# Software vulnerability disclosure and security investment

Arrah-Marie Jo[*]

May 2019
*Preliminary draft*

**Abstract**

Around the debate on software vulnerability disclosure, existent works have mostly explored how disclosure gives an incentive to software vendors to better secure their software. The role of third parties such as business users, security firms, downstream software vendors or service providers is rarely taken account, while in fact these actors are increasingly involved in improving the security of a software.

In this paper, we examine how the public disclosure of a critical vulnerability impacts not only the software vendor's behavior but most of all that of other third parties. Using data from 2009 to 2018 on vulnerabilities disclosed on SecurityFocus BugTraq, we compare how the contribution of each type of actors in finding new security flaws evolves for the software affected by a critical vulnerability announcement and for the others. We find that the overall number of discovered vulnerabilities increases after a highly publicized vulnerability announcement. In particular, third parties' contribution – more specifically, all actors other than the software vendor and its competitors – are more positively affected by the announcement than the software vendors' contribution. Our findings bring into focus the need to take account the contribution of third party actors when analyzing the incentives in providing software security.

**Keywords:** information security economics, software vulnerability disclosure, vulnerability discovery, security investment

[*]Télécom ParisTech (Institut Polytechnique de Paris). E-mail: arrahmarie.jo(a)gmail.com

## 1 Introduction

In January 2018, Intel revealed that millions of their computer processors were exposed to a critical vulnerability named Spectre and Meltdown.[1] Instead of going public right after its discovery, Intel had exclusively informed some of its main customers and kept the information confidential for more than half a year.[2] The secrecy kept by Intel can be explained by the increasing security risk it would have been exposed to if the vulnerability information became public before they find a solution to secure the flaw (Schneier, 2000). However, it also prevented other firms and users to timely assess their own risk and to react.

In line with Intel's defense, the public disclosure of a vulnerability can be harmful to a system's security because it increases the probability that the disclosed information is exploited by a malevolent actor. Empirical estimates support this idea, showing how the frequency of attacks increases when the vulnerability is disclosed to public (Arora, Nandkumar, and Telang, 2006). On the other side, a greater number of studies, both theoretical and empirical, find that vulnerability disclosure encourages software vendors to deliver patches more quickly and to provide a better software quality over time (Nizovtsev and Thursby, 2007; Cavusoglu, Cavusoglu, and Raghunathan, 2007; Arora, Telang, and Xu, 2008; Arora, Krishnan, Telang, and Yang, 2010). All these papers key on the fact that the disclosure of a vulnerability allows the attackers to specifically exploit the disclosed information and how in turn it affects the users and the software vendors' behavior. Besides, the public disclosure of a vulnerability may also have some effect that is not directly related to the disclosed vulnerability. For instance, Telang and Wattal (2007) show that vulnerability announcements lead to a significant loss in the affected software vendor's market value. In the same vein, we consider that vulnerability disclosure may act as a signal that the actual security quality of the software may be lower than what it was perceived until now. In our case, we study whether the disclosure of a vulnerability on a software gives an incentive to improve its security. We are not only interested in the

[1]`https://meltdownattack.com`

[2]https://www.wsj.com/articles/intel-warned-chinese-companies-of-chip-flaws-before-u-s-government

software vendor's behavior but also of other third party actors that actively contribute to software security, like the (business) users, competitors, security firms, public organizations or individual contributors.

The existent literature considers that users are passive agents which undertake damage control activities (patch installations, work-arounds) rather than actively engage in preventive actions. A dearth of research exists on the role that users can play in improving software security, yet a considerable part of users, especially business users – i.e. companies that use the software, including downstream and upstream software vendors and service providers –, contribute actively and significantly to global cybersecurity. They often manage their own security research and incident response team, pay security firms to secure their systems, collaborate with public CERTs and academic researchers, crowdsource individuals through vulnerability reward programs. In fact, businesses with large scale information systems necessarily have "a lot at stake", hence they have not only the ability but also the incentives to actively find and fix software vulnerabilities. It could be even more the case as the larger a business is – i.e. the larger and complex its information system is –, the less flexible it is in switching from one software to another even though it realizes that the software it uses has a poor security quality. In other words, companies who have big switching cost may prefer to actively collaborate with other actors (the software vendor, security firms, individual security experts, public organizations...) to improve the security of the software rather than waiting for the vendor to provide a better security.

In this paper, we aim at assessing the impact of a vulnerability disclosure on the behavior of each actors that might exert an effort to improve the security of the software affected by the disclosure. We consider the number of vulnerabilities that are discovered by a given actor as a measure of the effort it exerts, i.e. its security investment level.

Specifically, we examine how the public disclosure of a vulnerability with a heavy media coverage affects the discovery of new vulnerabilities in the software that suffer the disclosure. For that, we study the case of three markets – by market we designate a group of software that belongs to the same type and which present strong substitutability

– at the heart of Internet security and which are well-known to standard users and thus generates significant media attention from general media as well as from the security community – namely the web browser market, the desktop operation system and the mobile operation system markets. For each market, we identify a vulnerability that has received a particularly large media coverage and has generated a peak in web search volumes. Then, using data collected from a public vulnerability database on software vulnerabilities – SecurityFocus BugTraq – reported from January 2009 to December 2018, we examine the impact of the disclosure event on the number of vulnerabilities reported by each type of actors over time. We use a difference-in-difference specification in order to measure the change in the level of effort exerted to the software affected by the disclosure compared to other software.

We find that after the disclosure of a critical vulnerability, the vulnerability research activity on the affected software significantly increases compared to other software. Interestingly, actors that are more affected by the disclosure are rather third parties than the software vendor itself. In particular, the number of vulnerabilities discovered by users (including downstream or upstream software vendors), security firms and individual researchers increases, while the effect on competing software vendors is not significant. This result is all the more important as our analysis shows that third party actors contribute more than the software vendor to the discovery of new security flaws in general. Our findings suggest that one should not ignore the incentives and potential contribution of third party actors when studying software security.

This paper is structured as follows. In the next section, we review the relevant literature. Section 3 presents the identification strategy and the data. This section comprises 5 subsections: in Subsection 3.1, we explain how we categorize the different actors that contribute to the security of a software, we present the econometric specification in Subsection 3.2, detail how we build the dependent and the treatment variables in Subsections 3.3 and 3.4. Subsection 3.5 verify the parallel trend assumption and presents some descriptive statistics. We then present the estimation results in Section 4 and Section 5 concludes.

## 2 Literature Review

Our paper draw from three literature streams within the economics of information security.

First, it participates to the debate on whether promoting software vulnerability disclosure is socially preferable. Cavusoglu et al. (2007) and Arora et al. (2008) both develop a theoretical model to identify the optimal timing for vulnerability disclosure, considering the damage cost to users and the patch development cost for the software vendor as societal losses. In both papers, users' and attackers' behavior is exogenously fixed, and they focus on the decisions of the social planner and the software vendor. Their main finding is that, when vendors do not sufficiently internalize user losses, vulnerability disclosure provides an incentive for vendors to secure their product more quickly. Nizovtsev and Thursby (2007) also study the same question and find that vulnerability disclosure has a positive effect on software vendors' responsiveness. Besides, one of the models in Nizovtsev and Thursby (2007) considers the case of open source software where users can actively participate in finding and fixing vulnerabilities. They show that the positive effect of vulnerability disclosure becomes greater when users are able to fix a software by themselves. This idea is in line with the question we study in our paper, since we look at how users' participation in improving security evolves after the disclosure of a critical vulnerability.

In parallel to analytical models, some empirical studies have examined the impact of vulnerability disclosure on the attack frequencies (Arora et al., 2006), on software vendors' market value (Telang and Wattal, 2007), and on software vendors' patch release behavior (Arora et al., 2010). In our paper, we study the impact of the disclosure on the behavior of all actors that actively participate in improving the security of a software, including the users. Our paper also differs from existent empirical works as we are not studying the effect of a vulnerability disclosure that is directly associated to the disclosed information but we rather consider vulnerability disclosure as a signal that updates the perceived security quality of the affected software.

Secondly, our work is related to papers that analyze the consequences of security interdependence. In particular, several papers account for the externality caused between users due to the interdependence of security (Kannan and Telang, 2005; Arora et al., 2008; Choi, Fershtman, and Gandal, 2010). Overall, results suggest that an inequality in users' ability to secure their system or software reduces the social benefits of vulnerability disclosure.

Lastly, a stream of research in the economics of informations security studies software liability and risk sharing mechanisms between users and software vendors. Kim, Chen, and Mukhopadhyay (2009) investigate how risk sharing of security losses between software vendors and users improves software quality. August and Tunca (2006, 2011) analyze the impact of different liability policies to users or/and the software vendor given the security risk and the patching cost. Cavusoglu, Cavusoglu, and Zhang (2008) consider the coordination possibilities between users and vendors in managing security patches. Lam (2016) develops a model where the software vendor can undertake multiple types of security investments and show that giving full liability to the vendor makes him underinvest in attack prevention and over-invest in damage control. It is worth noting that all of these works are analytical. Regarding the coordination issue between the software provider and users, we consider a situation where both the vendor and other third party stakeholders can undertake actions to improve the software security. We particularly focus on the "proactive" role that these third parties can play by actively finding and fixing security flaws in collaboration with the vendor.[3]

## 3 Data and Empirical Strategy

Our goal is to examine how a critical vulnerability disclosure on a software affects the security investment on it. For that, we consider the effort provided by an actor to find new security flaws as a measure of the effort it exerts to improve the security of the software. We study three markets, which attract significant media attention from end users and thus for which we are able to identify a vulnerability disclosure that raised a particular media coverage compared to others: the web browser market, the mobile OS and the desktop OS markets.

[3]Contrary to Lam (2016) that includes vulnerability research and patch development activities in damage control investment and remediation, we consider them as attack-prevention strategies as it reduces the probability of (zero-day) attacks.

In order to study the causal relationship between a vulnerability disclosure and the security investment behavior of each actors that contributes actively to the security of the software, we use a difference-in-difference specification. That is, we compare the difference in the number of vulnerabilities reported by each actors, before and after the extensive media coverage of a security flaw on the targeted software (the treatment group) and the others (the control group).

The main data set we use comes from Security Focus bugtraq, which is a public database on software vulnerabilities. The treatment is identified using Google Trends data. More precisely, we identify a particular vulnerability disclosure that have generated a spike in media coverage compared to all other vulnerability disclosures in a market (i.e. either on web browsers, on mobile OS or on desktop OS). We then examine how this event affects the number of vulnerabilities that are discovered for each software belonging to the market. Each markets are analyzed separately.[4]

In the next subsection (Subsection 3.1), we explain how we categorize the different actors contributing to the security of a software. Then we detail the econometric specification in Subsection 3.2. Subsection 3.3 describes how we build our dependent variable for the three different specifications we use. Then follows 3.4 where we discuss the identification of our treatment variable. Lastly, we verify the parallel trend assumption and detail the descriptive statistics in Subsection 3.5.

## 3.1 Categorization of the actors

Who should invest in security and how to encourage the right actor in the right way is a question at the heart of information security economics. To answer to this question, economists generally focus on the interaction between the software vendor, users, and attackers. However in practice, various other actors come into play. For instance, a company's information system is formed by multiple software; these software use various frameworks and libraries created and maintained by external organizations, they use components, modules and extensions provided by other editors, they communicate with

[4]For the web browser market, the data set is constituted by vulnerabilities that affect only the web browsers and for operation system markets only vulnerabilities that affect the operation systems.

each other and with the outside network through various protocols which guidelines are maintained by public entities... Thus the security of a company's system depend on a multitude of actors that in turn have various dependency each other. In fact, this complexity already exists for the security of a single software. For example, any organization that has some networked data accessible on the Web (e.g., e-commerce companies, website hosting service providers) is necessarily dependent to the security of web browsers, since a web browser is the main tool used to access to the World Wide Web. The security of a web browser is in turn dependent to a multitude of components, from language like Javascript, runtime environment like Adobe Flash, communication protocol and cryptography library like OpenSSL, Plug-ins and web applications... Since the developers and users of each components internalize a part of the security risk, each of them may have an incentive to improve web browsers' security.

Depending on how a given actor values the externality caused by a vulnerability disclosure (or more generally by the security of a software), we can categorize them as following:

- Competitors: by competitors we designate software vendors that play in the same market. Their behavior can be affected by the disclosure in several ways. First, it may have a negative effect on the affected software reputation (i.e. the competing product). This can be an incentive for a competitor to put more effort in finding new flaws in its adversary's product. At the same time, investing in a competitor's product security can be costly and may not be so profitable. Secondly, vulnerability disclosure on one product in the market can deteriorate the overall reputation of the market and thus have a negative effect on the overall demand. Considering this effect, vulnerability disclosure may give an incentive to firms to provide an effort to secure competitors' product as much as theirs. More importantly, products within the same market often share common vulnerabilities. Thus the effort made by a vendor to improve its own product's security may have some spillover on the security of competing products whether it is intentional or not. Overall, it is difficult to predict whether a vulnerability disclosure on a software would have a positive or negative effect on a competitor's effort to improve the security of the software affected by the disclosure.

- Users, downstream and upstream software vendors and service providers: they are dependent to the security of the affected software in various degrees and thus internalize a part of the risk due to vulnerability disclosure. If these actors have the possibility to choose between switching to another product or spending some effort to secure the vulnerable software, their behavior would depend on how high the switching cost is compared to the security investment cost.

- Security firms: these firms provide security solution and services to vendors and users. We include here firms that sell all types of security solutions, from anti-virus software to incident response services, as well as consulting services such as security assessment or penetration testing. The profits of a security firm comes from selling security solutions to its clients whether the client is the software vendor or the users, and finding a new vulnerability increases the value of its services. The disclosure of a new vulnerability can work as a signal that the actual security quality of the software may be worse than it was perceived until now. That is, the discovery of a new vulnerability may be perceived as an opportunity to find some additional vulnerabilities that are not disclosed yet. Thus it can be an incentive to security firms to look more thoroughly at the security of the affected software. Additionally, a security firm which has signed a contract with the software users or has sold a security product to them internalizes a part of the user damage cost. Overall, one can predict that vulnerability disclosure would incentivize security firms to make more effort in finding new security flaws in the affected software.

- Academic researchers, public CERTs, and public organizations:[5] we group in this category actors for which the main goal is to improve global security rather than making their own profit. They may internalize a part of the loss due to a vulnerability disclosure on a software, but this might be insignificant compared to the end users.

[5]Private CERTs are accounted as a private company

- Individuals: in our dataset, the discovery of numerous vulnerabilities are credited to an individual or a group of individuals without an affiliation. Even though they can actually be affiliated to an organization, we consider that when the affiliation is not specified, the discovery of the vulnerability is voluntarily credited to the individual itself. Here, we can relate the motivation of an individual to find and fix security flaws to the intrinsic and extrinsic motives attributed to open source phenomenon, which has been widely dealt in the literature. A vulnerability disclosure can signal the existence of additional undiscovered vulnerabilities and give an incentive to individuals that look for an opportunity to signal their skills to the community.

This categorization suggests that the public announcement of a vulnerability may affect each type of actors for different reasons and in different degrees.

The raw data set we collected from Security Focus Bugtraq indicates in detail who is/are credited for the discovery of each vulnerability. For the purpose of our study, we have categorized them manually into the 5 types of third party actors - as categorized above – and the software vendor. When a vulnerability is credited to more than one contributor, we duplicate the observation as many times as the number of contributors and we attribute to each observation a weight of $\frac{1}{number\_of\_contributors}$.

It is worth noting that this categorization presents some limits in practice. Indeed, a vulnerability identifier does not belong necessarily to only one category. For instance, a security expert can both come from an academic research laboratory, work for a company, be a member of an open source community and participate in a vulnerability research program organized by a security firm, all at the same time.[6]

## 3.2 Empirical specifications

We use a difference-in-difference specification to study how a vulnerability disclosure affects the effort made by an actor to secure the software. This identification strategy allows us in particular to overcome the reverse causality issue between the number of vulnerabilities

[6]For example, if the discovery of a vulnerability is credited to an individual that was payed by a private reward program, we consider that the effort is made by the organization that finance the reward program and not the individual.

and the media coverage intensity that we would have had if we simply used the intensity of a vulnerability media coverage as our regressor.

Specifically, the baseline specification we use is as following:

$$y_{it} = \beta_0 + \beta_{1a} A_i + \beta_{1b} A_i \cdot P_t + \beta_{1c} P_t + FE_i + FE_t + \gamma X_{it} + \epsilon_{it}, \tag{1}$$

Where, $y_{it}$, our dependent variable, is the total number of vulnerabilities affecting software $i$, reported at period $t$ (monthly date). In this first specification, the effort of the different actors are taken altogether and we first focus on how the disclosure affects the global security investment level. $A_i$ (referring to “Affected software") is a dummy which indicates whether software $i$ is the software targeted by the vulnerability disclosure or not, i.e. whether it belongs to the treatment group ($A_i = 1$) or to the control group ($A_i = 0$).[7] $P_t$ (referring to “treatment Period”) is a dummy which is equal to one for the period we consider as “affected" by the critical vulnerability disclosure event. We use 4 alternative specifications for this treatment period: the first 6 months following the vulnerability disclosure ($post6m$), the first year ($post12m$), the two first years ($post24m$), and the whole period after the vulnerability disclosure ($post$). $FE_i$ and $FE_t$ are software and time fixed effects. $X_{it}$ is a vector of control variables at the software level. It includes the $SoftwareAge$ and a dummy which indicates whether the vendor provides support for the software at period $t$ ($EndofLife$). Lastly, $\epsilon_{it}$ is an error term. Our explanatory variable of interest is the interaction term $A_i \cdot P_t$, which represents the difference in the effect of the vulnerability disclosure – the treatment – between the treatment group and the control group. We expect that the sign of the coefficient $\beta_{1b}$ is positive, i.e. that a critical vulnerability disclosure would increase the global effort made in securing the software that suffers the vulnerability disclosure.

[7] In our data set, the treatment group is the software that suffers the disclosure and all other software in the same market belongs to the control group

Next, we estimate the following equation:

$$\begin{aligned} y_{ijt} =& \beta_0 + \beta_{1a}A_i + \beta_{1b}A_i \cdot P_t + \beta_{1c}P_t + \beta_2 A_i \cdot P_t \cdot ThirdParty_j + \beta_3 A_i \cdot ThirdParty_j \\ & + \beta_4 P_t \cdot ThirdParty_j + \beta_5 ThirdParty_j + FE_i + FE_t + \gamma X_{it} + \epsilon_{it}, \end{aligned} \tag{2}$$

where $y_{ijt}$ is the number of vulnerabilities affecting software $i$, discovered by type of actors $j$ at period $t$. $ThirdParty_j$ is a dummy which is equal to 0 if the identifier of the vulnerabilities is the software vendor and equal to 1 if it is a third party actor. The interaction between our treatment variable $A_i \cdot P_t$ and the $ThirdParty_j$ dummy allows us to measure the difference between the impact of the vulnerability disclosure on a third party actor and on a software vendor (the software vendor being the base value).

Lastly, we use the following specification to study the effect of the vulnerability disclosure on each actors separately:

$$\begin{aligned} y_{ijt} =& \beta_0 + \beta_{1a}A_i + \beta_{1b}A_i \cdot P_t + \beta_{1c}P_t \\ & + \sum_{K_j \in Identifier\_Type} \beta_2^{K_j} A_i \cdot P_t \cdot K_j + \beta_3^{K_j} A_i \cdot K_j + \beta_4^{K_j} P_t \cdot K_j + \beta_5^{K_j} K_j \\ & + FE_i + FE_t + \gamma X_{it} + \epsilon_{it}, \end{aligned} \tag{3}$$

where $Identifier\_Type = \{Competitors_j, Users_j, Sec\_firms_j, Individuals_j, Public\_org_j\}$ and $K_j \in Identifier\_Type$ is a dummy equal to one if $j$ belongs to the identifier type K. In this specification, the coefficients of interest are the five different $\beta_2^{K_j}$, which reflect the difference in the effect of a vulnerability disclosure on each actors' behavior, while the base value is the software vendor's behavior.

## 3.3 Dependent variable

Our dependent variable for the first specification (See Subsection 3.2 for the econometric specifications) is the total number of vulnerabilities discovered in each software, each month, while for the second and third specifications, it is the number of vulnerabilities discovered in each software each month by each type of actors. In order to build our dependent variable, we collected information from Security Focus Bugtraq from January 2009 to December 2018 on all the vulnerabilities that affect a software that belongs to one of the three markets we study. The reason we limit our study to this period is because a considerable amount of manual checks is needed to build our data set, especially in order to categorize the actors that have identified each vulnerabilities. We consider that a period of 10 years is large enough to have a robust result. For each vulnerability, we collected the date it was disclosed, the list of the vulnerable software and the identifiers of the vulnerability. For the three specifications, we have a balanced panel data set over time. The description of the variables are reported in Table 4 and the summary statistics are reported in Table 6.

The raw data consists in each vulnerabilities associated to one or multiple software, disclosed at a given date. All the vulnerabilities in our data set have a patch at the date they are disclosed to public.[8] For vulnerabilities that affect more than one product, we have duplicated the observation in order to take the vulnerability into account for each software. Additionally, the discovery of each vulnerability is credited either to an individual, an organization, or a group of individuals and organizations. For each vulnerability, we manually indicated to which type of actors the identifiers belong to. We have categorized the actors into 6 groups: the software vendor, academics and public organizations, users (including downstream and upstream vendors, companies that use the software or provide a service related to the affected vulnerability), security firms and individuals. When a vulnerability is credited to more than one contributor, we replicate the observation as many times as the number of contributors and we attribute to each observation a weight of $\frac{1}{number\_of\_contributors}$.

This raw data at vulnerability level is aggregated at a monthly level in 3 different manners so as to be used in the 3 different specifications we presented in Subsection 3.2. For the first data set, we count the number of vulnerabilities affecting each software each month without considering who is the actors that contributed. In the second data set, we define a dummy variable which indicates whether the identifier of the vulnerability is the

[8]We exclude from our data set vulnerabilities that cannot be fixed with a patch. Within the scope we study, only 2 vulnerabilities belongs to this case.

software vendor or a third party actor (*ThirdParty* dummy). Then we count the number of vulnerabilities in each software each month either by a third party or the software vendor. In the third data set, we count the number of vulnerabilities in each software, each month for each type of actors. An example of how we have built our dependent variable is illustrated in Table 5.

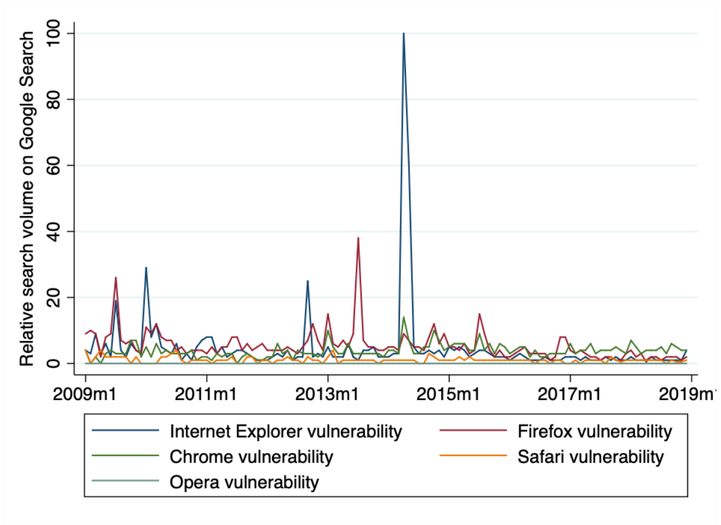
## 3.4 Identification of the treatment variable

Our goal is to measure the impact of a vulnerability disclosure on vulnerability discovery activity. In the three markets we study, an average of 5 to 14 vulnerabilities are reported each month for each software, all severity level taken together (See summary statistics Table 6). Measuring the effect of all of these vulnerability disclosure separately is not possible; we thus focus on the effect of a disclosure that is sufficiently serious and critical to have a significant impact compared to other events. For that, we need to identify a vulnerability that have raised a particularly large media attention compared to other vulnerabilities. By considering a vulnerability that has been particularly critical and highly publicized compared to other, we are able to claim that the effect we attribute to the disclosure is sufficiently large compared to the impact of other events.

In order to identify a vulnerability disclosure that has received a particularly intense media coverage, we use Google Trend (`http://trends.google.fr`), which allows us to visualize the relative evolution of a given search term on Google Search compared to other search terms. Indeed, we consider that the overall information seeking behavior on a search engine is correlated to the magnitude of the media coverage. Specifically, we checked the search trend on Google for the terms that associate the name of a software and the word "vulnerability". For example, in the case of the web browser market, we compare the search trends for the terms "Internet Explorer vulnerability", "Chrome vulnerability", "Safari vulnerability", "Firefox vulnerability" and "Opera vulnerability" (See Figure 1). For each of the three markets we study, we identify one vulnerability that have generated a peak in web search volume according to Google Trend. We detail below the three vulnerabilities identified as our treatments in each market.

- Treatment for the case of web browsers: Microsoft Internet Explorer CVE-2014-1776 Zero-Day disclosed in April 2009

Figure 1: Google Search trend for web browser vulnerabilities from 2009 to 2018



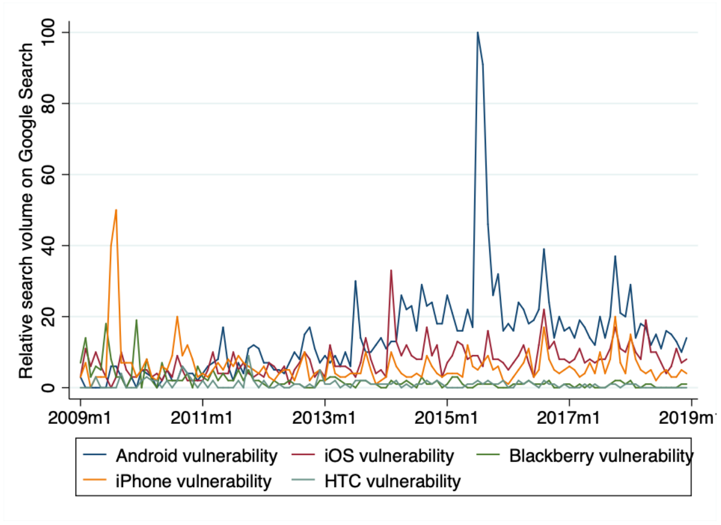
In Figure 1 we observe a peak search volume in mid 2014 for the search term "Internet Explorer vulnerability". This corresponds to a vulnerability that was announced in April 26th 2014 by Microsoft and the security firm FireEye. It is a zero day vulnerability – i.e. a vulnerability that did not have a security patch at the time it was disclosed – which allows an attacker to take full control over the system after a user views a specific web page in its web browser. Its severity scores are evaluated at the highest level of criticality and it affects all existent versions of Microsoft Internet Explorer.[9] The vulnerability was exploited in several targeted attacks. The exact date the vulnerability was discovered and reported to Microsoft is not known, but a patch was published on the 1st May, after the public disclosure. The flaw was so widespread that Microsoft has released patches for Windows versions for which support was already ended.[10]

[9]These scores are called Common Vulnerability Scoring System (CVSS) and prioritize the vulnerabilities according to the threats they represent. Scores are calculated based on a formula that depends on several metrics that approximate the ease of exploit and the impact of exploit. The scores range from 0 to 10, with 10 being the most severe. While the average severity score for web browser vulnerabilities is around 5, this vulnerability presents a score of 10 for every criteria. *Source: the National Vulnerability Database*

[10]*Source:* `https://nvd.nist.gov/vuln/detail/CVE-2014-1776`, `https://blogs.technet.microsoft.com/srd/2014/04/26/more-details-about-security-advisory-2963983-ie-0day/`, `https://www.fireeye.com/blog/threat-research/2014/04/new-zero-day-exploit-targeting-internet-explore` `html`

- Treatment for the case of mobile OS: Google Android Stagefright vulnerability disclosed in July 2015

Figure 2: Google Search trend for mobile OS vulnerabilities



The peak search volume we visualize in Figure 2 corresponds to the disclosure of Android StageFright vulnerability in July 2015. Indeed, the security firm Zimperium announced on July 27th that it had discovered a serious vulnerability in the core of Google Android operation system, which is a flaw related to the way Android handled media, allowing a remote code execution without users opening a malicious file. News headlines announced that nearly a billion of Android devices could potentially be taken over without their users even knowing it.[11] The vulnerability was previously reported to Google in April 2015 and details of an exploit was disclosed at the BlackHat conference in August 2015. Google's security team released a patch for the initial bug within weeks, but it inspired a wave of new attacks on the way Android processes audio and video files. The first copycat bugs were reported just days after the first patch, with more serious exploits arriving months later.[12]

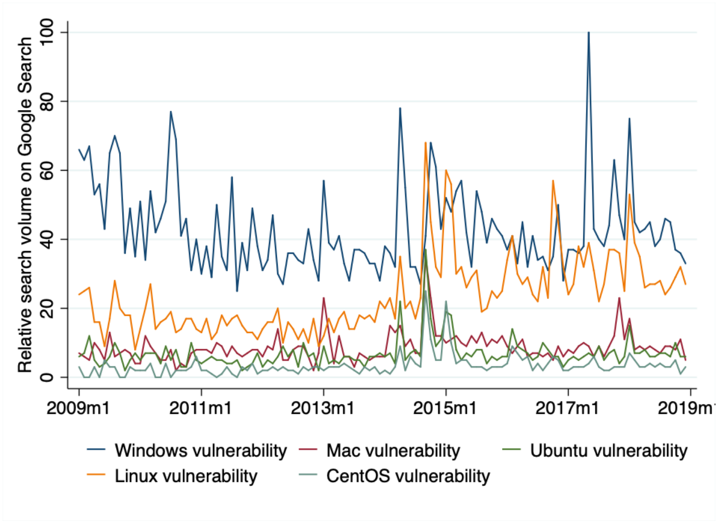
- Treatment for the case of desktop OS: Microsoft Windows Eternal blue and the famous *Wannacry* malware

Figure 3 plots the relative search volumes for terms that are the most popular in

[11]source `https://www.theguardian.com/technology/2015/jul/28/stagefright-android-vulnerability-heartb` `http://blog.zimperium.com/experts-found-a-unicorn-in-the-heart-of-android/`

[12]`https://www.theverge.com/2016/9/6/12816386/android-nougat-stagefright-security-update-mediaserve`

Figure 3: Google Search trend for Desktop and Server OS vulnerabilities



Google search related to operation systems' vulnerability. Note that we have also included CentOS and we include the term Ubuntu while Linux is already included in another search term. Search terms related to other operation systems are not included in the graph because they do not display sufficient search volumes. The peak search volumes occurs in mid 2017, which corresponds to the famous WannaCry ransomware attack happened in May 2017. The WannaCry attack uses an exploit that is originally created by the U.S. National Security Agency (NSA) named *EternalBlue*, which exploits the Microsoft Server Message Block, a network file sharing protocol that allows applications on a computer to read and to write to files within the same network.[13] The exploit was leaked by a hacker group named Shadow Brokers in April 14th 2017 and was used in WannaCry ransomware attack on May 12th 2017. The exploit was also used to carry out the NotPetya cyberattack on late June 2017. Previously, the NSA warned Microsoft after learning about EternalBlue's possible theft, allowing the company to prepare a software patch issued in March 2017, after cancelling all security patches in February 2017.[14] Microsoft released a patch event for Windows XP which support ended in 2014.

The disclosure of these three particularly critical vulnerabilities are considered as the

[13]The vulnerability is denoted by entry CVE-2017-0144 in the Common Vulnerabilities and Exposures (CVE) catalog.

[14]source: Wikipedia

treatments in each market. We consider the software that is targeted by the vulnerability disclosure as the treatment group while other software in the market belongs to the control group. With regard to the treatment period, we consider that a "treatment" begins at the date the vulnerability is disclosed to public, which does not systematically corresponds to the date of the peak search volumes. Indeed, in the case of Microsoft Windows Eternal Blue, the media coverage happens on the day the WannaCry attack happens, which is 2 months after the actual disclosure of the vulnerability. We consider four alternative treatment periods which correspond to a 6 months to 2 year-period after the disclosure. Specifically, we consider the first 6 months after the disclosure, the first year, the first two years, and the whole period after the disclosure as the alternative treatment periods.

Besides, in the three cases we study, the software vendor who is targeted by the critical vulnerability disclosure is alerted about the existence of the vulnerability before the public announcement. This means that an increase in the number of vulnerabilities reported on Security Focus at the moment (just before or just after) the vulnerability is disclosed – our treatment – can be due to an action that does not reflect the actual effort put in vulnerability discovery activity. Indeed, the software editor can suddenly become responsive in patching vulnerabilities that were actually reported by third party identifiers before the critical disclosure happens. To overcome this bias, we exclude all the vulnerabilities that are disclosed during a six months period that surrounds the disclosure date. Indeed, most organizations apply these days a disclosure policy of 90 days.[15] Excluding the last three months preceding the disclosure and the first three months following it insures that we do not take into account the flaws that would have been reported to the vendor before the discovery of the critical vulnerability and which would have been fixed by the software editor in response to the disclosure.

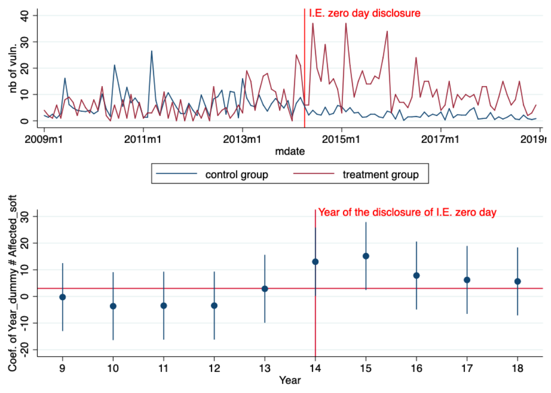
## 3.5 Descriptive statistics

In this subsection, we discuss some descriptive statistics related to the dependent variable and the impact of the treatment. Additionally, we check whether our data satisfies the

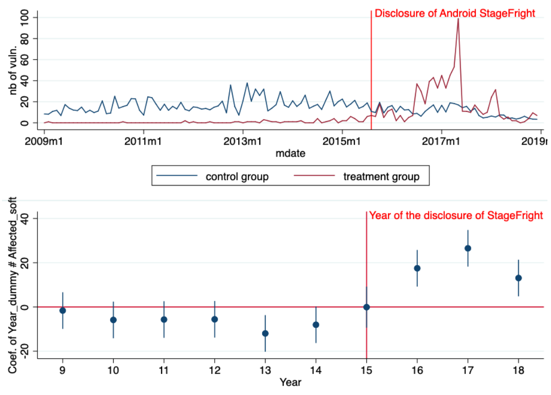
[15]In the case of web browsers, Jo (2017) estimates the average patching time by a software vendor at 88 days.

parallel trend assumption to ensure that we can use a difference-in-difference estimation. Then we discuss the distribution of the different actors' contribution in our data set.

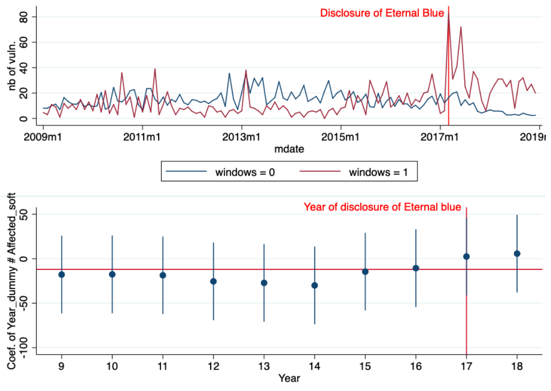
Figure 4: Dependent variable and parallel trend assumption for Specification 1



(a) Web browser case



(b) Mobile OS case

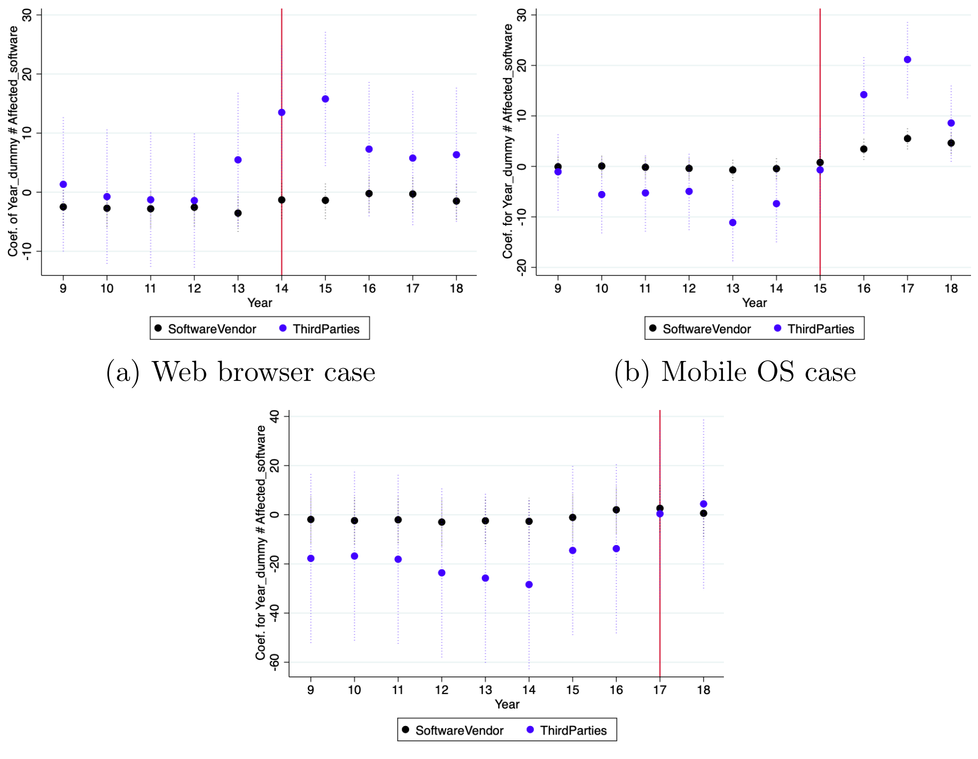


(c) Desktop OS case

In Figures 4a, 4b, and 4c we first plot on the first row (graph on the top side) the evolution of the dependent variable over time, for the treatment group and the control group. The graphs on the second row display the coefficients of the interaction term between the year dummies and the $A_i$ (Affected Software) dummy which identifies whether the observation belongs to the treatment or the control group, with 95% confidence intervals. The plotted estimation includes all the control variables and fixed effects included in Specification 1. This visual inspection allows us to check the validity of the parallel trend assumption.

First of all, in each case, we do not observe any remarkable difference in the evolution of the number of vulnerabilities between the treatment and the control groups, before the critical vulnerability disclosure occurs. Each graph on the second-row also shows that the difference between the treatment and the control groups is not varying significantly over time during the non-treated period. Thus we consider that the parallel trend assumption is satisfied. Then, for each cases, we visualize a significant increase in the number of vulnerabilities after the year the public announcement of a critical vulnerability occurs. We note that for Figure 4a and Figure 4c, the number of vulnerabilities drastically increases just after the disclosure, while it is less the case for Google Android. There are two possible explanations for this immediate reaction. First, as mentioned in subsection 3.4, the sudden increase in the number of vulnerabilities could reflect the software vendor's behavior that suddenly publishes patches for security flaws that were discovered before, to lessen the negative impact of the announcement of a critical vulnerability. We deal with this potential source of bias by excluding a 6-month period before and after the disclosure date. Secondly, firms that actively participate in finding vulnerabilities and securing the software could be alerted in advance about the existence of the flaw before its public disclosure (like the case of Intel we mention in our introduction), which then would not distort our result.

Figure 5: Difference between treatment and control group's outcome, comparison of the software vendor's contribution and third party actors' contribution

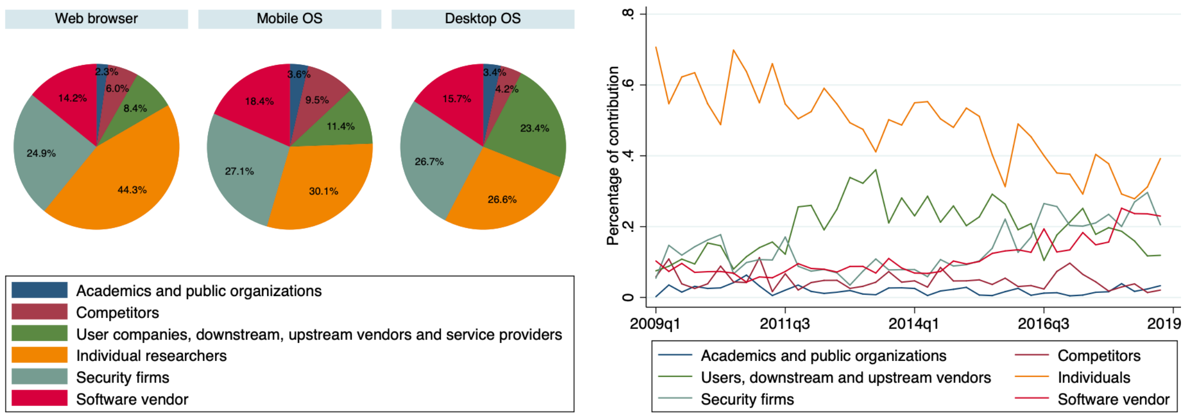


(a) Web browser case (b) Mobile OS case

(c) Desktop OS case

Next, in Figure 5, we plot separately the contribution of the software vendor and third party actors. Specifically, we plot again the yearly evolution of the difference in the number of vulnerabilities between the treatment and control group, but we separate the effort of the software vendor (in black dots) from the third party actors' effort (in blue dots). The graphs clearly show that in each of the three cases, third party actors' contribution increases significantly after the critical vulnerability disclosure occurs, while the change is less significant for the software vendor's contribution.

Figure 6: Distribution of vulnerability identifiers



(a) Distribution of vulnerability identifiers (b) Evolution of the distribution

Lastly, Figure 6 shows the distribution of vulnerability identifiers in the markets we study. Figure 6a plots the distribution of the total contribution in vulnerability discovery in each market and Figure 6b shows the evolution of the distribution over time. We observe that in each market, less than 20 % of the total vulnerabilities are found by the software vendor itself. That is, third party actors contribute 4 times more than the software vendor in discovering new vulnerabilities. In particular, individuals without a precise affiliation contribute the most in general. Interestingly, their contribution is more significant in the web browser market. Note that the contribution of individual researchers is much more considerable in open source software than in closed or mixed source ones.[16] As our estimations account for product fixed effects, effects specific to open source is controlled by the product fixed effect dummies. We also observe that security firm also contributes more than software vendors in general. As to business users' contribution, we note that

[16]Our data set allows us to check this trend but we do not use this information in our estimations as we have a panel data set and we account for software fixed effects

their contribution vary from 8.4% to 23.4% depending on the market. We also observe that the overall contribution of individual researchers decreases over time, while security firms and the software vendor's contribution increase.

## 4 Results

Table 1 reports the estimation results for our baseline specification Eq 1, for the three cases we study. In each column, we report the results for each alternative treatment periods, from the first 6 months after the vulnerability disclosure ($post6m$) to the whole period after the disclosure ($post$).

We have count data and given the presence of significant over-dispersion of the dependent variable with standard deviation superior to the mean, we use a negative binomial regression to estimate the equations. To facilitate the lecture, we only report the main regressors of interest, although all the regressions include the product and time fixed effects as well as other controls ($SoftwareAge$, $EndofLife$) and a constant. For the Eternal Blue case, the last column (4) is empty as our data is limited to the period before 2019 (thus $post24m = post$).

As expected, all the coefficients for $A\#P$ are positive for each of the three markets we study. That is, after the disclosure of a critical vulnerability, the total number of vulnerabilities discovered on the software concerned by the disclosure increases. Specifically, 5 more vulnerabilities are discovered in Microsoft Internet Explorer each month during the two years after the disclosure of a critical Zero Day on it compared to the usual rate of vulnerability discovery on a web browser. In the same way, Android Stagefright vulnerability disclosure has increased the number of vulnerabilities that were found in Google Android by 12 additional vulnerabilities each month during the first two years following the disclosure, while the number of vulnerabilities in Microsoft Windows has increased by 7 more vulnerabilities per month after the disclosure of Eternal Blue. We note that the effect becomes more significant when we consider a longer period as the treatment period, and that the magnitude of the effect also increases with a longer period. This suggests that the vulnerability disclosure does not have an immediate effect on the behavior

Table 1: Effect of a critical vulnerability disclosure on the number of discovered vulnerabilities

Dependent variable: number of reported vulnerabilities

| | (1) | (2) | (3) | (4) |
|---|---|---|---|---|
| treatment period is: | *post*6*m* | *post*12*m* | *post*24*m* | *post* |
| | Case of Internet Explorer Zero Day vulnerability disclosure | | | |
| $A$ (for "Affected software") | 0.160 | 0.244 | 0.0391 | -0.931 |
| | (1.276) | (1.274) | (1.259) | (1.229) |
| $A\#P$ | 0.808* | 0.712** | 1.057*** | 1.775*** |
| | (0.481) | (0.357) | (0.273) | (0.220) |
| $P$ (for "treatment Period") | 0.401 | 0.119 | -0.0522 | -2.165 |
| | (1.215) | (1.308) | (1.414) | (1.771) |
| Observations | 819 | 819 | 819 | 819 |
| Wald $\chi^2$ | 432.85 | 433.99 | 445.66 | 490.81 |
| AME of $A\#P$ | 4.179* | 3.673** | 5.372*** | 8.752*** |
| | (2.510) | (1.867) | (1.449) | (1.271) |
| | Case of Android Stagefright vulnerability disclosure | | | |
| $A$ | 0.0799 | 0.0479 | -0.796*** | -2.001*** |
| | (0.246) | (0.249) | (0.254) | (0.281) |
| $A\#P$ | 0.747 | 0.611* | 2.194*** | 3.332*** |
| | (0.493) | (0.369) | (0.270) | (0.265) |
| $P$ | 0.263 | -0.121 | -0.345 | -1.888* |
| | (0.862) | (0.926) | (0.993) | (1.083) |
| Observations | 1,872 | 1,872 | 1,872 | 1,872 |
| Wald $\chi^2$ | 646.61 | 646.95 | 720.50 | 814.01 |
| AME of $A\#P$ | 4.123 | 3.369* | 12.00*** | 18.23*** |
| | (2.729) | (2.045) | (1.601) | (1.653) |
| | Case of Windows Eternal Blue vulnerability disclosure | | | |
| $A$ | 0.00501 | -0.240 | 0.0300 | |
| | (2.276) | (2.276) | (2.251) | |
| $A\#P$ | 0.686 | 0.648* | 1.320*** | |
| | (0.476) | (0.348) | (0.283) | |
| $P$ | 0.0751 | -0.145 | -0.678 | |
| | (1.040) | (1.099) | (1.137) | |
| Observations | 1,856 | 1,856 | 1,856 | |
| Wald $\chi^2$ | 626.02 | 627.50 | 648.96 | |
| AME of $A\#P$ | 3.676 | 3.464* | 7.066*** | |
| | (2.574) | (1.881) | (1.603) | |

Note: Negative binomial regressions. Product fixed effects and time fixed effects are included in all specifications as well as other controls ($SoftwareAge$, $EndofLife$) and constant. For Eternal Blue case, *post*24*m* = *post*. Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1

of the vulnerability identifiers but rather a gradual effect over time. Two explanation can be advanced. First, discovering new security flaws in a software is not a trivial task; it is not because one put an effort in vulnerability research that it would systematically find some relevant information to improve software security. Secondly, we count the number of vulnerabilities found in a given period using the date each vulnerabilities were disclosed. The actual discovery of the vulnerability might have occurred before the date we consider in our estimations, which means that our result may show a lagged effect. Nevertheless, this imprecision does not alter the main result we are interested in.

Table 2 reports the estimation results using our second specification (Eq 2), in which we examine the effort exerted either by the software vendor or other third party actors separately. Going from column (1) to (3), we consider a longer period as the treated period. Again, we only report the main regressors of interest in order to facilitate the lecture, although all the regressions include the other interaction terms we specified in Specification 2 (the terms $A\#ThirdParty$, $P\#ThirdParty$), as well as product and time fixed effects, other controls ($SoftwareAge$, $EndofLife$) and a constant.

First, all the coefficients for $ThirdParty$ are positive and significant for the three markets we study, meaning that in general, third party actors contribute more in finding new security flaws than the software vendor. The positive sign of the coefficient is as expected, as the distribution of each actors' contribution (see Subsection 3.1) already shows that in general less than 25 % of the vulnerabilities are found by the software vendor itself. Regarding the interaction term between $ThirdParty$ and the treatment variable $A\#P$, the coefficient is not systematically significant, but it is always positive. That is, overall, the behavior of third party actors is more affected by the critical vulnerability disclosure than the software vendor, but the effect is not always significant. Indeed, the effect of the vulnerability disclosure on a given actor could be different from another one among the third party actors but as we encompass all the different actors in one category – the"$ThirdParty$" –, the significant positive effect on some actors could be mitigated by a less significant or negative effect on other actors.

Lastly, Table 3 reports the effect of a critical vulnerability disclosure on each actors'

Table 2: Effect of a highly publicized vulnerability disclosure on the software vendor vs. third party identifiers' behavior

Dependent variable: number of reported vulnerabilities

| | (1) | (2) | (3) |
|---|---|---|---|
| treatment period is: | $post12m$ | $post24m$ | $post$ |
| | Case of Internet Explorer Zero-day vulnerability disclosure | | |
| $A$ | 0.152 | 0.355 | 0.103 |
| | (1.102) | (1.103) | (1.087) |
| $A\#P$ | -0.556 | -1.051** | -0.619* |
| | (0.746) | (0.468) | (0.333) |
| $P$ | 0.826 | 0.446 | 0.463 |
| | (1.074) | (1.147) | (1.230) |
| $A\#P\#ThirdParty$ | 1.500 | 1.870*** | 1.339*** |
| | (0.912) | (0.584) | (0.412) |
| $ThirdParty$ | 1.971*** | 2.034*** | 2.137*** |
| | (0.0970) | (0.100) | (0.106) |
| Observations | 1,734 | 1,734 | 1,734 |
| Wald $\chi^2$ | 992.48 | 998.28 | 1027.34 |
| | Case of Android Stagefright vulnerability disclosure | | |
| $A$ | 1.108*** | 1.031*** | 0.488* |
| | (0.261) | (0.266) | (0.286) |
| $A\#P$ | 0.424 | 0.627 | 1.499*** |
| | (0.661) | (0.488) | (0.369) |
| $P$ | -0.0741 | -0.242 | -0.0654 |
| | (0.816) | (0.871) | (0.938) |
| $A\#P\#ThirdParty$ | -0.0146 | 0.487 | 1.124** |
| | (0.910) | (0.658) | (0.496) |
| $ThirdParty$ | 2.297*** | 2.309*** | 2.365*** |
| | (0.0641) | (0.0656) | (0.0679) |
| Observations | 3,744 | 3,744 | 3,744 |
| Wald $\chi^2$ | 2028.30 | 2036.17 | 2119.79 |
| | Case of Windows Eternal Blue vulnerability disclosure | | |
| $A$ | 0.466 | 0.421 | 0.505 |
| | (2.097) | (2.092) | (2.067) |
| $A\#P$ | 0.675 | 0.121 | 0.549 |
| | (0.629) | (0.463) | (0.378) |
| $P$ | 0.367 | 0.723 | 0.105 |
| | (0.979) | (1.024) | (1.057) |
| $A\#P\#ThirdParty$ | 0.284 | 1.030 | 1.351*** |
| | (0.865) | (0.636) | (0.514) |
| $ThirdParty$ | 2.304*** | 2.376*** | 2.465*** |
| | (0.0648) | (0.0662) | (0.0677) |
| Observations | 3,712 | 3,712 | 3,712 |
| Wald $\chi^2$ | 2016.87 | 2038.32 | 2093.08 |

Note: Negative binomial regressions. $A\#ThirdParty$, $P\#ThidParty$, Product fixed effects and time fixed effects are included in all specifications as well as other controls ($SoftwareAge$, $EndofLife$). For Eternal Blue case, $post24m = post$. Coefficients are average marginal effects. Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1

Table 3: Effect of a highly publicized vulnerability disclosure on each actors ($post24m$ as the treatment period)

Dependent variable: number of reported vulnerabilities

| | (1) | (2) | (3) |
|---|---|---|---|
| | Web browser case | Mobile OS case | Desktop OS case |
| $A\#P$ | -0.475 | 1.321*** | 0.515 |
| | (0.350) | (0.364) | (0.371) |
| $A\#P\#Public\_org$ | 0.455 | 1.042 | 2.232*** |
| | (0.960) | (0.775) | (0.663) |
| $A\#P\#Competitors$ | 0.916* | -1.789 | -15.74 |
| | (0.556) | (28,494) | (2,417) |
| $A\#P\#Users$ | 0.264 | 1.179** | 1.665*** |
| | (0.664) | (0.549) | (0.522) |
| $A\#P\#Individuals$ | 0.826* | 1.126** | 1.017* |
| | (0.472) | (0.512) | (0.524) |
| $A\#P\#Sec\_firms$ | 1.805*** | 0.969* | 0.956* |
| | (0.492) | (0.512) | (0.516) |
| $Public\_org$ | -1.256*** | -1.644*** | -1.603*** |
| | (0.160) | (0.0829) | (0.0814) |
| $Competitors$ | -0.0505 | -0.991*** | -0.831*** |
| | (0.127) | (0.0751) | (0.0724) |
| $Users$ | -1.300*** | 0.518*** | 0.655*** |
| | (0.163) | (0.0665) | (0.0646) |
| $Individuals$ | 1.815*** | 1.552*** | 1.635*** |
| | (0.114) | (0.0651) | (0.0632) |
| $Sec\_firms$ | -0.0633 | -0.276*** | -0.244*** |
| | (0.129) | (0.0695) | (0.0678) |
| Observations | 5,202 | 11,232 | 11,136 |

Note: Negative binomial regressions. $A$, $P$, $A\#K$ and $P\#K$ with $K \in Identifier\_type$, product fixed effects and time fixed effects, $SoftwareAge$, $EndofLife$ are included in all the regressions, but are not reported (See Appendix for more detailed results). $P = post24m$. Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1

behavior. To facilitate the lecture, we report the results using only one specification for the treatment period ($post24m$). Moreover, we only report the coefficients for our main explanatory variables, namely the treatment variable, the interaction between the treatment variable and the $Identifier\_type$ dummies, and the $Identifier\_type$ dummies. The estimation results using other specifications and the coefficients for other variables are reported in Table 7 in Appendix.

With regard to the $Identifier\_type$ dummies, the base line value is the $Software\_vendor$. Estimation results show that actors that contribute the most are the individuals, while academics and public organizations contribute the less. Software vendors contribute less than individual researchers and users that are dependent to the software (including downstream and upstream vendors) but they contribute more than their competitors or the security firms. Note that the $Users$ dummy coefficient is negative for the web browser case, meaning that contrary to the case of operation systems, the contribution of the business users is lower than the software vendor's. This could be explained by the fact that there might be a greater number of companies contributing actively to IT security and which considers that operation systems' security is more important than the security of web browsers. As to the interaction terms between the treatment variable $A\#P$ and the identifier's dummy, we observe that most of the coefficients are positive except for the term $A\#P\#Competitors$. That is, our estimation results suggest that third party actors are more sensible to vulnerability disclosure than the software vendor, except the competing software vendors. Overall, users are the most affected by the vulnerability disclosure, but they are more affected in the case of operation systems than in web browser. On the contrary, security firms react more to vulnerability disclosure in web browsers than in operation systems. While security firms contribute less than software vendors in general to the discovery of new security flaws, their effort is more positively affected by the vulnerability disclosure. Besides, it is interesting to note that the actors that contribute the most are still the individuals that do not specify their affiliation and that they are also affected positively by the vulnerability disclosure.

In sum, the increasing number of vulnerabilities after the disclosure of a critical vulnerability is likely to be largely produced by actors that want to seize the opportunity to find new vulnerabilities, such as the security firms and the individual researchers. The increase in the contribution of companies that are dependent to the security of the affected software – those that we designate as "Users" – after the vulnerability disclosure is also significant and greater than the change in the software vendor's effort to find new security flaws.

## 5 Interpretation and conclusion

By studying the impact of three renowned vulnerability disclosure on three different types of software, we analyze how the vulnerability discovery activity on a software is impacted by the disclosure of a critical vulnerability.

First, our results show that after the disclosure of a critical vulnerability, the number of vulnerabilities that are found in the software affected by the disclosure increases significantly compared to other software. Moreover, the effect becomes greater and more significant over time. Secondly, we find that third party actors are more affected by vulnerability disclosure than the software vendor. Users and individual researchers are not only contributing more than the software vendor in general but their contribution is also more affected by the disclosure. While security firms are contributing less than the software vendor in general, the number of vulnerabilities they find increases after the disclosure of a critical vulnerability. These results are all the more important as (1) existent works on software security focus much more on the behavior of software vendors in providing security than on the contribution of other third party actors, while (2) we show that third party actors' overall contribution in software security is considerable, and (3) their contribution is significantly affected by externalities like the disclosure of a critical vulnerability. Overall, our results suggest that is is important to take account the incentives of third party actors to invest in security to better understand the economic mechanisms behind software security.

With regard to the larger impact of a vulnerability disclosure on users than on software vendors, it may be explained by the fact that the vulnerability disclosure acts more significantly as a negative signal to users than to the software vendor. Indeed, the software vendor may be more aware of its actual security quality than other companies. As to the effect on individuals and security firms, vulnerability disclosure is likely to be perceived as an opportunity to find new security flaws and to benefit from it: security firms would benefit from selling new security solutions, individuals would have more opportunity to gain reputation and peer recognition.

This work is still at its preliminary stage and presents a number of limitations, that present opportunities for future research. First, our findings rely on three specific cases on three markets that are linked each other. The study of an additional case may strengthen the reliability of our results. Secondly, it would be useful to build a theoretical framework that explains the effect we obtain from the data. Especially, it would be possible to model the relation between the software vendor, a company that is dependent to the security of the software (a business user, a downstream or an upstream software vendor), and a security firm that sells security solutions to the company. Lastly, it might be possible to go further in the empirical analysis by building some proxies for the users' switching cost, which would allow to study whether vulnerability disclosure affects the users in different magnitude according to their switching cost.

## References

A. Arora, A. Nandkumar, and R. Telang. Does information security attack frequency increase with vulnerability disclosure? an empirical analysis. *Information Systems Frontiers*, 8(5):350–362, 2006.

A. Arora, R. Telang, and H. Xu. Optimal policy for software vulnerability disclosure. *Management Science*, 54(4):642–656, 2008.

A. Arora, R. Krishnan, R. Telang, and Y. Yang. An empirical analysis of software vendors' patch release behavior: impact of vulnerability disclosure. *Information Systems Research*, 2010.

T. August and T. I. Tunca. Network software security and user incentives. *Management Science*, 52(11):1703–1720, 2006.

T. August and T. I. Tunca. Who should be responsible for software security? a comparative analysis of liability policies in network environments. *Management Science*, 57(5):934–959, 2011.

H. Cavusoglu, H. Cavusoglu, and S. Raghunathan. Efficiency of vulnerability disclosure mechanisms to disseminate vulnerability knowledge. *IEEE Transactions on Software Engineering*, 33(3):171–185, 2007.

H. Cavusoglu, H. Cavusoglu, and J. Zhang. Security patch management: Share the burden or share the damage? *Management Science*, 54(4):657–670, 2008.

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

A.-M. Jo. The effect of competition intensity on software security-an empirical analysis of security patch release on the web browser market. In *Proceedings of the 16th Annual Workshop on the Economics of Information Security (WEIS 2017), San Diego*, 2017.

K. Kannan and R. Telang. Market for software vulnerabilities? think again. *Management Science*, 51(5):726–740, 2005.

B. C. Kim, P.-Y. Chen, and T. Mukhopadhyay. An economic analysis of the software market with a risk-sharing mechanism. *International Journal of Electronic Commerce*, 14(2):7–40, 2009.

W. M. W. Lam. Attack-prevention and damage-control investments in cybersecurity. *Information Economics and Policy*, 37:42–51, 2016.

D. Nizovtsev and M. Thursby. To disclose or not? an analysis of software user behavior. *Information Economics and Policy*, 19(1):43–64, 2007.

B. Schneier. Managed security monitoring: Closing the window of exposure. *Counterpane Internet Security*, 2000.

R. Telang and S. Wattal. An empirical analysis of the impact of software vulnerability announcements on firm stock price. *Software Engineering, IEEE Transactions on*, 33 (8):544–557, 2007.

# Appendix

Table 4: Description of the variables

| Variable | Description |
|---|---|
| $y_{it}$ | The number of vulnerabilities on software $i$ discovered at period $t$ |
| $y_{ijt}$ | The number of vulnerabilities on software $i$ discovered by type of actor $j$ at period $t$ |
| $A_i$ | A dummy which indicates whether software $i$ is the software targeted by the vulnerability disclosure (= 1 if it belongs to the treatment group) |
| $Post6m$ | A dummy which is equal to one if the vulnerability disclosure took place less than 6 months ago |
| $Post12m$ | A dummy which is equal to one if the vulnerability disclosure took place less than 1 year ago |
| $Post24m$ | A dummy which is equal to one if the vulnerability disclosure took place less than 2 years ago |
| $Post$ | A dummy which is equal to one for if the vulnerability is disclosed |
| $SoftwareAge$ | Number of months since the software was launched (Versions are not taken account) |
| $EndofLife$ | A dummy which indicates whether the vendor provides support for the software at period $t$ |
| $mdate$ | Monthly date (used for time fixed effects) |
| $id\_software$ | ID for each software (used for product fixed effects) |

Table 5: Example illustrating how we have built the three data sets

Exemple of raw data set

| id | software | Disclosed date | Credit |
|---|---|---|---|
| 49732 | Android | 12/4/2019 | the software vendor and Individual $\alpha$ |
| 49900 | Android | 27/4/2019 | Individual $\beta$ and a security firm |
| 49999 | Android | 01/5/2019 | Individual $\gamma$ |
| 50206 | Android | 01/5/2019 | Downstream vendor, Individual $\alpha$ and the software vendor |
| 58326 | Android | 29/5/2019 | Public organization |

Aggregated data set for Specification 1

| $y_{it}$ | $software$ | mdate |
|---|---|---|
| 2 | Android | 2019m4 |
| 3 | Android | 2019m5 |

Aggregated data set for Specification 2

| $y_{ijt}$ | $software$ | $mdate$ | $ThirdParty$ |
|---|---|---|---|
| 0.5 | Android | 2019m4 | 0 |
| 1.5 | Android | 2019m4 | 1 |
| 0.33 | Android | 2019m5 | 0 |
| 2.67 | Android | 2019m5 | 1 |

Aggregated data set for Specification 3

| $y_{ijt}$ | $software$ | $mdate$ | $Identifier\_type$ |
|---|---|---|---|
| 0.5 | Android | 2019m4 | Software vendor |
| 1 | Android | 2019m4 | Individuals |
| 0.5 | Android | 2019m4 | Security firms |
| 0.33 | Android | 2019m5 | Software vendor |
| 1.33 | Android | 2019m5 | Individuals |
| 0.33 | Android | 2019m5 | Users |
| 1 | Android | 2019m5 | Public Organizations |

Table 6: Summary statistics

| | | Web browser | | | | | Mobile OS | | | | | Desktop OS | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Variable | Obs | Mean | SD | Min | Max | Obs | Mean | SD | Min | Max | Obs | Mean | SD | Min | Max |
| 1st | $Y_{it}$ | 819 | 5.17 | 7.4 | 0 | 67 | 1872 | 14.07 | 17.7 | 0 | 112 | 1856 | 13.95 | 17.68 | 0 | 112 |
| specification | $A_i$ | 819 | 0.14 | 0.35 | 0 | 1 | 1872 | 0.06 | 0.24 | 0 | 0 | 1856 | 0.06 | 0.24 | 0 | 1 |
| | $post6m$ | 819 | 0.05 | 0.22 | 0 | 1 | 1872 | 0.05 | 0.22 | 0 | 1 | 1856 | 0.05 | 0.22 | 0 | 1 |
| | $post12m$ | 819 | 0.1 | 0.3 | 0 | 1 | 1872 | 0.1 | 0.3 | 0 | 1 | 1856 | 0.1 | 0.3 | 0 | 1 |
| | $post24m$ | 819 | 0.21 | 0.4 | 0 | 1 | 1872 | 0.21 | 0.4 | 0 | 1 | 1856 | 0.17 | 0.38 | 0 | 1 |
| | $post$ | 819 | 0.49 | 0.5 | 0 | 1 | 1872 | 0.34 | 0.47 | 0 | 1 | 1856 | 0.17 | 0.38 | 0 | 1 |
| | $id\_software$ | 819 | 4 | 2 | 1 | 7 | 1872 | 8.5 | 4.61 | 1 | 16 | 1856 | 8.5 | 4.61 | 1 | 16 |
| | $mdate$ | 819 | 2014m2 | 35 | 2009m1 | 2018m12 | 1872 | 2013m12 | 35 | 2009m1 | 2018m12 | 1856 | 2013m11 | 35 | 2009m1 | 2018m12 |
| | $SoftwareAge$ | 819 | 14.62 | 5.16 | 3 | 25 | 1872 | 15.56 | 8.34 | 0 | 35 | 1856 | 15.49 | 8.33 | 0 | 35 |
| | $EndofLife$ | 819 | 0 | 0 | 0 | 0 | 1872 | 0 | 0.05 | 0 | 1 | 1856 | 0 | 0.05 | 0 | 1 |
| 2nd | $Y_{ijt}$ | 1734 | 2.62 | 5.26 | 0 | 59 | 3744 | 6.95 | 13.47 | 0 | 111 | 3712 | 6.89 | 13.45 | 0 | 111 |
| specification | $A_i$ | 1734 | 0.19 | 0.39 | 0 | 1 | 3744 | 0.06 | 0.24 | 0 | 1 | 3712 | 0.06 | 0.24 | 0 | 1 |
| | $post6m$ | 1734 | 0.05 | 0.21 | 0 | 1 | 3744 | 0.05 | 0.22 | 0 | 1 | 3712 | 0.05 | 0.22 | 0 | 1 |
| | $post12m$ | 1734 | 0.1 | 0.3 | 0 | 1 | 3744 | 0.1 | 0.3 | 0 | 1 | 3712 | 0.1 | 0.3 | 0 | 1 |
| | $post24m$ | 1734 | 0.21 | 0.41 | 0 | 1 | 3744 | 0.21 | 0.4 | 0 | 1 | 3712 | 0.17 | 0.38 | 0 | 1 |
| | $post$ | 1734 | 0.52 | 0.5 | 0 | 1 | 3744 | 0.34 | 0.47 | 0 | 1 | 3712 | 0.17 | 0.38 | 0 | 1 |
| | $ThirdParty$ | 1734 | 0.5 | 0.5 | 0 | 1 | 3744 | 0.5 | 0.5 | 0 | 1 | 3712 | 0.5 | 0.5 | 0 | 1 |
| | $id\_software$ | 1734 | 4 | 2 | 1 | 7 | 3744 | 8.5 | 4.61 | 1 | 16 | 3712 | 8.5 | 4.61 | 1 | 16 |
| | $mdate$ | 1734 | 2014m2 | 35 | 2009m1 | 2018m12 | 3744 | 2013m12 | 35 | 2009m1 | 2018m12 | 3712 | 2013m11 | 35 | 2009m1 | 2018m12 |
| | $SoftwareAge$ | 1734 | 13.89 | 5.85 | 0 | 25 | 3744 | 15.56 | 8.34 | 0 | 35 | 3712 | 15.49 | 8.32 | 0 | 35 |
| | $EndofLife$ | 1734 | 0 | 0 | 0 | 0 | 3744 | 0 | 0.05 | 0 | 1 | 3712 | 0 | 0.05 | 0 | 1 |
| 3rd | $Y_{ijt}$ | 5202 | 0.87 | 2.49 | 0 | 34 | 11232 | 2.32 | 5.73 | 0 | 61 | 11136 | 2.23 | 5.73 | 0 | 61 |
| specification | $A_i$ | 5202 | 0.19 | 0.39 | 0 | 1 | 11232 | 0.06 | 0.24 | 0 | 1 | 11136 | 0.06 | 0.24 | 0 | 1 |
| | $post6m$ | 5202 | 0.05 | 0.21 | 0 | 1 | 11232 | 0.05 | 0.22 | 0 | 1 | 11136 | 0.05 | 0.22 | 0 | 1 |
| | $post12m$ | 5202 | 0.1 | 0.3 | 0 | 1 | 11232 | 0.1 | 0.3 | 0 | 1 | 11136 | 0.1 | 0.3 | 0 | 1 |
| | $post24m$ | 5202 | 0.21 | 0.41 | 0 | 1 | 11232 | 0.21 | 0.4 | 0 | 1 | 11136 | 0.17 | 0.38 | 0 | 1 |
| | $post$ | 5202 | 0.52 | 0.5 | 0 | 1 | 11232 | 0.34 | 0.47 | 0 | 1 | 11136 | 0.17 | 0.38 | 0 | 1 |
| | $public\_org$ | 5202 | 0.17 | 0.37 | 0 | 1 | 11232 | 0.17 | 0.37 | 0 | 1 | 11136 | 0.17 | 0.37 | 0 | 1 |
| | $competitors$ | 5202 | 0.17 | 0.37 | 0 | 1 | 11232 | 0.17 | 0.37 | 0 | 1 | 11136 | 0.17 | 0.37 | 0 | 1 |
| | $users$ | 5202 | 0.17 | 0.37 | 0 | 1 | 11232 | 0.17 | 0.37 | 0 | 1 | 11136 | 0.17 | 0.37 | 0 | 1 |
| | $individuals$ | 5202 | 0.17 | 0.37 | 0 | 1 | 11232 | 0.17 | 0.37 | 0 | 1 | 11136 | 0.17 | 0.37 | 0 | 1 |
| | $sec\_firms$ | 5202 | 0.17 | 0.37 | 0 | 1 | 11232 | 0.17 | 0.37 | 0 | 1 | 11136 | 0.17 | 0.37 | 0 | 1 |
| | $id\_software$ | 5202 | 4 | 2 | 1 | 7 | 11232 | 8.5 | 4.61 | 1 | 16 | 11136 | 8.5 | 4.61 | 1 | 16 |
| | $mdate$ | 5202 | 2014m2 | 35 | 2009m1 | 2018m12 | 11232 | 2013m12 | 35 | 2009m1 | 2018m12 | 11136 | 2013m11 | 34 | 2009m1 | 2018m12 |
| | $SoftwareAge$ | 5202 | 13.89 | 5.85 | 0 | 25 | 11232 | 15.56 | 8.34 | 0 | 35 | 11136 | 15.49 | 8.32 | 0 | 35 |
| | $EndofLife$ | 5202 | 0 | 0 | 0 | 0 | 11232 | 0 | 0.05 | 0 | 1 | 11136 | 0 | 0.05 | 0 | 1 |

Table 7: Effect of highly publicized vulnerability disclosure (detailed)

| | Case 1 | | | Case 2 | | | Case 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| treatment period is: | | | | | | | | | |
| P | 1.209 | 1.320 | 1.039 | -0.173 | -0.184 | -0.101 | 0.192 | 0.378 | -0.222 |
| | (0.901) | (0.947) | (1.028) | (0.558) | (0.582) | (0.621) | (0.654) | (0.671) | (0.700) |
| A | 0.708 | 0.922 | 0.910 | 0.600*** | 0.545** | 0.0841 | -0.592 | -0.857 | -0.685 |
| | (0.884) | (0.886) | (0.885) | (0.210) | (0.215) | (0.240) | (1.351) | (1.348) | (1.337) |
| 1.id_identifier_type | -1.467*** | -1.367*** | -1.256*** | -1.812*** | -1.794*** | -1.644*** | -1.758*** | -1.684*** | -1.603*** |
| | (0.152) | (0.155) | (0.160) | (0.0785) | (0.0803) | (0.0829) | (0.0789) | (0.0804) | (0.0814) |
| 2.id_identifier_type | -0.319*** | -0.205* | -0.0505 | -1.034*** | -1.009*** | -0.991*** | -1.011*** | -0.923*** | -0.831*** |
| | (0.119) | (0.122) | (0.127) | (0.0697) | (0.0713) | (0.0751) | (0.0702) | (0.0714) | (0.0724) |
| 3.id_identifier_type | -1.409*** | -1.293*** | -1.300*** | 0.434*** | 0.450*** | 0.518*** | 0.448*** | 0.546*** | 0.655*** |
| | (0.150) | (0.153) | (0.163) | (0.0623) | (0.0638) | (0.0665) | (0.0624) | (0.0635) | (0.0646) |
| 4.id_identifier_type | 1.658*** | 1.718*** | 1.815*** | 1.458*** | 1.469*** | 1.552*** | 1.471*** | 1.555*** | 1.635*** |
| | (0.105) | (0.108) | (0.114) | (0.0610) | (0.0626) | (0.0651) | (0.0608) | (0.0619) | (0.0632) |
| 5.id_identifier_type | -0.240** | -0.156 | -0.0633 | -0.282*** | -0.289*** | -0.276*** | -0.384*** | -0.325*** | -0.244*** |
| | (0.119) | (0.123) | (0.129) | (0.0643) | (0.0661) | (0.0695) | (0.0653) | (0.0667) | (0.0678) |
| P#A | -0.518 | -0.947* | -0.475 | 0.402 | 0.602 | 1.321*** | 0.591 | -0.0118 | 0.515 |
| | (0.793) | (0.491) | (0.350) | (0.658) | (0.483) | (0.364) | (0.617) | (0.454) | (0.371) |
| 1.id_identifier_type#P#A | 1.293 | 1.039 | 0.455 | -20.17 | 0.0871 | 1.042 | 0.0663 | 0.699 | 2.232*** |
| | (1.524) | (1.296) | (0.960) | (26,854) | (1.083) | (0.775) | (1.282) | (0.891) | (0.663) |
| 2.id_identifier_type#P#A | -0.292 | 1.086 | 0.916* | -0.667 | -0.486 | -1.789 | -15.47 | -14.99 | -15.74 |
| | (1.215) | (0.756) | (0.556) | (27,412) | (12,213) | (28,494) | (2,577) | (2,314) | (2,417) |
| 3.id_identifier_type#P#A | -21.03 | 1.169 | 0.264 | -1.721 | -0.408 | 1.179** | 0.712 | 2.051*** | 1.665*** |
| | (39,847) | (1.023) | (0.664) | (1.150) | (0.736) | (0.549) | (0.866) | (0.639) | (0.522) |
| 4.id_identifier_type#P#A | 0.452 | 1.169* | 0.826* | 0.670 | 0.953 | 1.126** | 0.0696 | 0.616 | 1.017* |
| | (1.068) | (0.668) | (0.472) | (0.925) | (0.671) | (0.512) | (0.874) | (0.646) | (0.524) |
| 5.id_identifier_type#P#A | 3.808*** | 2.760*** | 1.805** | -0.701 | -0.105 | 0.969* | -0.449 | 0.280 | 0.956* |
| | (1.250) | (0.705) | (0.492) | (0.942) | (0.674) | (0.512) | (0.862) | (0.635) | (0.516) |
| Observations | 5,202 | 5,202 | 5,202 | 11,232 | 11,232 | 11,232 | 11,136 | 11,136 | 11,136 |

Note: Negative binomial regressions. Product fixed effects and time fixed effects are included in all specifications as well as other controls ($SoftwareAge$, $EndofLife$). Coefficients are average marginal effects. Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1