Exploring the Clustering of Software Vulnerability Disclosure Notifications Across Software Vendors

Jukka Ruohonen
University of Turku, Finland
juanruo@utu.fi

Johannes Holvitie
University of Turku, Finland
jjholv@utu.fi

Sami Hyrynsalmi
University of Turku, Finland
sthyry@utu.fi

Ville Leppänen
University of Turku, Finland
ville.leppanen@utu.fi

***Abstract*—This exploratory empirical paper investigates annual time delays between vulnerability disclosure notifications and acknowledgments by means of network analysis. These delays are approached through a potential clustering effect of vulnerabilities across software vendors. The analysis is based on a projection from bipartite vendor-vulnerability structures to one-mode vendor-vendor networks, while the hypothesized clustering effect is approached with a conventional community detection algorithm. According to the results, (a) vulnerabilities cluster across vendors, (b) which also explains a portion of the time delays, although (c) the clustering is not stable annually. The computed network (d) clusters can be also interpreted by reflecting these against common software security attack surfaces. The results can be used to contemplate (e) practical means with which the efficiency of vulnerability disclosure could be improved.**

***Index Terms*—software vulnerability, vulnerability disclosure, bug collision, social network, bipartite network, US-CERT**

I. Introduction

This exploratory empirical paper seeks to answer the following three questions by means of applied network analysis.

RQ$_1$ *Do software vulnerability disclosure notifications sent by a disclosure institution cluster across vendors?*

RQ$_2$ *If there is clustering, how stable is the effect annually?*

RQ$_3$ *Can a potential clustering effect explain a portion of the variance in the time delays between third-party vulnerability notifications and the subsequent acknowledgments in a particular responsible vulnerability disclosure type?*

These research questions (RQs) contain a number of terms that should be clarified before continuing. First and foremost, a software *vulnerability* is understood as a software defect with security implications. Second, software *vendors* are abstract entities that develop software products. This deliberately vague definition underlines that, in the present context, a vendor may be a private company, a non-profit, a public sector organization, an open source community, or an individual. The empirical analysis is based on network projections to vendor-vendor dimension – to the common vulnerabilities that specific vendors have shared in the subset of vulnerabilities disclosed via a third-party disclosure institution. Here, third, the term *disclosure* is understood as a practice via which vulnerability discoverers make the vulnerabilities known to other actors.

Fourth, the term *network* analysis refers to the use of graph theoretical concepts and methods for studying complex real-world phenomena; social network analysis is the most notable scholarly branch, although the domain extends well beyond (social) networks based on human behavior.

It should be further emphasized that the word cluster is a rather ambiguous term in the network literature. In this paper, therefore, fifth, a word *cluster* is used loosely in the sense of a dictionary definition ("*a group or bunch of objects or things joined together*"), while *clustering* is understood "*to be, form, or grow in a cluster or clusters*". In network analysis, these dictionary definitions include the so-called clustering coefficient [1] and different *community detection* algorithms. Usually, the latter build on the general idea that networks can be often divided into densely connected groups, which are only sparsely connected to each other [2]. Since the clustering coefficient is not utilized in this paper, also the noted dictionary definitions refer implicitly to clustering in the sense of community detection. Finally, as in many clustering problems, the basic approach to community detection is either theoretical or empirical; either some known groups unrelated to empirical network data are of interest, or the goal is to locate the communities from relational network data algorithmically [3]. The latter algorithmic path is pursued in this paper, although a brief qualitative assessment is carried out for evaluating whether it is also possible to interpret the computed clusters.

Sixth, in this paper, the term *time delay* is defined according to two endpoints: (a) a notification sent by a *third-party disclosure institution* [4] to a software vendor about a potential software vulnerability in its software product; and (b) the potential response from the vendor to the institution. The empirical sample is based on the notifications and acknowledgments sent and received by the United States Computer Emergency Readiness Team (US-CERT), which is in a third-party role with respect to the software vendors, providing also a third-party vulnerability database. These time delays can be seen as coarse *process metrics* for measuring the *coordination efficiency* of a specific *institutional* process for disclosing vulnerabilities. This process has been known as a *hybrid* vulnerability disclosure type in order to separate the process from other means by which vulnerabilities are disclosed [5]. In other words, there are also companies that have specialized into conquering a third-party role in vulnerability disclosure [6], although it has been also common to disclose vulnerabilities directly either to the public or to the corresponding vendors.

Last but not least, the particular hybrid disclosure process led by US-CERT has often focused on particularly severe vulnerabilities [7]. Multiple vendors may be affected, or the processed vulnerabilities may disturb the whole Internet, for instance. Furthermore, because US-CERT is backed by the United States government, the highest possible level of institutional trust is arguably provided. The associated hybrid model also provides a well-understood policy for further provisioning the trust between vendors and discoverers [8]. For instance, a grace period of 45 days is provided for vendors to patch their products after the notifications sent by US-CERT. This safety period underlines the term *responsible disclosure*. Because information is eventually disclosed to the public sphere after the grace period, it is ensured that the affected vendors do not have incentives to avoid patching their products, which was relatively common historically [5]. It should be emphasized, however, that also the hybrid model is entirely voluntary; there are no juridical requirements, for instance, and there are still today software vendors that are reluctant for participating.

The practical relevance of the paper can be motivated with a couple of arguments. The first is related to the efficiency aspect: if factors affecting the observed time delays can be robustly known, it should be possible to optimize the institutional disclosure process. If there are clusters (RQ_1), and these can explain the time delays (RQ_3), there may be means to consider different notification bundles, for instance, or the information can be used to contemplate whether some particular groups of vendors would benefit from some form of special targeted coordination. Second, provided that there are clusters (RQ_1), and regardless of the answers to RQ_2 and RQ_3, the empirical clustering effects – or, rather, the lack of thereof – provide a grain of information about *natural software diversity* [9] associated with the usually severe vulnerabilities handled by US-CERT. That is to say, clustering can generally reveal information about potential common attack surfaces.

If Apple and FreeBSD are systematically in a same cluster, for instance, it would seem that there is common code that may require special attention. The two vendors might be then encouraged by US-CERT for jointly coordinating their patching efforts. If the two vendors are systematically in the same cluster, it might be also reasonable for criminals to allocate specific resources for studying FreeBSD in order to find vulnerabilities that might be used for attacking supposedly more prestigious Apple targets. This particular example is deliberately rather obvious because Apple and FreeBSD indeed share code. Therefore, it would be more interesting when, say, FreeBSD and a car industry company would be systematically located in the same cluster. To these exploratory ends, the paper proceeds by first introducing the materials and methods in Section II. Results are presented in Section III. Discussion and concluding remarks follow in the final Section IV.

II. Materials and Methods

The empirical approach is based on the concept of bipartite networks, although the actual empirical analysis proceeds by means of conventional, unipartite, network methods. Examples of real-world bipartite networks come in vast and different numbers. Bipartite networks have been used for modeling scientific collaboration networks in terms of papers and authors [10], networks of directors and corporate boards [11], networks of countries and international organizations [12], plant-animal networks [13], and networks between distinct protein types [14], to point out only a few examples from a few distinctively different fields. All these examples share one similarity: nodes (vertices) are defined in terms of two non-overlapping sets.[1] In addition to briefly elaborating this two-mode structure, the forthcoming discussion introduces the dataset, and shortly outlines the methodological toolbox.

A. Bipartite Networks

Bipartite networks contain links (edges) between two different node types. The classical social network example is a relation between "actors" and "events", both of which are entirely abstract concepts. For instance, actors may refer to individuals or groups of individuals, which are affiliated with some abstract events, possibly including other non-overlapping groups. Therefore, bipartite networks are sometimes referred to as mutualistic networks [13], affiliation networks [15], or two-mode networks [1]. When the additional terminology is omitted, the underlying two-mode structure implies an $m \times n$ incidence matrix, say $\mathbf{B}$, from which the corresponding network is constructed. In other words, there are two node dimensions, in contrast to the more conventional, square adjacency matrix, $\mathbf{A}$, which, in terms of the bipartite terminology, would store a one-mode network of actors in relation to other actors, or events in relation to other events.

The basic idea is illustrated in Fig. 1 with three actors and two events, E_1 and E_2. As can be seen, the conventional representation has actors on the m rows, while events are referenced on the n columns. It is also possible to consider weighted bipartite networks, although the much simpler binary variants are sufficient for the empirical purposes of this paper.

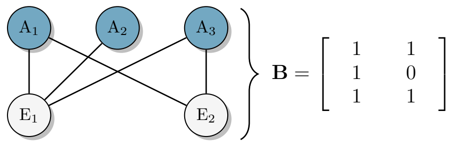


Fig. 1. A Small Bipartite Network

There are two common projections for transforming a bipartite network into a conventional, unipartite network [16]. When the incidence matrix $\mathbf{B}$ is post-multiplied with its transpose, $\mathbf{A}_a = \mathbf{B}\mathbf{B}'$, the non-diagonal elements in the square, $m \times m$, and symmetric $\mathbf{A}_a$ capture the number of event affiliations that are common to any two actors. To backtrack to Fig. 1, the lower-triangular part would contain elements $a_{21} = 1$, $a_{31} = 2$, and $a_{32} = 1$ because only A_1 and A_3 share an affiliation with both E_1 and E_2. In addition to this actor

[1] To clarify: the observed (undirected) bipartite graph can be represented by using three finite (unordered) sets, $G = (V_1, V_2, E)$, where an edge $(v, u) \in E$ is always placed from a vertex $v \in V_1$ to a vertex $u \in V_2$.

viewpoint, the duality perspective admits a product $\mathbf{A}_e = \mathbf{B}'\mathbf{B}$ within which a non-diagonal element $a_{ij} = a_{ji}$ displays the number of actors that are common to events i and j. For example, in the small example network the two non-diagonal elements a_{21} and a_{12} in the $n \times n$ matrix $\mathbf{A}_e$ would attain a value two. The diagonal elements in $\mathbf{A}_a$ and $\mathbf{A}_e$, respectively, capture the total number of event affiliations for each actor, and the overall number of actors affiliated with each event. It is also possible to combine the two projections into a single unified community detection framework [17]. The duality perspective is difficult to maintain with the projections, nevertheless.

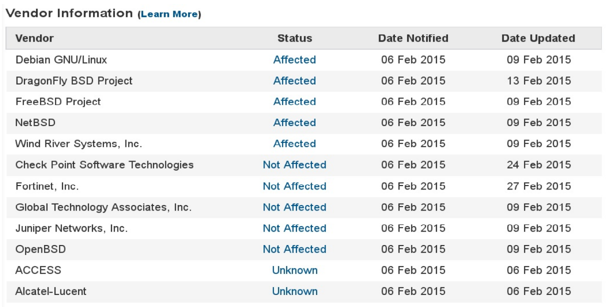
The two projections imply also potential biases [12]. For instance, the number of links is inflated [18], potentially relevant information is lost, and dense one-mode networks may result from sparse bipartite structures [1]. (The last point is easy to see already by considering the small example network for which both $\mathbf{A}_a$ and $\mathbf{A}_e$ represent maximally complete graphs.) Some of these issues can be balanced with different normalization procedures. One option is to normalize along the diagonal. In terms of $\mathbf{A}_a$, then, the non-diagonal elements $\widetilde{a}_{ij} = a_{ij} \,/\, a_{ii}$ represent the number of actors' common event-affiliations as a proportion of the total number of affiliations for each (row) actor [12]. The resulting matrix, denoted by $\widetilde{\mathbf{A}}_a$, is asymmetric. In Fig. 1, for instance, there would be an element $\widetilde{a}_{12} = 0.5$ because the two actors A_1 and A_2 are linked through E_1 but A_1 has two affiliations. That is, for the first actor the common affiliation with A_2 carries half of the weight of the affiliations with the third actor, $\widetilde{a}_{13} = 1$. This diagonal normalization procedure is also utilized in the subsequently described empirical analysis.

B. Data

There are numerous databases that cover public information about software vulnerabilities. These include private commercial sources, governmental databases, and different open source or volunteer collections. In general, the databases are only poorly compatible with each other, each source reflecting a slightly different empirical nuance to vulnerabilities – together with the database-specific limitations and inaccuracies. Many of the open data collections are based on two-mode relations, linking software vendors to software vulnerabilities.

In this paper, the empirical data is based on the vulnerability notes (VNs) database maintained by the Software Engineering Institute (SEI) at the Carnegie Mellon University for the U.S. Department of Defense [19]. While being operated by SEI, US-CERT, and other related U.S. institutions, the database is different from the more comprehensive National Vulnerability Database (NVD) maintained by the same parties, and from the Common Vulnerabilities and Exposures (CVEs) database provided by the MITRE corporation. In general, vulnerability notes are more informal than CVEs, covering information for remediation and short technical summaries, among other things. One thing makes the VN database preferable for the empirical purposes of this paper: it contains also partial historical records on the affected software vendors.

There were 3,175 entries during the data collection in the noon of March 13, 2015. For each of these entries, a table is provided for the vendors that are known, by US-CERT, to have been affected by a particular vulnerability, which may or may not have a more formal CVE record. For instance, the vulnerability note for a heap overflow in a regular expression library enumerates five vendors that are acknowledgedly affected [20]. The corresponding table is illustrated in the cut graphical excerpt shown in Fig. 2. As can be seen, there were, in addition, five vendors that were known to be unaffected (at the time of the query), and a number of vendors with unknown status, despite of the notifications sent to the vendors. Given these remarks, the empirical data was collected as follows.


Vendor Information (Learn More)

Vendor	Status	Date Notified	Date Updated
Debian GNU/Linux	Affected	06 Feb 2015	09 Feb 2015
DragonFly BSD Project	Affected	06 Feb 2015	13 Feb 2015
FreeBSD Project	Affected	06 Feb 2015	09 Feb 2015
NetBSD	Affected	06 Feb 2015	09 Feb 2015
Wind River Systems, Inc.	Affected	06 Feb 2015	09 Feb 2015
Check Point Software Technologies	Not Affected	06 Feb 2015	24 Feb 2015
Fortinet, Inc.	Not Affected	06 Feb 2015	27 Feb 2015
Global Technology Associates, Inc.	Not Affected	06 Feb 2015	09 Feb 2015
Juniper Networks, Inc.	Not Affected	06 Feb 2015	09 Feb 2015
OpenBSD	Not Affected	06 Feb 2015	09 Feb 2015
ACCESS	Unknown	06 Feb 2015	06 Feb 2015
Alcatel-Lucent	Unknown	06 Feb 2015	06 Feb 2015

Fig. 2. A Cut Example Record (from [19] for [20])

The selected longitudinal sample was first intentionally restricted to the past nine years. Then, for each year between 2006 and 2014, the date-updated field was used for annual sampling. For each vulnerability note in each year, only the (a) affected vendors were included for which both (b) the date-notified and the (c) date-updated fields were present. These three restrictions can be interpreted to shift the focus toward those particular software vendors that have actively participated in the coordination efforts led by US-CERT.

TABLE I
MERGED VENDOR ENTRIES

3com, Alcatel, Apache, Apple, Attachmate, Avaya, Cisco, Computer Associates, Debian, EMC, F5 Networks, FreeBSD, F-Secure, Fujitsu, Global Technology Associates, Huawei, IBM, Invensys, Linksys, Macrovision, Mandriva, Mozilla, NEC, Network Appliance, OmniGroup, Opera, QNX, Redback Networks, Red Hat, SCO, Sendmail, Shenzhen, Siemens, Sony, Sun Microsystems, SUSE, Symantec, TurboLinux, Verisign, Verity, WatchGuard, Yamaha

A given vendor entry is merged into a given listed string whenever a lowercase version of the string occurs in the entry. In addition, the two cases "Redhat" and "Trendmicro" were merged manually to the groups "Red Hat" and "Trend Micro".

In addition, some vendor-entries were corrected by merging similar entries (see Table I). While this improves slightly the reliability of the data, also some small ambiguities must be acknowledged; namely, the unified group for Apache contains both the hypertext transfer protocol server as well as one other project developed by the Apache Software Foundation. In general, however, the corrections seem reasonable, covering mostly cases in which a separate entry is given for a company and the company with its suffix (such as Inc. or Ltd.). For instance, Apple is listed as "Apple Inc.", "Apple Computer Inc.", "Apple Computer, Inc.", and "Apple". Although comparable issues are known to exist with respect to the uniqueness of vulnerability names [21], no further manipulations were carried out in the small programmed data collection procedure.

C. Manipulation

Two different types of incidence matrices were constructed:

$$\mathbf{B}_{t,V} \quad \text{and} \quad \mathbf{B}_{t,\Delta}, \tag{1}$$

where $t = (2006, \ldots, 2014)$. In both matrix sequences the included vendors are recorded on the m rows, while the n columns reference the vulnerability notes by their unique identification numbers. Thus, in terms of the bipartite network terminology, vendors are actors, while vulnerability notes are events. In the sequence $(\mathbf{B}_{2006,V}, \ldots, \mathbf{B}_{2014,V})$ the i:th row is a vector of binary numbers that score a value one in case the i:th vendor was affected by the j:th vulnerability (note). These nine matrices constitute the nine networks of interests (RQ$_1$), which are based on

$$\widetilde{\mathbf{A}}_{a,t,V} = f(\mathbf{A}_{a,t,V}) = f(\mathbf{B}_{t,V}\mathbf{B}'_{t,V}), \tag{2}$$

where $f(\cdot)$ denotes a function that carries out the discussed diagonal normalization procedure (see Section II-A). A non-diagonal element $\widetilde{a}_{ij}$ in a given annual $\widetilde{\mathbf{A}}_{a,t,V}$ represents the number of vulnerabilities that the i:th vendor has had in common with the j:th vendor in the t:th year, scaled by the total number of vulnerabilities that have affected the i:th vendor during the year t.

In the sequence $(\mathbf{B}_{2006,\Delta}, \ldots, \mathbf{B}_{2014,\Delta})$ a given element $b_{ij} \in \mathbf{B}_{t,\Delta}$ records the time difference (in days) that was required for the i:th vendor to respond to the notification from US-CERT about the j:th vulnerability (note). By reducing construct validity, these time delays might be interpreted as coarse proxies for the overall time it takes for vendors to construct patches. However, even without such stretching of interpretation, the delta, denoted generally by the symbol Δ, should be assumed to include all delays that were required for the communication between the vendor and US-CERT, and the latter to update its database, among other delays, empirical inaccuracies, and other reliability concerns. Since the vulnerability matrices are normalized with (2), but a similar procedure is not meaningful – in terms of the research questions – for the delay matrices, average annual amounts are used for observing the time delays:

$$\bar{b}_{i,t} = \frac{1}{n}\sum_{j=1}^{n} b_{i,j}, \quad i = 1, \ldots, m, \quad b_{i,j} \in \mathbf{B}_{t,\Delta}, \tag{3}$$

which implies that the averaging is done across all of the n products, irrespective whether these are linked to the i:th vendor. The row means in (3) are, however, relatively strongly influenced by outliers. A closer examination further reveals that these outlying vendors have not updated many products with relatively high delays, but have rather updated few products for which the notifications have been sent years ago. These are freely allowed to influence the observed delays on the grounds that the outliers imply long disclosure (and, presumably, patching) periods for vendors' software portfolios in general. Following the existing empirical research [22], in the empirical modeling, (3) was further passed through

$$g(\bar{b}_{i,t}) = \ln(\bar{b}_{i,t} + 1) \tag{4}$$

for obtaining the dependent variable for RQ$_3$. In general, the average delays have been quite reasonable, ranging roughly from one day to five days, which can be considered as fast.

D. Methods

The paper adopts the seminal community detection algorithm developed by M. E. J. Newman during the early 2000s (for an outline see [2]). The algorithm builds on the classical work from the 1970s during which the concept of betweenness was introduced [23]. This concept refers to the number of shortest paths that pass through an edge (or a vertex). The basic idea behind the divisive algorithm is to break a network into communities by iteratively removing links based on the highest (edge) betweenness scores. At each iteration, the betweenness scores are recomputed for accounting the removed links. The result is a dendogram, and the basic problem relates to suitable cuts in the dendogram for gaining optimal divisions. For this purpose, Newman introduced the modularity concept, which extends also to other algorithms, as well as to descriptive purposes with known community structures.

The modularity scalar is defined as the fraction of all links in a network that are connected to other nodes within the same communities minus the expected fractions with the same community structure but with random links between the nodes [2]. The closer the value to the maximum of unity, the stronger the strength of the community structure. Small, or even negative [24], values indicate that the algorithmic solution is no better than random. This early definition is rather imprecise regarding the second term in the subtraction, that is, the expected value of links placed at random, while keeping the node characteristics fixed. Therefore, the implementation used [25] utilizes a later, explicit definition [24]. Also many other alternatives and normalizations [26] have been used, partially owning to the fact that modularity maximization is known to misestimate the number of communities in some real-world networks [24], [27]. Nevertheless, the modularity measure is widely used in applied work, and, hence, well suited for exploratory empirical purposes. The same applies to the divisive edge-betweenness community detection algorithm.

Given the detected k edge-betweenness communities for the $i = 1, \ldots, m$ vendors during the t:th year, the following simple regression model is estimated:

$$g(\bar{b}_i) = \sum_{j=1}^{k} \zeta_j C_{i,j} + \varepsilon_i, \tag{5}$$

where the index t is omitted for emphasizing that nine separate models are estimated, $\bar{b}_i$ and $g(\cdot)$ are defined in (3) and (4), ε_i is the residual term, and $C_{i,j}$ is zero unless the i:th vendor belongs to the j:th community in which case $C_{i,j} = 1$.

The regression coefficients $\zeta_1, \ldots, \zeta_k$ capture the mean of the averaged annual delays. The basic interest, therefore, is to observe (H_0) whether $\zeta_1 = \zeta_2 = \cdots = \zeta_k$, meaning that the detected communities do not discriminate the time delays. The formula (5) refers to a classical one-way analysis of variance (ANOVA) model. The two fundamental statistical assumptions are: (a) the errors are independent and normally distributed, while (b) the within-community variances are approximately comparable. Both are generally important assumptions.

III. Results

The empirical exploration is carried out in three steps.

A. Clustering

The basic numerical information is shown in Table II both for the (a) complete networks and for (b) subnetworks (subsamples) from which those vendors have been removed that have had no shared vulnerabilities with other vendors. As could be expected, the edge-betweenness clustering algorithm places the isolate nodes into their own communities, which is evident already in the large number of computed communities, denoted by k. The number of communities varies considerably also in the subnetworks, $k \in [4, 23]$. These numbers do not match well the modularity scores, however. For instance, the correlation (Pearson) between k and the modularity scores are 0.43 and 0.17 in the whole networks and the subsamples, respectively. The equivalent correlations between m and modularity scores are 0.14 and -0.48, in the same order of listing. The latter negative sign is worth noting: in the subnetworks, an inverse relation seems to be present between the number of nodes and the modularity scores, whereas the contrary has been observed in some [26] biological networks.

TABLE II
Clustering (edge-betweenness communities)

	(a) Complete networks				(b) Subnetworks			
Year	m	e	k	Q	m	e	k	Q
2006	70	678	33	0.16	53	678	16	0.16
2007	111	384	71	0.63	54	384	14	0.63
2008	97	1158	52	0.18	68	1158	23	0.18
2009	68	622	20	0.45	60	622	12	0.45
2010	88	2442	27	0.04	65	2442	4	0.04
2011	75	350	42	0.33	39	350	6	0.33
2012	105	568	64	0.47	52	568	11	0.47
2013	99	542	66	0.46	51	542	18	0.46
2014	126	1466	68	0.16	68	1466	10	0.16
Mean	93	912	49	0.32	57	912	13	0.32
Std. deviation	20	680	19	0.20	10	680	6	0.20

The symbols refer to the number of nodes (m), edges (e), and computed communities (k), followed by the modularity scores (Q). These four scalar quantities are presented for the whole networks (see Section II-C) as well as for smaller subnetworks from which nodes without links have been removed, respectively.

In general, the modularity scores seem to be decent enough for many, but not all, annual subsample networks. For adjusting the interpretation, it is worth noting that a range from 0.3 to 0.7 has been observed to be typical [2], [3]. Five out of nine scores fall to this range in the annual subnetworks, but no

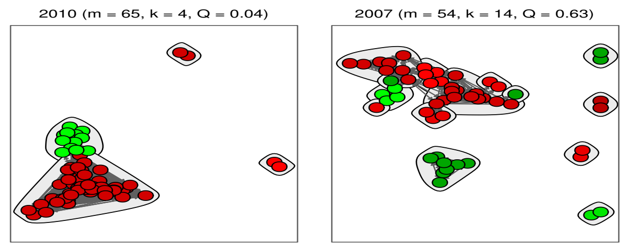

2010 (m = 65, k = 4, Q = 0.04)
2007 (m = 54, k = 14, Q = 0.63)

Fig. 3. Clusters in Two Annual Subnetworks

uniform annual trend can be observed. The subnetworks with the lowest and highest modularity scores are further visualized in Fig. 3, respectively. The solutions are distinctively different graphically. The annual sample from 2010, in particular, shows a large cluster of vendors that are all connected to each other. This is reflected in the very modest modularity score, although a subjective graphical interpretation is inclined to view also the left-hand side plot as a rather good-looking solution.

In conclusion, it is clear that the vendor-vendor projections do yield sound network clustering solutions (RQ$_1$). However, there is substantial annual variation. Thus, a negative answer must be given for the research question RQ$_2$. Instability rather than stability seem to characterize the clustering tendency.

B. Associations

The basic results from the regression models are shown in Table III. These can be summarized with four observations. First and second: although normality is problematic, the variance of the delays within the communities tend to be equal; only two subsample models fail at this check. Therefore, the basic requirements of ANOVA are mostly satisfied. Third, the actual F-tests show that only two subsamples fail to reject the assumption that the coefficients would be equal between the computed communities. In other words, the computed clustering structures can discriminate between the annual average time delays. Fourth, the adjusted R^2-values are decent, keeping in mind the limited information delivered via (5) alone.

TABLE III
Regression Models (ANOVA)

	(a) Complete networks				(b) Subnetworks			
Year	R^2	(1)	(2)	(3)	R^2	(1)	(2)	(3)
2006	0.68	✓	✓		0.62	✓	✓	
2007	0.52	✓	✓		0.47	✓	✓	
2008	0.76	✓	✓		0.76	✓	✓	✓
2009	0.67	✓	✓		0.66	✓	✓	
2010	0.88	✓	✓		0.82	✓	✓	
2011	0.58	✓	✓		0.67	✓		✓
2012	0.25		✓		0.46	✓	✓	
2013	0.41	✓	✓		0.61	✓	✓	
2014	0.16		✓		0.30	✓	✓	

The tabulated information contains: an adjusted R^2; (1) a F-test for H_0 that $\zeta_1 = \cdots = \zeta_k$ in (5); (2) a test (Shapiro-Wilk) for H_0 that ϵ_i is normally distributed; and a (3) test (Levene) that the variances are homogeneous across the k communities (H_0). For the last three, the symbol ✓ denotes $p < 0.05$. Analogous to Table II, the results are reported separately for the whole networks and the noted subnetworks within which nodes without links have been removed.

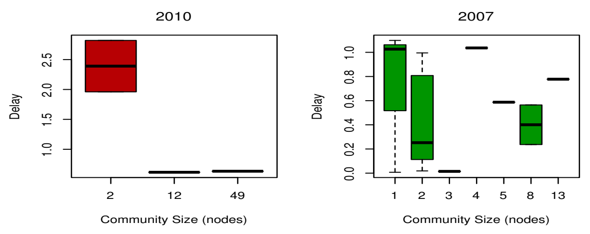


Fig. 4. Within-Community Delays by Community Size

It is illustrative for briefly returning to the left-hand side plot in Fig. 3 because this structure also attains the highest adjusted R^2-value. The four coefficients for the four communities are $\widehat{\zeta}_1 \simeq 0.62$, $\widehat{\zeta}_2 \simeq 1.96$, $\widehat{\zeta}_3 \simeq 2.82$, and $\widehat{\zeta}_4 \simeq 0.63$, leading to suspect that $\widehat{\zeta}_3$ might be the big cluster of 49 nodes. This is not the case, however. In fact, the approximated log-transformed average delay mean is highest in a community with only two nodes in Fig. 3. This observation is further illustrated in Fig. 4 by using the 2010 and 2007 subsamples, respectively, and by representing the estimated coefficients on the y-axes and the unique community sizes on the x-axis. The latter are defined in terms of the number of within-community nodes. Although the logarithm transformation smoothens the delays, the plots hint that the community structures may account also for the outlying vendors and their outlying products. All in all, the answer for RQ_3 is positive: a portion of the variance in disclosure delays is explained by the network clustering effect.

C. Qualitative Assessments

The answer to RQ_1 is positive: the projected vendor-vendor pairs do cluster. A brief qualitative assessment is required for evaluating whether the clusters can be also interpreted.

A visual illustration is shown in Fig. 5 by using the annual 2009 cluster as an example. As can be seen, Lenovo shared a vulnerability with IBM, which seems relatively logical against the business history of the two companies. Another example would be the cluster of SAP, Siemens, and Unigraphics Solutions. Also this cluster likely reflects the historical market situation; the company behind Unigraphics Solutions, UGS, was acquired by Siemens in 2007, while Siemens and SAP have long shared different business arrangements. Likewise, VanDyke Software has specialized into secure shell (SSH) software solutions, and, hence, it is no surprise that the company clusters with PuTTY, the most common SSH client for Microsoft Windows platforms.

A further qualitative illustration is shown in Table IV for the case of NetBSD, the oldest open source operating system project in existence. The results are relatively clear. A few observations seem warranted. First, NetBSD expectedly often clusters with FreeBSD and OpenBSD because all three operating system projects share a large amount of code. Second, and again expectedly, NetBSD often clusters with other open source projects that are either distributed by the project, or with which NetBSD has had close collaboration relationships. The particular examples would be OpenSSH and the Internet Software (Systems) Consortium, which is associated with the development of the Bind (DNS) and DHCP servers, both of which have also been traditionally distributed within NetBSD. Last, and most interestingly, this open source operating system project clusters particularly with a few notable networking companies and embedded software vendors. Also this observation is expected because the no-strings-attached, BSD-licensed, open source code from NetBSD has been used by many companies over the years. There are a few perhaps more noteworthy further details. For instance, Microsoft has donated code to NetBSD, and, either by implication or by chance, both vendors are located in the same annual 2012 cluster. Analogously, NetBSD is the only BSD project that offers support for the Xen (Dom0) hypervisor, and indeed the two vendors were affected by the same vulnerabilities in 2012. In terms of attack surfaces, the annual 2008 cluster offers a good final illustration; a number of SSH implementations and vendors were affected by the same vulnerability.

In summary, these qualitative observations suggest that it is indeed possible to interpret the clusters – it is possible to draw analytical conclusions from the exploratory results. These

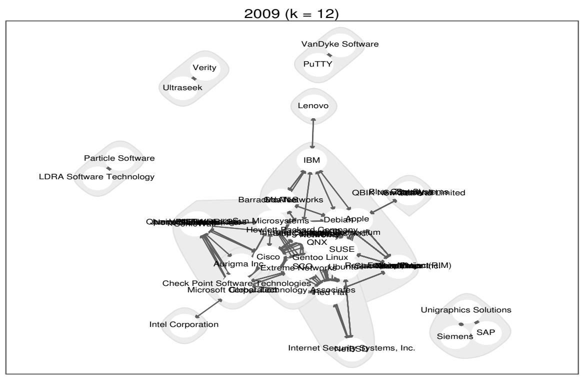


Fig. 5. Vendors in the Annual 2009 Subnetwork Cluster

TABLE IV
THE NETBSD CLUSTERS

Year	Vendors
2006	*Slackware Linux Inc., Fedora Project, TurboLinux, NetBSD, OpenBSD, Stonesoft, OpenSSL, Trustix Secure Linux, Internet Software Consortium*
2007	*NetBSD, Internet Software Consortium*
2008	*Icon Labs, NetBSD, Wind River Systems, Inc., OSSH, TTSSH, FiSSH, Redback Networks, SSH Communications Security Corp, OpenSSH, Bitvise*
2009	*Gentoo Linux, QNX, SCO, NetBSD, Internet Security Systems, Inc.*
2010	–
2011	*FreeBSD, Force10 Networks, Inc., NetBSD*
2012	*Microsoft Corporation, Xen, NetBSD, Intel Corporation, FreeBSD*
2013	–
2014	*Red Hat, FreeBSD, Apple, Symantec, IBM, F5 Networks, Google, VMware, Ubuntu, SUSE, Oracle Corporation, NEC, Global Technology Associates, Hewlett-Packard Company, Fedora Project, Debian, McAfee, CA Technologies, NIKSUN, Slackware Linux Inc., Novell, Inc., Palo Alto Networks, Juniper Networks, Inc., Gentoo Linux, Check Point Software Technologies, Fortinet, Inc., Extreme Networks, D-Link Systems, Inc., Cisco, Blue Coat Systems, Avaya, Barracuda Networks, Unisys, Wind River Systems, Inc., OpenBSD, NetBSD, Mandriva, Hitachi, Intel Corporation*

The table shows all vendors that were located in the same annual clusters as NetBSD.

conclusions can also bring additional value to security decision making, or to the evaluation of different strategies, policies, and risks. In theory, a more detailed analysis could be used as a heuristic for evaluating common attack surfaces in preparation of targeted attacks – or the strategic solutions required for building defenses against such attacks, for instance.

IV. Discussion

This empirical paper examined three questions. First and foremost: the vulnerability notifications sent by US-CERT cluster across software vendors (RQ_1), and it is also possible to interpret the vendor-vendor clusters. Second and third: although the clustering effects have not been stable over the years (RQ_2), these effects can still explain some of the variation in the time delays between the vulnerability notifications sent by US-CERT and the corresponding acknowledgements from the affected vendors (RQ_3). The remainder of this paper enumerates a few limitations, notes a few of related works, and finally discusses a few practical aspects related to the results.

A. Limitations

The limitations can be addressed by considering the concepts of reliability and validity, which in the context can be discussed in terms of the robustness of data (reliability of measures) and the soundness of analysis (what is measured is valid and validly measured). The reliability of the dataset is low, and arguably lower than the validity of the paper.

As was demonstrated, sensible vendor-vulnerability analysis requires reclassifications, although such corrections were done only for the observed vendor entries in the VN database. Comparable data collection problems are also well-recognized in the literature. A little more interesting are the techniques used for balancing the issues in the public databases. Manual inspections, reclassifications, and educated but subjective exclusion of cases are all common in the field [21], [22], [28]. Needless to say, the general situation is suboptimal for further advances in the empirical modeling of vulnerability data.

This paper used standard open data from a public database. Already since this database is actively maintained and widely used also by practitioners, the basic vendor-delay information is contextually valid. One particular (internal) validity issue can be noted about the methods: the paper was based on a simple projection from the conventional two-mode vendor-vulnerability pair [22], [28] to a one-mode vendor-vendor dimension. Although this projection was suitable for the research questions examined, it is unclear how different duality-preserving methods compare to the one-mode analysis. As much of the recent research agrees that the duality should be preserved [15], [17], [18], the one-mode analysis can be also noted as a limitation. This remark can be accompanied with a more practical concern about the averaging of the time delays across the software products of all observed vendors.

B. Related Work

The paper is located in the intersection of two scholarly branches. As the paper elaborates, bipartite networks have been extensively studied from applied computational perspectives [1], [12], [27]. Moreover, computer science incorporates a large and overlapping literature that focuses on the more formal graph theoretical aspects related to bipartite networks. The second scholarly branch is located in the empirical software vulnerability research domain generally, and the smaller subdomain of vulnerability disclosure research in particular.

Although network analysis has not been previously used in the domain of software vulnerability (disclosure) research, a few of the related contributions are worth elevating from the literature. In particular, the hybrid US-CERT-led disclosure process has been studied with both predictive [22] and descriptive [7] statistical methods, as well as with fully theoretical models [5]. Although even rather exotic questions have been examined, such as whether the hybrid vulnerability disclosure model is associated with firms' stock prices [29], a more traditional question has been the empirical determination of the time delays that occur during disclosure process and the associated stage of software patching [22], [28]. Given the observed vendor-vendor clustering effect, the aspect of competition between vendors [22] is particularly noteworthy. The perspective adopted in this paper builds on a rather contrary interpretation that emphasizes coordination rather than competition. In other words, as the presumable explanation for the clustering relates to the shared code bases and other collaborative software industry dynamics, coordination and collaboration are arguably more important for the efficiency of the particular hybrid disclosure type examined in the paper.

Finally, to summarize, in software engineering the subdomain of vulnerability disclosure research traces to the concept of vulnerability life cycle [30], [31], which is also related to the more encompassing concept of software life cycle [32]. Vulnerabilities are introduced, discovered, disclosed, patched, and exploited. All these concepts are representatives of different theoretical vulnerability life cycle stages. The life cycle thinking also connects the research domain to traditional security risks. If exploits are already available in the wild Internet but patches have not been delivered, vulnerability disclosure must be efficient, regardless of the particular means by which the disclosure is arranged between discoverers, vendors, the public sphere, and potential third-parties. Such risks also signify the practical relevance of applied empirical vulnerability disclosure research; the potential for analytical, theoretical, and empirical optimizations that may improve the efficiency of this particular type of software engineering work.

C. Concluding Remarks

Many software vendors are continuously affected by same vulnerabilities. For further research, an interesting question would be to examine whether more thorough clustering analysis could be adopted for contributing to the recent debate on security *bug collisions* [33]. On the more practical side, this exploratory empirical paper hints that the elaborated clustering effect may be potentially used as a heuristic for optimizing the concrete work associated with sending and receiving vulnerability notifications and related sensitive information. If nothing else, it can be contemplated whether some particular groups of vendors would benefit from targeted or personalized information delivered via electronic mail or by other means.

The discussed reliability problems further hint that US-CERT may have difficulties in database management with respect to systematic tracking of various software vendors that are suspected to be affected by the disclosed vulnerabilities. Software industry has always been affected by continuous mergers and acquisitions, which supposedly make it difficult to deduce which particular software vendors are responsible for some particular software products at some particular instant of time. Therefore, empirical clustering of database contents may help at bringing more rigor to practical database maintenance. The net result would not only benefit scholarly research, but it would also make it easier for the numerous Internet sites to robustly parse the information for tracking and security intelligence purposes. Concrete improvements in the global vulnerability disclosure processes require coordination and trust between different parties, but also the distinct processes and databases should be arguably coordinated.

Acknowledgments

The authors gratefully acknowledge Tekes – the Finnish Funding Agency for Innovation, DIMECC Oy, and the Cyber Trust research program for their support.

References

[1] M. Latapy, C. Magnien, and N. Del Vecchio, "Basic Notions for the Analysis of Large Two-Mode Networks," *Social Networks*, vol. 30, no. 1, pp. 31–48, 2008.

[2] M. E. J. Newman and M. Girvan, "Finding and Evaluating Community Structure in Networks," *Physical Review E*, vol. 69, no. 2, pp. 026 113–1–026 113–15, 2004.

[3] R. H. Griffin and C. L. Nunn, "Community Structure and the Spread of Infectious Disease in Primate Social Networks," *Evolutionary Ecology*, vol. 26, no. 4, pp. 779–800, 2012.

[4] A. Ozment, "Improving Vulnerability Discovery Models: Problems with Definitions and Assumptions," in *Proceedings of the 2007 ACM Workshop on Quality of Protection (QoP 2007)*. Alexandria: ACM, 2007, pp. 6–11.

[5] H. Cavusoglu, H. Cavusoglu, and R. Raghunathan, "Efficiency of Vulnerability Disclosure Mechanisms to Disseminate Vulnerability Knowledge," *IEEE Transactions on Software Engineering*, vol. 33, no. 3, pp. 171–185, 2007.

[6] J. Ruohonen, S. Hyrynsalmi, and V. Leppänen, "Trading Exploits Online: A Preliminary Case Study," in *Proceedings of the IEEE Tenth International Conference on Research Challenges in Information Science (RCIS 2016)*. Grenoble: IEEE, 2016.

[7] Y. S. Baker, R. Agrawal, and S. Bhattacharya, "Analyzing Security Threats as Reported by the United States Computer Emergency Readiness Team (US-CERT)," in *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI 2013)*. Seattle: IEEE, 2013, pp. 10–12.

[8] Carnegie Mellon University, Software Engineering Institute, CERT Division, "Vulnerability Disclosure Policy," 2015, available online in October 2015: http://www.cert.org/vulnerability-analysis/vul-disclosure.cfm?

[9] B. Baudry and M. Monperrus, "The Multiple Facets of Software Diversity: Recent Developments in Year 2000 and Beyond," *ACM Computing Surveys*, vol. 48, no. 1, pp. 16–26, 2015.

[10] M. E. J. Newman, "The Structure of Scientific Collaboration Networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 98, no. 2, pp. 404–409, 2001.

[11] A. Piepenbrink and A. S. Gaur, "Methodological Advances in the Analysis of Bipartite Networks: An Illustration Using Board Interlocks in Indian Firms," *Organizational Research Methods*, vol. 16, no. 3, pp. 474–496, 2013.

[12] Z. Maoz, *Network of Nations: The Evolution, Structure, and Impact of International Networks, 1816 – 2001*. New York: Cambridge University Press, 2011.

[13] J. Bascompte, P. Jordano, C. J. Melián, and J. M. Olesen, "The Nested Assembly of Plant-Animal Mutualistic Networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, no. 16, pp. 9383–9387, 2003.

[14] P. Uetz, L. Giot, G. Cagney, T. A. Mansfield, R. S. Judson, J. R. Knight, D. Lockshon, V. Narayan, M. Srinivasan, P. Pochart, A. Qureshi-Emili, Y. Li, B. Godwin, D. Conover, T. Kalbfleisch, G. Vijayadamodar, M. Yang, M. Johnston, S. Fields, and J. M. Rothberg, "A Comprehensive Analysis of Protein-Protein Interactions in *Saccharomyces cerevisiae*," *Nature*, vol. 403, no. 6770, pp. 623–627, 2001.

[15] S. Field, K. A. Frank, K. Schiller, C. Riegle-Crumb, and C. Muller, "Identifying Positions from Affiliation Networks: Preserving the Duality of People and Events," *Social Networks*, vol. 28, no. 2, pp. 97–123, 2006.

[16] R. L. Breiger, "The Duality of Persons and Groups," *Social Forces*, vol. 53, no. 2, pp. 181–190, 1974.

[17] D. Melamed, "Community Structures in Bipartite Networks: A Dual-Projection Approach," *PLOS ONE*, vol. 9, no. 5, p. e97823, 2014.

[18] Y. Cui and X. Wang, "Uncovering Overlapping Community Structures by the Key Bi-Community and Intimate Degree in Bipartite Networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 407, pp. 7–14, 2014.

[19] Carnegie Mellon University, Software Engineering Institute, CERT Division, "Vulnerability Notes Database," 2015, available online in March 2015: http://www.kb.cert.org/vuls/.

[20] ——, "Vulnerability Note VU#695940," 2015, available online in March 2015: http://www.kb.cert.org/vuls/id/695940.

[21] L. Allodi and F. Massacci, "Comparing Vulnerability Severity and Exploits Using Case-Control Studies," *ACM Transactions on Information and System Security*, vol. 17, no. 1, pp. 1:1–1:20, 2014.

[22] A. Arora, C. Forman, A. Nandkumar, and R. Telang, "Competition and Patching of Security Vulnerabilities: An Empirical Analysis," *Information Economics and Policy*, vol. 22, no. 2, pp. 164–177, 2010.

[23] L. C. Freeman, "Centrality in Social Networks: Conceptual Clarificiation," *Social Networks*, vol. 1, no. 3, pp. 215–239, 1979.

[24] A. Kehagias and L. Pitsoulis, "Bad Communities with High Modularity," *The European Physical Journal B*, vol. 86, no. 7, 2013.

[25] G. Csárdi and T. Nepusz, "The igraph Software Package for Complex Network Research," 2006, InterJournal, Complex Systems CX.18. Available online in June 2014: http://www.interjournal.org/manuscript_abstract.php?361100992.

[26] W. Zhou and L. Nakhleh, "Convergent Evolution of Modularity in Metabolic Networks Through Different Community Structures," *BMC Evolutionary Biology*, vol. 12, pp. 181–195, 2012.

[27] M. T. Schaub, J. Delvenne, S. N. Yaliraki, and M. Barahona, "Markov Dynamics as a Zooming Lens for Multiscale Community Detection: Non Clique-Like Communities and the Field-of-View Limit," *PLOS ONE*, vol. 7, no. 2, p. e32210, 2012.

[28] O. Temizkan, R. L. Kumar, S. Park, and C. Subramaniam, "Patch Release Behaviors of Software Vendors in Response to Vulnerabilities: An Empirical Analysis," *Journal of Management of Information Systems*, vol. 28, no. 4, pp. 305–337, 2012.

[29] R. Telang and S. Wattal, "An Empirical Analysis of the Impact of Software Vulnerability Announcements on Firm Stock Price," *IEEE Transactions on Software Engineering*, vol. 33, no. 8, pp. 544–557, 2007.

[30] W. A. Arbaugh, W. L. Fithen, and J. McHugh, "Window of Vulnerability: A Case Study Analysis," *Computer*, vol. 32, no. 12, pp. 52–59, 2000.

[31] J. Ruohonen, S. Hyrynsalmi, and V. Leppänen, "Software Vulnerability Life Cycles and the Age of Software Products: An Empirical Assertion with Operating System Products," in *Proceedings of the CAiSE 2016 International Workshops, Lecture Notes in Business Information Processing (Volume 249)*, J. Krogstie, H. Mouratidis, and J. Su, Eds. Ljubljana: Springer, 2016, pp. 207–218.

[32] ——, "The Sigmoidal Growth of Operating System Security Vulnerabilities: An Empirical Revisit," *Computers & Security*, vol. 55, pp. 1–20, 2015.

[33] N. Cardozo and A. Crocker, "We Shouldn't Wait Another Fifteen Years for a Conversation About Government Hacking," 2016, Electronic Frontier Foundation (EFF), Deeplinks Blog, August 12, 2016. Available in July 2016: https://www.eff.org/deeplinks/2016/08/we-shouldnt\-wait-another-fifteen-years-conversation-about-government-hacking.