017). Zero Days, Thousands of Nights: The Life and Times of Zero-Day Vulnerabilities and Their Exploits. RAND Corporation, Santa Monica. Available online in September 2017: https://www.rand.org/content/dam/rand/pubs/research_reports/RR1700/RR1751/RAND_RR1751.pdf.

Algarni, A. M. and Malaiya, Y. K. (2014). Software Vulnerability Markets: Discoverers and Buyers. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 8(3):480–490.

Allodi, L. and Massacci, F. (2013). Poster: Analysis of Exploits in the Wild – Or: Do Cybersecurity Standards Make Sense? In *Proceedings of the IEEE Symposium on Security and Privacy (S&P 2013)*, pages 1–2, San Francisco. IEEE.

Almukaynizi, M., Nunes, E., Dharaiya, K., Senguttuvan, M., Shakarian, J., and Shakarian, P. (2018). Patch Before Exploited: An Approach to Identify Targeted Software Vulnerabilities. In Sikos, L. F., editor, *AI in Cybersecurity*, pages 81–113. Springer, Cham.

Arbaugh, W. A., Fithen, W. L., and McHugh, J. (2000). Window of Vulnerability: A Case Study Analysis. *Computer*, 32(12):52–59.

Arora, A., Forman, C., Nandkumar, A., and Telang, R. (2010). Competition and Patching of Security Vulnerabilities: An Empirical Analysis. *Information Economics and Policy*, 22(2):164–177.

Bilge, L. and Dumitras, T. (2012). Before We Knew It: An Empirical Study of Zero-Day Attacks in the Real World. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS 2012)*, pages 833–844, Raleigh. ACM.

Böhme, R. (2006). A Comparison of Market Approaches to Software Vulnerability Disclosure. In Müller, G., editor, *Proceedings of the International Conference on Emerging Trends in Information and Communication Security (ETRICS 2006), Lecture Notes in Computer Science (Volume 3995)*, pages 298–311, Freiburg. Springer.

Bozorgi, M., Saul, L. K., Savage, S., and Voelker, G. M. (2010). Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2010)*, pages 105–114, London. ACM.

Breusch, T. S. and Pagan, A. R. (1979). A Simple Test for Heteroskedasticity and Random Coefficient Variation. *Econometrica*, 47(5):1287–1294.

Bullough, B. L., Yanchenko, A. K., Smith, C. L., and Zipkin, J. R. (2017). Predicting Exploitation of Disclosed Software Vulnerabilities Using Open-Source Data. In *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics (IWSPA 2017)*, pages 45–53, Scottsdale. ACM.

Byron, K. (2008). Carrying Too Heavy a Load? The Communication and Miscommunication of Emotion by Email. *Academy of Management Review*, 33(2):309–327.

Carayon, P., Kianfar, S., Li, Y., Xie, A., and Bashar (2015). A Systematic Review of Mixed Methods Research on Human Factors and Ergonomics in Health Care. *Applied Ergonomics*, 51:291–321.

Cavusoglu, H., Cavusoglu, H., and Raghunathan, R. (2007). Efficiency of Vulnerability Disclosure Mechanisms to Disseminate Vulnerability Knowledge. *IEEE Transactions on Software Engineering*, 33(3):171–185.

Choi, J. P., Fershtman, C., and Gandal, N. (2010). Network Security: Vulnerabilities and Disclosure Policy. *The Journal of Industrial Economics*, 58(4):868–894.

Cimpanu, C. (2018). 26% of Companies Ignore Security Bugs Because They Don't Have the Time to Fix Them. Bleeping Computer LLC. Available online in May 2018: https://www.bleepingcomputer.com/news/security/26-percent-of-companies-ignore-security-bugs-because-they-don-t-have-the-time-to-fix-them/.

Ķinis, U. (2017). From Responsible Disclosure Policy (RDP) towards State Regulated Responsible Vulnerability Disclosure Procedure (hereinafter – RVDP): The Latvian Approach. *Computer Law & Security Review*, 34(3):508–522.

Corfield, G. (2019). Security Gone in 600 Seconds: Make-Me-Admin Hole Found in Lenovo Windows Laptop Crapware. Delete It Now: Solution Centre WONTFIX Amid EOL Date Shenanigans. The Register, available online in September 2019: https://www.theregister.co.uk/2019/08/23/lenovo_solution_centre_cve_2019_6177/.

Cruzes, D. S., Dybå, T., Runeson, P., and Höst, M. (2015). Case Studies Synthesis: A Thematic, Cross-Case, and Narrative Synthesis Worked Example. *Empirical Software Engineering*, 20(6):1634–1665.

EDB (2016). Offensive Security Exploit Database. Available online in November 2016: https://www.exploit-db.com/.

Finifter, M., Akhawe, D., and Wagner, D. (2013). An Empirical Study of Vulnerability Reward Programs. In *Proceedings of the 22nd USENIX Security Symposium*, pages 273–288, Washington. USENIX.

Freeman, E. H. (2007). Vulnerability Disclosure: The Strange Case of Bret McDanel. *Information Systems Security*, 16(2):127–131.

Garcia, M., Bessani, A., Gashi, I., Neves, N., and Obelheiro, R. (2014). Analysis of Operating System Diversity for Intrusion Tolerance. *Software: Practice and Experience*, 44(6):735–770.

Guest, G., Bunce, A., and Johnson, L. (2006). How Many Interviews Are Enough? An Experiment with Data Saturation and Variability. *Field Methods*, 18(1):59–82.

Hafiz, M. and Fang, M. (2016). Game of Detections: How Are Security Vulnerabilities Discovered in the Wild? *Empirical Software Engineering*, 21(5):1920–1959.

Hahn, A. and Govindarasu, M. (2012). Cyber Vulnerability Disclosure Policies for the Smart Grid. In *Proceedings of the IEEE Power and Energy Society General Meeting*, pages 1–5, San Diego. IEEE.

Hesse-Biber, S. (2010). Qualitative Approaches to Mixed Methods Practice. *Qualitative Inquiry*, 16(6):455–468.

Holm, H. and Afridi, K. K. (2015). An Expert-Based Investigation of the Common Vulnerability Scoring System. *Computers & Security*, 53:18–30.

Householder, A. D., Wassermann, G., Manion, A., and King, C. (2017). The CERT® Guide to Coordinated Vulnerability Disclosure. Special Report, CMU/SEI-2017-SR-022, CERT Division, Carnegie Mellon University. Available online in August 2017: https://resources.sei.cmu.edu/asset_files/SpecialReport/2017_003_001_503340.pdf.

ISO/IEC (2014). Information Technology – Security Techniques – Vulnerability Disclosure, ISO/IEC 29147:2014(E). The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). Available online in August 2017: http://standards.iso.org/ittf/PubliclyAvailableStandards/c045170_ISO_IEC_29147_2014.zip.

Kranenbarg, M. W., Holt, T. J., and van der Ham, J. (2018). Don't Shoot the Messenger! A Criminological and Computer Science Perspective on Coordinated Vulnerability Disclosure. *Crime Science*, 7(16):1–9.

Le, B. D., Wang, G., Nasim, M., and Babar, A. (2019). Gathering Cyber Threat Intelligence from Twitter Using Novelty Classification. In *Proceedings of the International Conference on Cyberworlds (CW 2019)*, Kyoto. IEEE. Available online in September 2019: https://arxiv.org/abs/1907.01755.

Li, F. and Paxson, V. (2017). A Large-Scale Empirical Study of Security Patches. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS 2017)*, pages 2201–2215, Dallas. ACM.

Licorish, S. A., MacDonell, S. G., and Clear, T. (2015). Analyzing Confidentiality and Privacy Concerns: Insights from Android Issue Logs. In *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering (EASE 2015)*, pages 18:1–18:10, Nanjing. ACM.

Massacci, F. and Nguyen, V. H. (2014). An Empirical Methodology to Evaluate Vulnerability Discovery Models. *IEEE Transactions on Software Engineering*, 40(12):1147–1162.

McKeown, J. and Zhang, Q. (2015). Socio-Pragmatic Influence on Opening Salutation and Closing Valediction of British Workplace Email. *Journal of Pragmatics*, 85:92–107.

McQueen, M., Wright, J. L., and Wellman, L. (2011). Are Vulnerability Disclosure Deadlines Justified? In *Proceedings of the Third International Workshop on Security Measurements and Metrics (Metrisec 2011)*, pages 96–101, Banff. IEEE.

Mitra, S. and Ransbotham, S. (2015). Information Disclosure and the Diffusion of Information Security Attacks. *Information Systems Research*, 26(3):565–584.

Nakajima, A., Watanabe, T., Shioji, E., Akiyama, M., and Woo, M. (2019). A Pilot Study on Consumer IoT Device Vulnerability Disclosure and Patch Release in Japan and the United States. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications (Asia CCS 2019)*, pages 485–492, Auckland. ACM.

Nappa, A., Johnson, R., Bilge, L., Caballero, J., and Dumitras, T. (2015). The Attack of the Clones: A Study of the Impact of Shared Code on Vulnerability Patching. In *Proceedings of the IEEE Symposium on Security and Privacy (IEEE S&P 2015)*, pages 692–708, San Jose. IEEE.

Ozment, A. (2005). The Likelihood of Vulnerability Rediscovery and the Social Utility of Vulnerability Hunting. In *Proceedings of the Workshop on Economics and Information Security (WEIS 2005)*, Cambridge. Available online in September 2019: https://pdfs.semanticscholar.org/1e83/3bab15a975dd71af5fcbbaed4a8df3a5be8a.pdf.

Ozment, A. (2007). Improving Vulnerability Discovery Models: Problems with Definitions and Assumptions. In *Proceedings of the 2007 ACM Workshop on Quality of Protection (QoP 2007)*, pages 6–11, Alexandria. ACM.

Park, J., Kim, M., and Bae, D.-H. (2017). An Empirical Study of Supplementary Patches in Open Source Projects. *Empirical Software Engineering*, 22(1):436–473.

Pupillo, L., Ferreira, A., and Varisco, G. (2018). Software Vulnerability Disclosure in Europe: Technology, Policies and Legal Challenges. Report of a CEPS Task Force, Centre for European Policy Studies (CEPS), available online in September 2019: https://www.ceps.eu/wp-content/uploads/2018/06/CEPS%20TFRonSVD%20with%20cover_0.pdf.

Ransbotham, S., Mitra, S., and Ramsey, J. (2012). Are Markets for Vulnerabilities Effective? *MIS Quarterly*, 36(1):43–64.

Ring, T. (2015). White Hats Versus Vendors: The Fight Goes On. *Computer Fraud & Security*, (10):12–17.

Ruohonen, J. and Allodi, L. (2018). A Bug Bounty Perspective on the Disclosure of Web Vulnerabilities. In *Proceedings of the 17th Annual Workshop on the Economics of Information Security (WEIS 2018)*, pages 1–14, Innsbruck. Available online in June 2019: https://weis2018.econinfosec.org/wp-content/uploads/sites/5/2018/05/WEIS_2018_paper_33.pdf.

Ruohonen, J., Hyrynsalmi, S., and Leppänen, V. (2015). The Sigmoidal Growth of Operating System Security Vulnerabilities: An Empirical Revisit. *Computers & Security*, 55:1–20.

Ruohonen, J., Hyrynsalmi, S., and Leppänen, V. (2017). Modeling the Delivery of Security Advisories and CVEs. *Computer Science and Information Systems*, 14(2):537–555.

Ruohonen, J., Rauti, S., Hyrynsalmi, S., and Leppänen, V. (2018). A Case Study on Software Vulnerability Coordination. *Information and Software Technology*, 103:239–257.

Sabottke, C., Suciu, O., and Dumitraş, T. (2015). Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting Real-World Exploits. In *Proceedings of the 24th USENIX Security Symposium*, pages 1041–1056, Washington. USENIX.

Sarker, S., Xiao, X., and Beaulieu, T. (2013). Qualitative Studies in Information Systems: A Critical Review and Some Guiding Principles. *MIS Quarterly*, 37(4):iii–xviii.

Schneier, B. (2007). Schneier: Full Disclosure of Security Vulnerabilities a 'Damned Good Idea'. Orginally published in CSO Online. Available online in August 2017: https://www.schneier.com/essays/archives/2007/01/schneier_full_disclo.html.

Sen, R. and Heim, G. R. (2016). Managing Enterprise Risks of Technological Systems: An Exploratory Empirical Analysis of Vulnerability Characteristics as Drivers of Exploit Publication. *Decision Sciences*, 47(6):1073–1102.

Sharp, H., Dittrich, Y., and de Souza, C. R. B. (2016). The Role of Ethnographic Studies in Empirical Software Engineering. *IEEE Transactions on Software Engineering*, 42(8):786–804.

Shirren, S. and Phillips, J. G. (2011). Decisional Style, Mood and Work Communication: Email Diaries. *Ergonomics*, 54(10):891–903.

Silfversten, E., Phillips, W., Paoli, G. P., and Ciobanu, C. (2018). Economics of Vulnerability Disclosure. European Union Agency for Network and Information Security (ENISA) & RAND Europe, available online in September 2019: https://www.enisa.europa.eu/publications/economics-of-vulnerability-disclosure.

Silvanovich, N. (2018). Adventures in Vulnerability Reporting. Project Zero: News and Updates from the Project Zero team at Google, available online in September 2019: https://googleprojectzero.blogspot.com/2018/08/adventures-in-vulnerability-reporting.html.

Silverman, D. (2006). *Interpreting Qualitative Data*. Sage, London, third edition.

Spanos, G. and Angelis, L. (2016). The Impact of Information Security Events to the Stock Market: A Systematic Literature Review. *Computers & Security*, 58:216–229.

Syed, R., Rahafrooz, M., and Keisler, J. M. (2018). What It Takes to Get Retweeted: An Analysis of Software Vulnerability Messages. *Computers in Human Behavior*, 80:207–215.

Uchil, J. (2016). Commerce Survey: Cyber Researchers Fear Legal Repercussions. The Hill. Available online in May 2018: http://thehill.com/policy/cybersecurity/310524-commerce-survey-cyber-researchers-fear-legal-repercussions.

White, H. (1980). A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity. *Econometrica*, 80(4):817–838.

Whittaker, Z. (2018). Lawsuits Threaten Infosec Research – Just When We Need It Most. ZDNet, CBS Interactive. Available online in February 2018: http://www.zdnet.com/article/chilling-effect-lawsuits-threaten-security-research-need-it-most/.

Williams, E. N. and Morrow, S. L. (2009). Achieving Trustworthiness in Qualitative Research: A Pan-Paradigmatic Perspective. *Psychotherapy Research*, 19(4–5):576–582.

A. Appendix

Table A.1: Identifier Sets for Main Qualitative Observations in Subsection 3.1

Set	Description	Identifiers for qualitative evidence
S_1	Social media communication	EDB-9562, EDB-14360, EDB-24975, EDB-30914, EDB-36930, EDB-36942, EDB-37708, EDB-40298
S_2	Finding contacts via social media	EDB-19793, EDB-37527, EDB-37532, EDB-39696, EDB-40160
S_3	Surprisingly fast replies	EDB-9562, EDB-15298, EDB-20677
S_4	Status updates for discoverers	EDB-14369, EDB-24158, EDB-33942
S_5	Forwarding patches for evaluation	EDB-24202, EDB-24475, EDB-25138, EDB-39571, EDB-39664
S_6	Smooth disclosure process	EDB-10176, EDB-14360, EDB-34086, EDB-37449, EDB-38912
S_7	Deadlines for public disclosure	EDB-6278, EDB-8269, EDB-8922, EDB-22399, EDB-35936, EDB-39664, EDB-40414

Table A.2: Identifier Sets for Main Qualitative Observations in Subsection 3.2

Set	Description	Identifiers for qualitative evidence
S_8	Encrypted (PGP) communication	EDB-14360, EDB-15293, EDB-16016, EDB-35594, EDB-36619, EDB-37114, EDB-38197, EDB-40020, EDB-40335, EDB-40414
S_9	No communication after first reply	EDB-8880, EDB-8974, EDB-17117, EDB-21992, EDB-24453, EDB-38577, EDB-39798, EDB-40160
S_{10}	No communication after clarifications	EDB-10084, EDB-17113, EDB-17114, EDB-18186, EDB-20268, EDB-33455, EDB-39235, EDB-39385, EDB-39386, EDB-39784

Table A.3: Identifier Sets for Main Qualitative Observations in Subsection 3.3

Set	Description	Identifiers for qualitative evidence
S_{11}	No patches, no responses	EDB-1565, EDB-6301, EDB-8391, EDB-17116, EDB-17766, EDB-18085, EDB-22006, EDB-37708, EDB-38055, EDB-38225, EDB-39487, EDB-39516, EDB-39542, EDB-39659, EDB-39883, EDB-40229
S_{12}	No responses after reminders	EDB-12512, EDB-17011, EDB-24743, EDB-27285, EDB-38054
S_{13}	Messages not reaching target	EDB-9680, EDB-12330, EDB-23565, EDB-27402, EDB-37527, EDB-37527, EDB-37532, EDB-38321, EDB-38323, EDB-39246, EDB-39441
S_{14}	Silent patching	EDB-8406, EDB-8581, EDB-20877, EDB-29467, EDB-37346, EDB-37623
S_{15}	Incorrect patching	EDB-8257, EDB-8957, EDB-10665, EDB-17276, EDB-18090, EDB-22216, EDB-23565, EDB-34245, EDB-34308, EDB-34519, EDB-34918, EDB-37626, EDB-39119
S_{16}	Delivery of POCs upon request	EDB-14505, EDB-32919, EDB-35182, EDB-35936, EDB-40156
S_{17}	Help offers for testing and patching	EDB-5657, EDB-8269, EDB-9110, EDB-40063
S_{18}	Lack of patches due to EOL states	EDB-8963, EDB-10484, EDB-27805, EDB-31617, EDB-33792, EDB-34112, EDB-38245, EDB-39572, EDB-40207, EDB-40208, EDB-40298
S_{19}	Disclosure due to EOL states	EDB-21012, EDB-31337, EDB-36860, EDB-37266, EDB-40690
S_{20}	Disagreements	EDB-8957, EDB-34245, EDB-34254, EDB-34408, EDB-35182, EDB-35594, EDB-38197, EDB-40171, EDB-40414
S_{21}	Short grace periods (≤ 7 days)	EDB-3600, EDB-8384, EDB-14344, EDB-18200, EDB-18201, EDB-19266, EDB-27805, EDB-28183, EDB-34399, EDB-40474

Table A.4: Identifier Sets for Main Qualitative Observations in Subsection 3.4

Set	Description	Identifiers for qualitative evidence
S_{22}	Varying grace periods (8–90 days)	EDB-34131, EDB-37708, EDB-38055, EDB-38225, EDB-39441, EDB-39536, EDB-39555, EDB-39556, EDB-40360, EDB-40669
S_{23}	Delay requests from vendors	EDB-9110, EDB-31985, EDB-36950, EDB-36951, EDB-36953, EDB-36949, EDB-38323, EDB-39415, EDB-40161, EDB-40396, EDB-40669
S_{24}	Negotiated schedules	EDB-15145, EDB-15146, EDB-28183, EDB-36953
S_{25}	General coordination	EDB-8241, EDB-9151, EDB-10513, EDB-14687, EDB-17529, EDB-17929, EDB-18582, EDB-18985, EDB-21267, EDB-24932, EDB-26527, EDB-27011, EDB-31173, EDB-34086, EDB-35915, EDB-40161, EDB-40230

Table A.5: Identifier Sets for Main Qualitative Observations in Subsections 3.5 and 3.6

Set	Description	Identifiers for qualitative evidence
S_{26}	Open source coordination	EDB-8581, EDB-17174, EDB-22406, EDB-35595
S_{27}	Bug bounties	EDB-15463, EDB-35472, EDB-36903, EDB-37058, EDB-37172, EDB-38351, EDB-40045
S_{28}	Involvement of CERTs	EDB-9514, EDB-10185, EDB-10211, EDB-10213, EDB-19408, EDB-20011, EDB-20357, EDB-20360, EDB-21546, EDB-21866, EDB-23362, EDB-24463, EDB-33520, EDB-35357, EDB-36949, EDB-36950, EDB-36951, EDB-36953, EDB-40171
S_{29}	Reluctance despite of CERTs	EDB-18779, EDB-20063, EDB-20364, EDB-30914, EDB-40030, EDB-40200
S_{30}	Coordination problems	EDB-8777, EDB-10184, EDB-10187, EDB-37531
S_{31}	CVE allocation	EDB-6218, EDB-30062, EDB-34112, EDB-35382, EDB-37531, EDB-39402
S_{32}	Hackathons	EDB-40137, EDB-40220, EDB-40410
S_{33}	Disclosure under contracts	EDB-5232, EDB-5233, EDB-18822, EDB-19290, EDB-33894, EDB-34062, EDB-35442, EDB-39850
S_{34}	Backdoors	EDB-14875, EDB-29673, EDB-37625, EDB-37626, EDB-40106

Table A.6: Identifiers for Direct Quotations

Quote	Identifier	Quote	Identifier
Q_1	EDB-10364	Q_2	EDB-10364
Q_3	EDB-25987; cf. also EDB-40669	Q_4	EDB-2151
Q_5	EDB-14606; see also EDB-18788	Q_6	EDB-8801
Q_7	EDB-40414	Q_8	EDB-27286
Q_9	EDB-27129, EDB-27130	Q_{10}	EDB-34263
Q_{11}	EDB-9887	Q_{12}	EDB-38118
Q_{13}	EDB-35077; cf. also EDB-10390	Q_{14}	EDB-8818; cf. also EDB-29875
Q_{15}	EDB-37198; cf. also EDB-20677		