The effect of competition intensity on software security - An empirical analysis of security patch release on the web browser market*

Arrah-Marie Jo†

December 2017

Abstract

This paper examines the effect of competition intensity on software vendors security investments. We consider two aspects that reflect the competition intensity in a market: market concentration and the dominant position of a firm. We first develop a formal model of competition where the user demand depends only on product quality and investigate whether equilibrium levels of security quality increase as competition intensifies. Then, we test the model's predictions using a 10-year pooled cross-sectional data set on web browser vulnerabilities discovered and patched from 2007 to 2016. Contrary to many empirical works examining the link between competition and quality, we find that market concentration is not necessarily harmful to quality provision: a higher market concentration positively impacts the vendor's responsiveness in patching vulnerabilities, although this effect is reduced when the vendor is too dominant.

Keywords:

1 Introduction

As society gets increasingly dependent on networked computers and the Internet, practitioners warn against the danger of having homogenous systems. In 2003, a group of leading cyber-security experts including Bruce Schneier published a report claiming that "IT monoculture" is a major threat to global computer security (Geer, Bace, Gutmann, Metzger, Pfleeger, Quarterman, and Schneier, 2003). They argue that market concentration magnifies security risks because first, when users rely on a single system they are all subject to the same vulnerabilities and as such to the same attacks; secondly, a

*I thank my thesis advisor Marc Bourreau for his patient guidance and support. I also thank Rainer Böhme, Grazia Cecere, Jens Grossklags and Thomas Maillart for valuable suggestions. I also thank the anonymous reviewers at WEIS 2017 (San Diego), audience of AFSE 2016 (Nice), CRESSE 2016 (Rhodes), EARIE 2016 (Lisbon), and JEI 2016 (Palma).

†Telecom Paris Tech. E-mail: arrah-marie.jo@telecom-paristech.fr

dominant software vendor has less incentive to provide a good security level because of its strong market power.

A decade later, facts still support actively the idea that software monoculture is harmful to global security, as illustrated by the recent outbreaks of Wannacry ransomware and Petya wiper malware in early 2017. Both malware affected thousands of computers in no time and paralyzed critical infrastructures in numerous countries, from hospitals in England, telecom, gas and electrical companies in Spain, an Ukrainian nuclear power plant, airports and central bank, to ports of Mumbai and Los Angeles. In both cases, only systems running on Microsoft Windows OS – which currently still dominates the OS market for desktop – were vulnerable.[1]

Homogenous systems visibly enlarge the number of potential victims. Nonetheless, it is not clear whether a dominant position actually reduces the vendor's incentive to secure its product; in the case of WannaCry and Petya, Microsoft had delivered a security patch two months before the attacks.[2] Likewise, Google is now dominant in several markets such as the mobile OS market or the web browser market, but it seems difficult to prove that it provides a lower security quality then it would have if the market were more competitive.[3]

In this paper, we propose to analyze in greater detail and with empirical evidence, the relationship between competition intensity and software vendor's security investments, which will help to better assess the impact of "software monoculture" on cyber-security.

We study particularly two aspects that reflect competition intensity: market concentration and the dominant position of a firm. We first present a simple theoretical model in which firms compete in product's security quality and examine how the security investment level of a firm changes according to the number of competitors in the market. The model shows that the smaller the number of firms competing in the market, the higher the equilibrium level of security investment of a firm. However, when we account for the existence of an installed base of loyal users, the negative impact of competition on the security level choice becomes less clear: the larger the market share of the dominant firm, the less users are responsive to changes in security quality, and therefore, the lower the dominant firm's incentive to provide a higher security level.

Then we test empirically the model's predictions, using data on security patch release decisions on the web browser market for a period of 10 years. We compiled a pooled cross-sectional data set of

[1]The total market share of Microsoft in the desktop OS market in 2017 was 87.0% according to Statcounter.com.

[2]Both WannaCry (struck in May 2017) and Petya (struck in June 2017) exploited the same vulnerability, called EternalBlue, for which Windows delivered a patch in March 2017.

[3]In 2017, the market share of Google Android in the mobile OS market was 64.2% and Google Chome had 53.2 % of the web browser market according to Statcounter.com

586 web browser vulnerabilities, discovered and patched from January 2007 to December 2016. We find strong evidence that higher market concentration positively impacts the vendor's responsiveness in patching vulnerabilities. Nevertheless, this positive impact is reversed when the vendor's position is too dominant: we find that being in a dominant position reduces the positive effect of having less competitors on the responsiveness of the vendor. Moreover, for a given number of firms competing in the market, the more dominant the firm is, the less rapid it is in releasing security patches.

By considering the case of the web browser market, we study a market in which the good is provided free of charge to consumers. Since the software is offered for free to consumers, vendors primarily compete on the basis of quality. Indeed, free software and especially free web applications such as web browsers are primarily or even exclusively financed through the exploitation of user data.[4] In these markets, vendors compete in quality in order to attract users and derive revenue from another market that makes use of their web traffic. The case of free software is all the more of interest as it is very common in today's digital markets. Our study provides policy makers with empirical evidence on whether competition acts in the same way on firms' security investment incentives in such markets as it does in the case of one-sided business models.

The rest of the paper is organized as follows. In Section 2, we review the relevant literature. Then we describe the specificities of the web browser market and its revenue model in Section 3. Section 4 presents the theoretical model. We then describe the link between the security quality of a software and vulnerability patching and the econometric model in Section 5. Section 6 presents the data, estimation results follow in Section 7 and the final section provides some conclusions.

2 Literature review

This paper contributes to two main streams of research: economics of information security, and the relation between competition and quality provision.

The literature on the economics of information security is recent and thriving; it aims at studying the potential market failures causing information systems insecurity. Vulnerability discovery and patch management are one of the topics at the heart of this field. Our paper contributes empirically to this literature by studying the relationship between market structure and software vendors' security provision behavior, considering their patching decisions as a measure of the security quality they provide.

[4]Web browsers are primarily financed through search engines' revenue, which in turn are mostly financed through online advertising.

Analytical works on information security deal with various questions, from the welfare implications of different liability and information disclosure policies (August and Tunca, 2011; Choi, Fershtman, and Gandal, 2010; Arora, Caulkins, and Telang, 2006), coordination between software vendors and customers in the patch management process (Cavusoglu, Cavusoglu, and Zhang, 2008) to the role of different vulnerability discovery mechanisms (Kannan and Telang, 2005). Some theoretical papers account for the impact of market structure and competition intensity on vendors' security provision behavior, but their focus is on firms' cooperation decisions such as information sharing, damage costs and investment cost sharing (Kim, Chen, and Mukhopadhyay, 2009; Gal-Or and Ghose, 2005).

On the other hand, empirical work have essentially focused on issues related to vulnerability information disclosure (Campbell, Gordon, Loeb, and Zhou, 2003; Telang and Wattal, 2007; Gordon, Loeb, Lucyshyn, and Sohail, 2006; Arora, Krishnan, Telang, and Yang, 2010b). Arora et al. (2010b) exploit similar data as ours to investigate the vendor's responsiveness to vulnerability information disclosure and the impact of information disclosure on the frequency of cyber attacks. Ransbotham, Mitra, and Ramsey (2008) use security alerts data from a private security service provider to examine the impact of vulnerability disclosure on the risk of cyber attacks. Our empirical model accounts for the impact of vulnerability information disclosure; however, it is not our main subject of interest.

To the best of our knowledge, only one paper has studied the impact of competition on software publishers' patching behavior (Arora, Forman, Nandkumar, and Telang, 2010a). Our approach differs to theirs in several aspects. First, Arora et al. (2010a) analyze software vendors' reaction with regard to the patching behavior of other vendors that are affected by the same vulnerability. They consider that sharing a common vulnerability put vendors in a competitive situation, whether they are actually operating in the same market or not. In our case, we consider a narrower definition of competition by limiting our analysis to interactions within a single market and using market concentration as a measure of competition intensity. Secondly, we focus on a specific software market – the web browser market – in which vendors adopt a particular revenue model, providing the good free of charge to users. Lastly, we study a relatively long period of time – a ten-year-period – while Arora et al. (2010a) consider a large panel of different software during a shorter period.[5]

In order to define our empirical hypotheses, we start by examining theoretically the quality choice a firm would make, considering the number of competitors in the market and its market position. Our model applies to the case of information goods which are offered free of charge to consumers. The link between competition and quality has been a topic of interest in the literature for a long time, essentially

[5]Our data covers a 10-year-period from 2007 to 2016, while Arora et al. (2010a) consider a 4-year-period from 2000 to 2003.

in product differentiation models. Nevertheless, few papers consider a framework where firms compete in quality, and no existing model corresponds to the case we study. Among the few papers dealing with free products, Waterman (1990) finds that firms invest more in quality when consumers do not bear any cost; but his focus is on the effect of competition on product diversity. Argenton and Prüfer (2012) study competition between search engines. They assume that the larger the installed base of users, the less costly it is for a search engine to provide a given search quality. They then show that the competitive advantage of having a larger installed base allows the dominant firm to drive out its competitors, which leads to a stable monopoly. As they do, we model competition as a tournament between web browser publishers that choose simultaneously their security quality.

The relationship between competition and quality provision has been studied empirically in a broad range of industries, including the airline industry (Mazzeo, 2003), banking (Cohen and Mazzeo, 2004), the hotel industry (Mazzeo, 2002), legal services (Domberger and Sherr, 1989) and software (Arora et al., 2010a). All of these papers find that an increasing competition leads to higher quality provision, while our results are more nuanced. It is worth noting that our paper is one of the first ones that studies the impact of market competition on software (security) quality, and the first one that considers the case of "free" products where firms compete in quality.

3 The web browser and its revenue model

A web browser is a software that gives access to information resources on the World Wide Web (WWW). Its primary role is to retrieve and display content from remote web servers using the HyperText Transfer Protocol (HTTP).

Today, the most popular web browsers are Google Chrome, Mozilla Firefox, Microsoft Internet Explorer, and Apple Safari. Figure 1 shows the evolution of their market shares from 2006 to 2016. Although it is a market with a high degree of concentration – the four most popular browsers account for more than 80% of the market during our observation period – it is also a market characterized by strong entry: more than a hundred of web browsers have entered the market since 1990 (See Figure 2).

In the 1990's, web browsers were sold at a unit price or included in the application suite of an operation system. With the development of search engines, web browsers have become free of charge to users because of the importance of the web traffic they generate. Search engines realized they needed the web browser users in order to improve their algorithm and to generate revenues from advertising. Consequently, search engine companies started negotiating partnerships with web browsers to direct

Figure 1: Web browsers' market shares from 2006 to 2017.

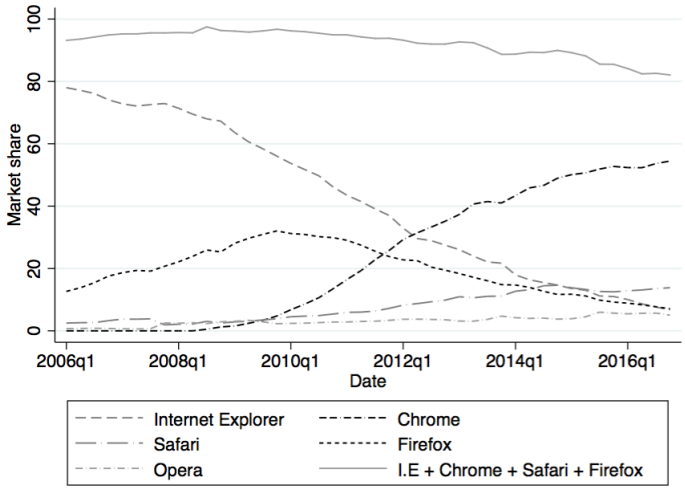
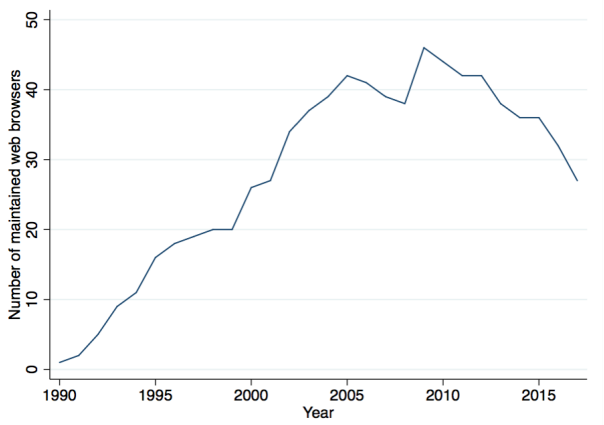


Figure 2: Number of web browsers maintained from 1990 to 2017.



web browser users towards their search engine. Nowadays, they also take advantage of their own web browser in case they have one, such as Google with Google Search and Chrome or Microsoft with Bing and Internet Explorer (See Table 5 and Figure 6 of the Appendix). Indeed, any words we type on a web browser's search bar, which is not a hyperlink, results in a request to a search engine.

Since web browsers are offered free of charge, web browser publishers primarily compete on the basis of quality. Indeed, as a consumer does not bear any financial cost to acquire the product, his utility is not a function of the price, but of other characteristics that he values and in particular of the product's quality. In the case of web browsers, consumers will principally evaluate how much it is secured (protection from data breaches, viruses, malware..), how fast it is (browsing and rendering speed), and how user-friendly it is. It is conceivably the reason why Google heavily communicates on its web browser's simplicity, performance and security for instance.[6] In this paper, we consider that

[6] "Speed, simplicity and security are the key aspects of Google Chrome": When launching Google Chrome, the official

the quality of a web browser corresponds to its security level.

Lastly, there may be a discrepancy between consumers' choice of web browser and the true quality of a product. First, because quality presents some degree of subjectivity. This aspect is disregarded in our work. Secondly, because of consumers' loyalty. Consumers can be loyal because they are imperfectly informed about the security quality or about the available choices, as well as because of the existence of switching costs.[7] In the theoretical model we present in the next section, we account for the existence of an installed base of loyal consumers who stick to their choice of web browser no matter the changes in each web browser's security quality. This allows us to model situations where a firm benefits from a larger installed base than its rivals.

4 A model

We have seen that web browser publishers offer their software free of charge to users and derive revenues from a neighboring market that monetize the browsing traffic.

The question we would like to investigate in this section is whether the security investment level for a free software increases or decreases as competition intensifies. For that purpose, we develop a formal model of oligopolistic competition where firms compete in security quality and study how the equilibrium levels of security investment change as the number of firms competing in the market increases. The web browser market motivates our study; however, the model would also fit to other markets with a similar revenue model.

We consider n firms, labeled 1 to n. Each firm $i \in \{1, \ldots, n\}$ offers one product - a web browser - for which it must choose a security quality level s_i. We assume that the security quality s_i is measurable, with values in $[0, \infty)$. Moreover, each firm has an installed base of loyal consumers $b_i \in [0, 1]$, where $\sum_{i=1}^{n} b_i \leq 1$. Loyal consumers stick to their choice of web browser, whereas non-loyal consumers can switch. For simplicity, we assume that firms are symmetric except for the size of their installed base of loyal consumers.

On the demand side, there is a unit mass of consumers, each of which acquiring one web browser. The market is fully covered.[8] We assume that consumers' utility depends only on the security quality level and that the revenue a software publisher derives per user is exogenous.

press events and press release communicated about Chrome web browser using key words such as "lightweight", "fastest", "speed", "simplicity", "secure", "open source". Source: https://googleblog.blogspot.fr/2009/11/releasing-chromium-os-open-source.html

[7] Indeed, although no contractual or compatibility cost exist in the case of web browsers, changing from a web browser to another may imply important migration costs and the loss of connected services.

[8] The assumption of a fully covered market is consistent with the web browser market, where every Internet user needs the product to surf on the Internet and there is no significant cost that of using the product.

Similar to Argenton and Prüfer (2012), we model competition as a tournament between web browsers with simultaneous security quality choices. More precisely, firms choose simultaneously their security quality and demands are allocated to firms in proportion to their relative security quality.

We assume that the marginal cost of production is equal to zero, which is a standard assumption in the literature on information goods (e.g; Arora et al. (2006); Kim et al. (2009); Choi et al. (2010)). Moreover, as it is standard in studies on quality provision (see, for instance, Allen (1984) and Ronnen (1991)), we consider an increasing and convex cost function for security quality investments. More precisely, a firm i has to invest $\phi s_i^2/2$ to deliver a security quality level s_i, where $\phi > 0$ is the fixed cost parameter for quality investments.

Denoting by a the per-capita revenue and by $B = \sum_{j=1}^{n} b_j$ the total installed base of loyal consumers in the market, firm i's profit is then:

$$\pi_i = a \left[b_i + (1 - B) \frac{s_i}{\sum_{j=1}^{n} s_j} \right] - \frac{\phi s_i^2}{2} . \tag{1}$$

The first-order condition (FOC) of profit maximization with respect to s_i is:

$$\frac{\partial \pi_i}{\partial s_i} = a(1 - B) \frac{\sum_{j=1}^{n} s_j - s_i}{(\sum_{j=1}^{n} s_j)^2} - \phi s_i = 0 . \tag{2}$$

Since $\partial^2 \pi_i / \partial^2 s_i = -2a(1 - B) \sum_{\substack{j=1 \\ j \neq i}}^{n} s_j / (\sum_{j=1}^{n} s_j)^3 - \phi < 0$, the second-order condition is always satisfied. Solving for the FOC (2), we obtain the symmetric equilibrium security quality level for each firm:

$$s_i^* = \sqrt{(1 - B) \cdot \frac{n - 1}{n^2} \cdot \frac{a}{\phi}} . \tag{3}$$

First of all, the following proposition outlines the main insight of this model:

Proposition 1. *In an oligopolistic market where firms compete in security quality, the security level provided by a firm decreases as the number of competitors increases (i.e., $\delta s_i^* / \delta n < 0$).*

Equation (3) shows that the security level provided by a monopolist is zero in our model. This is because security is costly and does not lead to market expansion. Therefore, in this setting, a monopolist would not invest. The equilibrium level of security then increases from a monopoly to a duopoly market. Then, for $n \geq 2$, the smaller the number of firms n, the higher the equilibrium level

of security investment of each firm. This result comes from the fact that, as the number of firms in the market decreases, firms can attract a larger share of consumers by investing in security.

All in all, market concentration has a positive effect on the security level provided by a firm.

Secondly, the security investment chosen by a vendor depends not only on the number of competitors in the market but also on the total share of loyal consumers B. Specifically, let us apply equation (3) to the case of a firm that highly dominates the market. Assuming that firm i's share of loyal consumer is large enough to ignore the effect of the rest of the market – that is, $B = b_i = \alpha_i m_i$, where $m_i \in (0, 1]$ is the firm's total market share and $\alpha_i \in (0, 1]$ the share of loyal consumers among firm i's consumers – we have that:

Proposition 2. *The security quality chosen by a highly dominant firm decreases with respect to its market share ($\partial s_i^*/\partial m_i < 0$).*

Proposition 3. *When a firm highly dominates the market, the positive effect of market concentration on the security level it provides is reduced ($\partial^2 s_i^*/\partial n \partial m_i < 0$).*

The equilibrium level of security quality is always lower than what it would be in a situation without an installed base of loyal consumers. This corresponds to the intuitive idea that the existence of consumers that are less responsive to quality changes reduce the firms' incentive to provide a higher quality. Moreover, we see the limits of considering only the number of firms competing in the market as a measure of market concentration when measuring the actual effect of competition intensity on security investment.

5 Empirical specification

We seek to examine the effect of competition intensity on security investments by software publishers. For this purpose, we consider the time a software publisher spends to release a patch when it discovers a *vulnerability* in its software – which we call the patching time – as a measure of the effort it makes to improve the security of its product.

In this section, we first describe the lifecycle of a software vulnerability and we make the link between the actions conducted by a software vendor and the exposure risk of the software throughout the vulnerability lifecycle. This allows us to explain the rationale behind considering the responsiveness of a software vendor in fixing a security failure as its security investment level. We then present in detail the econometric specification.

5.1 Software vulnerabilities and security quality

In information systems and computer security, a vulnerability refers to a weakness in the architecture, design, or code, which leaves the software or the system open to potential for exploitation.[9] Note that the notion of a vulnerability is different from a *bug*, employed to designate a situation where the system or the program is not behaving as it was designed to behave. In comparison, a vulnerability is a way of abusing the system.

Figure 3: Lifecycle of a vulnerability.

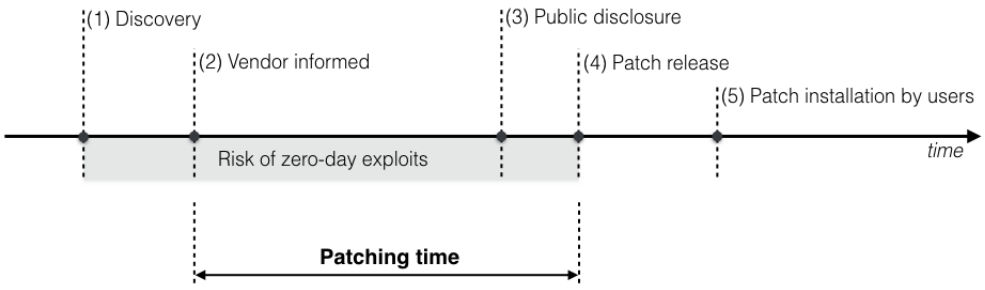


Figure 3 presents the major events that may occur during the lifecycle of a software vulnerability, from its discovery (either by a malevolent or a well-intentioned actor) to its securing by the delivery of a patch and its installation. This timeline is consistent with its representation by Ioannidis, Pym, and Williams (2012), Beres, Griffin, Shiu, Heitman, Markle, and Ventura (2008), or Frei, May, Fiedler, and Plattner (2006). Five key events outline the different phases during the lifecycle of a vulnerability: (1) the discovery of the vulnerability, (2) the vendor being informed about the existence of the vulnerability, (3) the public disclosure of the vulnerability, (4) patch release by the vendor, and (5) patch installation by the user.

The vulnerability may be discovered first by an agent who does not work for the affected software vendor (1). We call it then a *zero-day vulnerability*, since the software vendor has zero day left to deliver a patch before someone exploits the vulnerability. The exposure risk of the software can grow very quickly: information about the vulnerability can be kept by a discoverer who does not have any malicious intent, but it also may well be discovered by a hacker who releases an exploit and make it freely accessible on the Internet.

The software vendor can be notified by a researcher or an organization of the existence of the vulnerability before it finds it by itself (2). Indeed, some private organizations are specialized in vulnerability research and help software publishers and companies – the users of information systems –

[9] Source: `https://cwe.mitre.org`.

to identify security flaws and to secure them. These security firms, also called *security infomediaries*, pay researchers for the discovery of vulnerabilities and exchange with vendors to provide protection solutions for their clients.[10] Furthermore, an increasing number of software vendors manage their own security research team or offer bounties for vulnerability discovery. It is precisely these vulnerability research programs which are the main data sources of our study, namely Tipping Point's Zero Day Initiative (ZDI) program, Verisign's iDefense Vulnerability Contributor Program (VCP), and Google Project Zero.

From the moment the vulnerability is discovered, each day that passes without a patch increases the exposure risk of the software (Schneier, 2000; McGraw and CTO, 2004). Thus, the security quality of a software depends on how quickly the vendor releases a security fix (Arora et al., 2006). In our empirical model, we consider the promptness of the software vendor to provide security patches as a proxy of its investment level, which determines the software's security quality. More precisely, we measure the duration between the moment the software publisher is informed about the existence of the vulnerability and when it releases a patch. This duration corresponds to the difference between time (4) and time (2) in the timeline in Figure 3.

Two main considerations will affect the vendor's patching decision (4): the cost to develop the patch and the extent to which it internalizes the user losses related to the security flaw (Arora et al., 2010a). In more practical terms, the vendor will consider a number of factors such as the importance of the flaw in terms of security, the impact on its reputation especially when the information is disclosed to the public, the complexity to fix the flaw, the human resources available to develop the patch, or the competitive pressure. In our paper, we are particularly interested in the impact the presence of competitors may have on the vendor's patching decision, when everything else equal.

Vulnerability information can be publicly disclosed before the vendor releases a security fix (3). Though the public disclosure of vulnerability information increases the exposure to attack, many advocate that disclosure makes the vendors more responsive in patching their software (e.g., see Arora et al. (2006)). Our study accounts for the impact of vulnerability information disclosure although it is not our main focus. It is rather used as one of the controls that help to identify the effect of the competitive pressure.

Lastly, a system affected by the vulnerability will be exposed to security risks until the user actually installs the patch (6). Some researchers discuss the responsability of sofware vendors in users' patch installation (Cavusoglu et al., 2008), but it is beyond the scope of this paper.

[10] iDefense and Tipping Point are the two major security firms active in this market.

5.2 Econometric model

Our goal is to examine the effect of competition intensity on the time a software publisher spends to release a patch, taking into account other exogenous factors that can have an impact on its responsiveness. Relying on the theoretical results in Section 4, we particularly focus on two aspects of competition: market concentration and the dominant market position of a firm.

First, to estimate the effect of market concentration on the responsiveness of the vendor affected by the vulnerability, we define the following equation:

$$\begin{aligned} Patching\_time_{ij} = \beta_0 + \beta_1 Concentration \\ + \beta_2 X_j + \beta_3 X_i + \beta_4 Disclosure_j + \beta_5 Time\_effect + \epsilon\,, \end{aligned} \tag{1}$$

where $Patching\_time_{ij}$ is the time spent by the web browser publisher i in releasing a security patch for a given vulnerability j discovered at a given date. $Concentration$ is a measure of the market concentration at the date the vulnerability was discovered. We test two different measures of concentration: the number of firms competing in the market (-n) and the Herfindahl'Hirschman Index (HHI). In line with Proposition 1 which suggests a positive effect of market concentration on the security level provided by a vendor, we expect a negative sign for the parameter β_1.

Additionally, we control for a list of other factors that may have an impact on the responsiveness of the web browser publisher in releasing a security patch. First, X_j is a vector of variables accounting for the characteristics of the vulnerability, such as its severity or its type. Developers may be more or less rapid to find a secure solution according to the type of the vulnerability. On the other hand, a vendor would conceivably patch more rapidly a vulnerability that has a more severe security impact on the product. Specifically, we use two types of information: the Common Vulnerability Scoring System (CVSS) and the Common Weakness Enumeration (CWE).[11] CVSS has a value ranging from 1 to 10 that ranks a vulnerability according to the degree of threats it represents. The CWE categorizes software weaknesses into general classes.[12] In our model, CVSS values are directly used as a variable ($Vulnerability\_severity_j$), while the CWE catalog is used as a vector of dummies.

[11] CVSS is an industry standard, initially defined by the US government's National Infrastructure Advisory Council (NIAC) and maintained today by the Forum of Incident Response and Security Teams (FIRST). The CWE is created and maintained by the MITRE Corporation, which owns and maintains the Common Vulnerabilities and Exposures (CVE) system.

[12] In the case of web browser vulnerabilities, the referenced CWEs are: improper restriction of operations within the bounds of a memory buffer, the improper control of generation of code, configuration issues, information exposure, improper input validation, numeric errors, security features, improper limitation of a pathname to a restricted directory, permissions, privileges, and access controls, concurrent execution using shared resource with Improper synchronization, ressource management errors, and the incorrect type conversion or cast. Source: MITRE

Secondly, we control for the characteristics of the web browser and its publisher by X_i. We include the $Software\_age_i$, and either a dummy variable for whether the web browser has an open source core component ($Open\_source_i$) or dummy variables assigned to each web browser publisher.

The $Software\_age_i$ variable accounts for the difficulties to fix a flaw due to the code quality of the affected version. Indeed, according to practitioners, the quickness to find the root cause of a flaw depends considerably on the quality of the code, which generally depends on how old are the software and the development tools the original programmers have used.[13] The $Open\_source_i$ variable is a dummy variable which is equal to 1 if at least the rendering engine of the affected web browser has an open source license, and to 0 otherwise. A stream of work argues that open source providers offer better quality and are more responsive to customer needs (Arora et al., 2010b; Lerner and Tirole, 2002). We thus expect $Open\_source_i$ to reduce the $Patching\_time_{ij}$.

In alternative regressions, we account for the vendor-specific characteristics by a vector of dummies. Indeed, vendors may have specific patch release policies that affect their patch release decisions, but are unobserved by us. Some of them may care more about their reputation because of the spillover effect on their other products. The patching time can also depend on their general financial ability. In Arora et al. (2010b), this gap between vendors is controlled by the firm's size. In our case, web browser publishers present heterogeneous forms of organization and business models, from open source foundations to multinationals listed on the stock exchange. The size of a firm is hence not an adequate information to account for the vendor specific unobserved factors and we preferred to attribute a dummy variable to each vendor.

Third, the $Disclosure_j$ variable captures the impact of vulnerability information disclosure on the responsiveness of vendors. It is a dummy variable taking the value 1 if the disclosure is in agreement with the conventional duration stated by the disclosure policy of each vulnerability search programs from where our data set comes.[14] We attribute the value 0 if the vulnerability is disclosed to public later than the duration stated by the disclosure policy. Indeed, a recent stream of literature in information security economics deals with the impact of information disclosure in security provision (Nizovtsev and Thursby, 2007; Arora, Telang, and Xu, 2008; Arora et al., 2010b), supporting theoretically and empirically the idea that information disclosure helps vendors to be more responsive.

$Time\_effect$ captures a potential time trend, such as the growing awareness both among consumers and practitioners about security issues, which in turn, makes editors more responsive in patching

[13] Source: interview with Nicolas Ruff, security engineer at Google Security.

[14] Vulnerability research organisms generally apply a policy of *responsible disclosure*, notifying the affected vendor and keeping the vulnerability information confidential during a period of time. ZDI keeps the vulnerability information confidential for 4 months, iDefense for 6 months and Google Project Zero for 3 months.

vulnerabilities. For that, we use the date in which the vulnerability is discovered. Lastly, ϵ represents the unobservable error term.

Next, to estimate whether the effect of market concentration is identical when the affected firm highly dominates the market, we specify a variant model as follow:

$$Patching\_time_{ij} = \beta_0 + \beta_{1a} Concentration + \beta_{1b} Big\_mshare_i + \beta_{1c} Concentration \cdot Big\_mshare_i + \beta_2 X_j + \beta_3 X_i + \beta_4 Disclosure_j + \beta_5 Time\_effect + \epsilon \quad (2)$$

In this equation, we added in the baseline model (Equation 1) the Big_mshare_i variable and an interaction term of this variable with *Concentration*. Big_mshare_i is a dummy variable equal to 1 if the affected web browser publisher's market share is greater than a certain percentage – values of 40%, 50% are tested – of the market at the moment the vulnerability is discovered. It captures the effect for a firm to have a relatively large installed base compared to its competitors at the time it is informed about the security failure. The coefficient of the interaction term β_{1c} represents the difference in the effect of market concentration between the case the affected firm "highly dominates" the market or does not. We expect β_{1b} to be positive according to Proposition 2 and β_{1c} to be positive according to Proposition 3.

Our models are estimated with ordinary least squares (OLS) and negative binomial (NB) regressions.

6 Data and method

For the purpose of our empirical analysis, we have constructed a 10-year pooled cross-sectional data set consisting of web browser vulnerabilities identified and patched from January 2007 to December 2016. We combined data from a variety of sources: (1) vulnerability information collected from vulnerability research programs, (2) patch and update release dates of each web browser from publishers' websites, (3) quarterly market shares of each web browser from Statcounter.com, , and (4) additional vulnerability information from the National Vulnerability Database (NVD).

Our data come mainly from private vulnerability research programs, namely Tipping Point's Zero Day Initiative (ZDI), iDefense's Vulnerability Contributor Program (iDefense), and Google Project Zero. ZDI is a program run by the security firm Tipping Point since 2005. iDefense is the corresponding program of Verisign initiated in 2003. As mentioned in Section 5, ZDI and iDefense are the two main players in the commercial vulnerability market, which financially reward security researchers in

exchange for information they have about security flaws that were unknown before. Both ZDI and iDefense notify the affected vendors and communicate with them until they release a patch. Information about the vulnerability is kept confidential during a period of time specified by their disclosure policy. The disclosure timing is not strictly applied, but the actual duration of confidentiality is revealed on the program's website after the security patch is released. Similar to these programs, Google Project Zero is a project initiated by Google in 2014, which focuses on vulnerabilities that affect directly or indirectly Google's own products, meaning that a flaw affecting Internet Explorer can be treated by the project as much as a vulnerability in Google Chrome. The program also discloses the vulnerability information once a patch has been released or after a 90-day-deadline.

From these three programs, we collected the information related to the web browser vulnerabilities they identified. We consider only vulnerabilities specifically assigned to web browser publishers, i.e., that can be fixed by the web browser publishers themselves without a third-party intervention. For instance, Adobe Flash vulnerabilities are not taken into account since patches are mainly developed by Adobe and not by the web browser publishers themselves, although the security of web browsers is impacted by these vulnerabilities.

In our analysis, one observation consists in a pair of vulnerability (j) and the associated web browser publisher (i). For vulnerabilities that affect more than one web browser, we have duplicated the vulnerability information in several observations, each associated to a distinct publisher, in order to assess each publisher's strategy separately.

The time a software vendor spends in patching a vulnerability – the $Patching\_time$, our dependent variable – corresponds to the duration (in number of days) between the date at which the vulnerability is reported to the vendor and the the delivery date of the associated patch. The patch release date is collected from release notes of the web browser's official website. The notification date of the vulnerability to the publisher is collected from the vulnerability research programs. These programs are actually some of the few sources that provide publicly such information, given its highly confidential nature especially from the software vendor's point of view.

For each vulnerability-browser pair, we collect from Statcounter.com the market share of the affected web browser at the date the vulnerability was reported to the publisher. We determine the number of firms in the market and the *Herfindahl'Hirschman Index* (HHI) in each quarter from January 2007 to December 2016 using these data.[15] Market shares and market concentration measures are used as our

[15]In order to determine the number of firms in the market, we count a browser publisher as "present" in the market if it has more than 0.5% of the market at the time the vulnerability is discovered. Thus, the number of firms is the number of browsers that have more than 0.5% of market share at the date the vulnerability is discovered.

main explanatory variables.[16]

This database has been completed with information about vulnerability characteristics, such as the severity (CVSS) and the type of the vulnerability (CWE) from the National Vulnerability Database (NVD).[17] As an important part of our data - such as the characteristics of a vulnerability - is collected from NVD, we only retained NVD-listed vulnerabilities in our regressions.

The public disclosure date of vulnerability information is determined through a variety of sources, including the date indicated by the vulnerability research program and the release date on NVD. Lastly, from each web browser's website, we collected information about each version in order to determine how old the affected version of the web browser is ($Software\_age$ variable).

6.1 Descriptive statistics

Our analysis is based on 586 observations consisting in web browser vulnerabilities identified from January 2007 to December 2016 by the three vulnerability research programs mentioned earlier. Table 1 reports summary statistics comparing our data set to the population of web browser vulnerabilities referenced on NVD. The description of variables and summary statistics are reported in Table 6 and Table 7 of the Appendix.

Our data set represents 15.5% of the total web browser vulnerabilities referenced on NVD during the observation period, in a balanced manner over time (Figure 4, left). Nonetheless, we note that high severity (CVSS of 9 to 10) vulnerabilities are over-represented in our data compared to the parent population. We explain this bias by the commercial nature of the programs from where our data come (Figure 4, right). This bias is controlled in our regressions by the $Vulnerability\_severity$ variable.

6.2 Method

The dependent variable $Patching\_time$ is a positive integer as the time spent by the vendor to release a patch is measured in number of days. Although it is a count variable, we note that it takes a wide range of values from 0 to 302 and the mean and the median are distant from 0. We thus consider that both OLS and count models are suitable to estimate our models.[18] A Poisson regression would be a

[16] Market shares are not used as is. It is used in defining the dummy Big_mshare, which identifies whether the publisher of the affected web browser should be considered as "highly dominant". As mentioned in Subsection 5.2, Big_mshare is equal to 1 when the affected web browser's market share exceeds a certain amount; values of 40% and 50% are tested.

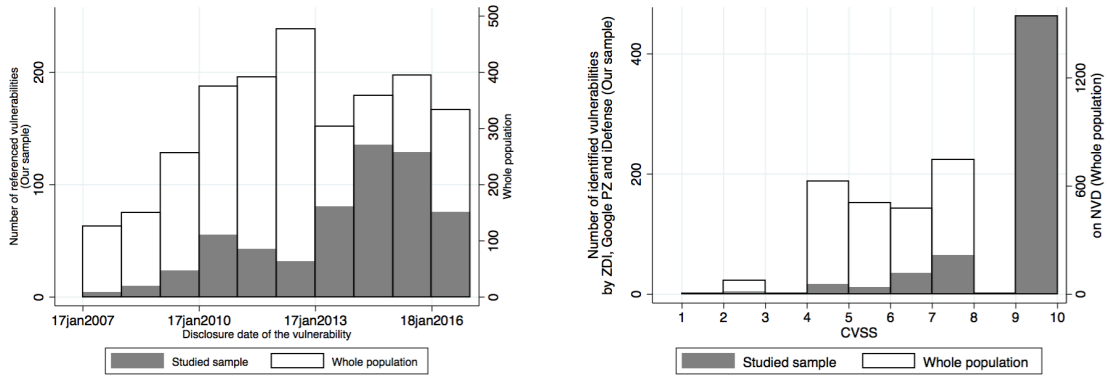
[17] The Common Vulnerability Scoring System (CVSS) is a scoring system that aims at prioritizing the vulnerabilities according to threats they represent. Scores are calculated based on a formula that depends on several metrics that approximate ease of exploit and the impact of exploit. They range from 0 to 10, with 10 being the most severe. The Common Weakness Enumeration (CWE) is a categorization of software weaknesses. The dictionary is maintained by the MITRE Corporation.

[18] When the mean of the outcome variable is relatively high (often defined as greater than 10 as a rule of thumb), an OLS regression can typically be applied to a count outcome with minimal difficulty.

Table 1: Comparative statistics between the studied data set and the NVD-listed vulnerabilities

Variable	Web browser vulnerabilities listed in NVD	Our data
Number of vulnerabilities affecting...		
Apple Safari	550	63
Google Chrome	1148	32
Microsoft Internet Explorer	1086	441
Mozilla Firefox	898	60
Other web browsers	81	0
Total number of vulnerabilities	3783	586
Mean of $Vulnerability\_severity$(CVSS)	7.25	8.75

Figure 4: Comparison of the studied data set to the whole population of NVD-listed vulnerabilities



baseline model for count data. However, given the presence of significant over-dispersion in our data set, with standard deviation superior to the mean, we use negative binomial to estimate the models.

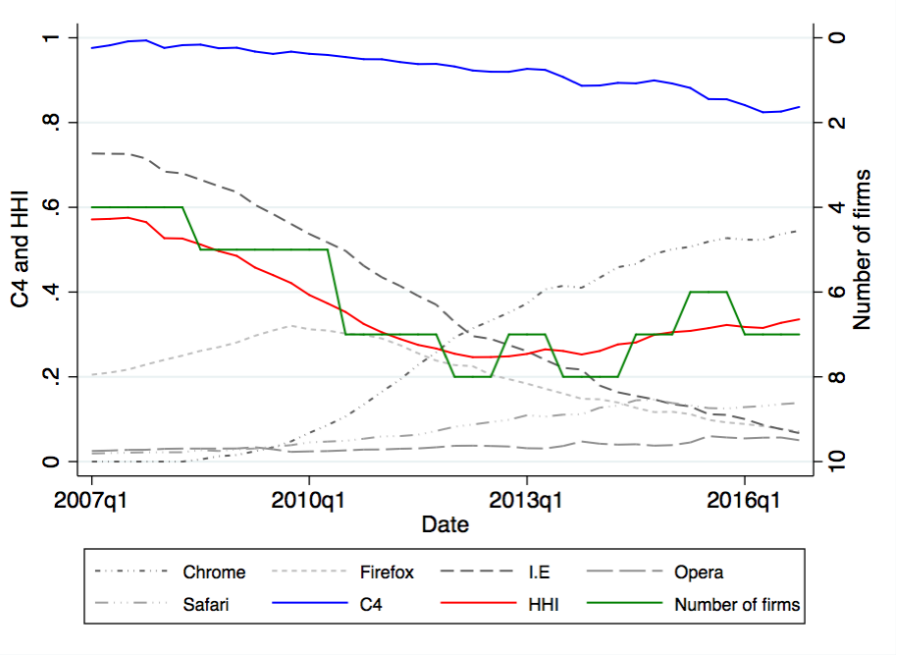
Another option would have been to use a survival model. Although it is clearly possible to use a proportional hazard model here, we do not employ it because our data set is only composed of vulnerabilities which have been patched. For this reason, we do not obtain any additional information by using a survival model.

Table 2: Correlation matrix of market concentration measures

Measures	$-n$	C_1	C_4	HHI	$qyear$
$-n$	1.00				
C_1	0.75	1.00			
C_4	0.39	0.01	1.00		
HHI	0.86	0.88	0.43	1.00	
$qyear$	0.45	0.15	0.95	0.57	1.00

For the main explanatory variable *Concentration*, we use two different measures: the number of firms competing in the market and the HHI. Figure 5 shows the evolution of market concentration in the web browser market according to different measures. The four-firm concentration ratio (C_4) is also widely used in the economic literature, but rather because the HHI requires the complete distribution

Figure 5: Evolution of market concentration in the web browser market



of market shares, which is difficult to obtain in most cases. We also note that some dynamic aspects of the web browser market such as the change in the dominant firm or the existence of web browsers with small market shares are not well reflected in the C_4 compared to the number of firms or the HHI.[19] Moreover, C_4 is highly correlated to the $Time\_effect$ variable (see Table 2). We thus consider that it is not a better measure of concentration in our model.

In model (2), the main explanatory variable Big_mshare is a dummy variable equal to 1 if the affected web browser publisher's market share is greater than a certain percentage – 40%, 50% – of the market at the moment the vulnerability is discovered. We limit estimations on these two values in order to avoid correlation problems with the $Concentration$ variable. Lastly, we do not test the model with the HHI because of correlation problem with Big_mshare: we use the number of firms as the only concentration measure.

7 Estimation results

In this section, we report the estimation results for the econometric models. We first show results for the baseline model (1), which estimates the effect of market concentration on the patch release time without accounting for the market share of the affected web browser. We next estimate model (2), in which we account both for the number of firms competing in the market and for whether the web

[19] More than a hundred web browsers have entered the market since the creation of the first web browser in 1990

browser is "highly dominant" at the time the security flaw is discovered.

7.1 Using the number of firms and the HHI as a concentration measure

To begin with, we report in Table 3 the regression results for model (1), which estimates the effect of market concentration on the patch release time. In each pair of columns, we report both OLS and NB regression results. We use two different measures of concentration – the number of firms and the HHI. Consistent with our theoretical model, the coefficients (for OLS) and the average marginal effects (for NB) of *Concentration* are negative and statistically significant for each regression. A web browser publisher releases a security patch about 5 days earlier when one less firm competes in the market.

Regarding the other variables, the impact of vulnerability severity score ($Vulnerability\_severity$) is negative and statistically significant, meaning that editors are likely to respond faster to more severe vulnerabilities. Besides, by regressing the $Vulnerability\_severity$ by $Vulnerability\_type$ dummies, we note that there is a strong relationship between the type of the vulnerability and the severity score (see Table 11 in Appendix). The characteristics of the vulnerability (its severity, its type) may influence the patching time not only because it has an impact on the intent of the vendor (prioritization according to the severity, impact on its reputation), but also because of technical reasons (difficulties to develop the patch for some types of vulnerability, learning effects on types that are frequently identified...). Although it is interesting to note it, our model does not study further the relationship between vulnerability characteristics and the patch release time as we focus specifically on the impact of the competition intensity. Not controlling for the vulnerability's characteristics does not qualitatively affect our main results (Table 10 in Appendix).

As to variables accounting for software characteristics, $Software\_age$ has a positive coefficient, meaning that patches are released slower when the software version is older. This is an expected result as the older the version of the software is, the less incentives the vendor has to invest in security. Moreover, the older the version is, the more likely it is that the code is longer and complex; therefore, it takes more time to identify and fix the flaw. As to vendor-specific characteristics, we find that Mozilla Firefox vulnerabilities are patched on average 25 days faster than the vulnerabilities of other web browsers. The estimation results using the *Open_source* variable are reported in Appendix as a robustness check (Table 9). We find that *Open_source* is highly significant and negative, confirming the idea suggested by the literature about the benefits of open source projects. That is, for a web browser, having an open source rendering engine reduces the patching time by more than 15 days.

Lastly, the *Disclosure* variable is statistically significant and negative, meaning that everything

Table 3: Results for model (1) using -n and HHI as concentration measures

Using as main explanatory variable (*Concentration*):	-n		HHI	
	OLS	NB	OLS	NB
Concentration	-5.483**	-4.794**	-85.35**	-114.3***
	(2.422)	(2.314)	(42.14)	(42.00)
Vulnerability specific effects				
$Vulnerability\_severity$	-5.210**	-5.308**	-5.704***	-6.219**
	(2.050)	(2.567)	(2.188)	(2.593)
$Vulnerability\_type$ dummies				
*cwe*119 (Improper Restriction of Operations [...])	-6.874	-3.904	-7.152	-3.399
	(10.65)	(10.50)	(10.63)	(10.46)
*cwe*17 (Source code)	-26.67	-34.90	-26.70	-33.33
	(29.80)	(25.06)	(29.60)	(25.60)
*cwe*94 (Improper control of generation of code)	20.46	22.53	18.83	21.32
	(14.84)	(14.97)	(14.68)	(14.78)
*cwe*200 (Information exposure)	-21.65	-16.96	-24.92*	-19.24
	(14.22)	(17.10)	(14.57)	(16.57)
*cwe*20 (Improper input validation)	-7.870	0.0355	-8.294	-0.566
	(12.54)	(13.71)	(12.52)	(13.60)
*cwe*189 (Numeric errors)	-22.46	-21.90	-22.23	-19.56
	(15.64)	(14.52)	(15.46)	(14.91)
*cwe*254 (Security features)	-10.81	-7.676	-10.64	-6.901
	(13.67)	(47.85)	(13.64)	(48.13)
*cwe*22 (Improper limitation of a pathname [...])	-68.12***	-98.82***	-66.44***	-98.80***
	(14.24)	(2.854)	(14.08)	(2.874)
*cwe*264 (Permissions, privileges, and access controls)	-18.10	-18.25	-19.77	-19.29
	(14.02)	(13.22)	(14.17)	(12.99)
*cwe*362 (Concurrent execution [...])	-28.56	-39.82*	-31.88*	-41.73*
	(20.29)	(23.71)	(18.84)	(22.86)
*cwe*399 (Ressource management errors)	8.639	15.90	7.956	16.68
	(11.79)	(11.77)	(11.74)	(11.79)
*cwe*704 (Incorrect type conversion or cast)	-8.606	-8.168	-7.544	-5.299
	(11.15)	(46.92)	(11.13)	(48.28)
Soft. and vendor specific effects				
$Software\_age$	0.755*	0.795	0.789*	0.798
	(0.438)	(0.530)	(0.442)	(0.527)
apple	-10.11	-13.92**	-11.54*	-14.95**
	(6.888)	(6.838)	(6.884)	(6.696)
google	-23.21*	-31.13***	-23.46*	-32.61***
	(12.16)	(7.648)	(12.01)	(7.526)
mozilla	-23.98***	-26.37***	-24.65***	-27.78***
	(8.303)	(6.776)	(8.172)	(6.645)
Disclosure effect				
Disclosure	-48.64***	-48.37***	-48.99***	-49.04***
	(3.611)	(4.462)	(3.631)	(4.474)
Time effect				
qyear	-1.513***	-1.513***	-1.605***	-2.276***
	(0.293)	(0.293)	(0.316)	(0.659)
Constant	458.7***		549.1***	
	(71.85)		(90.34)	
Observations	586	586	586	586
R-squared	0.388		0.386	
Wald chi-squared		236.43		239.51
VIF for *Concentration*	1.40		1.68	

Robust standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1
For OLS regressions, coefficients are reported. For NB regressions, average marginal effects are reported.

else equal, applying a "responsible disclosure policy" shortens the patch release time. More precisely, disclosing the vulnerability information publicly after a fixed period of time – as stated by each vulnerability research program – shortens the patch release time by around 50 days.

To sum up, from this baseline model, we find that market concentration is beneficial to improve web browser publishers responsiveness in patching vulnerabilities. This result is in line with Proposition 1 of the theoretical model, which states that in an oligopolistic market where firms compete in security quality, the security level provided by a firm decreases as the number of competitors increases ($\delta s_i^*/\delta n < 0$).

7.2 The effect of being a dominant web browser

Table 4: Results for model (2), effect of *Big_mshare*

Using as *Concentration*:	*-n*			
Big_mshare = 1 when the affected web browser's market share is:	≥ 0.40		≥ 0.50	
	OLS (coef.)	NB (IRR)	OLS (coef.)	NB (IRR)
Concentration	-5.361**	0.932**	-5.850**	0.914***
	(2.552)	(0.0265)	(2.602)	(0.0260)
Big_mshare	42.45	5.939***	34.01	22.02***
	(34.78)	(2.367)	(51.62)	(12.72)
Big_mshare × *Concentration*	8.942	1.298***	6.998	1.738***
	(5.447)	(0.0847)	(9.489)	(0.190)
Vulnerability specific variables	Yes			
Soft. and vendor specific variables	Yes			
Disclosure effect variables	Yes			
Time effect variables	Yes			
Observations	586	586	586	586
R-squared	0.394		0.389	
Wald chi-squared		241.40		246.04

Robust standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1
For OLS regressions coefficients are reported. For NB regressions IRR are reported.
Results with coefficients of control variables is available in Appendix.

Table 4 reports estimations for model (2), which focuses on the change in the responsiveness of a publisher when it highly dominates the market. In each pair of columns, we report results for the two threshold values of market share for the *Big_mshare* dummy. That is, *Big_mshare* is equal to 1 when the affected web browser's market share is more than 40% of the market in the two first columns, and 50% in the two other columns. As in Table 3, we report both OLS and NB regression results.

First of all, the estimation results for model (2) confirms the result obtained in the model (1): we obtain qualitatively and quantitatively similar results for the *Concentration* variable for the two models. Indeed, similar to the estimation from model (1), a web browser publisher releases a security patch for about 5 days more quickly when there is one less competitor in the market.

Nevertheless, this "positive" effect of market concentration is reduced when the affected web browser is "highly dominant" at the time the vulnerability is discovered: a publisher is less responsive in releasing security patches when its web browser has a significant market share. Moreover, the positive effect of having less competitors on the publisher's responsiveness is reduced when the web browser dominates the market.

First, either both coefficients of Big_mshare and $Big\_mshare \times Concentration$ are positive or their incident rate ratios (IRR) are larger than 1, meaning that a firm releases security patches more slowly when it is highly dominant. Moreover, comparing the magnitude of the coefficients in the case of NB estimations – which estimates are significant both for Big_mshare and for the interaction term –, the higher the threshold market share, the larger the coefficients higher is the IRR. In other words, the higher the market share of the publisher, the larger is the negative effect on the patch release time. This result corroborates Proposition (2) which suggests that the security quality chosen by a highly dominant firm decreases with respect to its market share ($\partial s_i^*/\partial m_i < 0$).

Second, the interaction term $Big\_mshare \times Concentration$ shows also a negative effect on the speed of patch release. That is, the positive effect of market concentration on reducing the patching time is weakened when the affected publisher has a significant market share. Moreover, this negative effect is strengthened with a higher threshold of market share for the Big_mshare dummy. For instance, the positive effect of market concentration is likely to be reduced by 44% ($1.738 - 1.298 = 0.44$) when firms have more than 50% of the market compared to when they have more than 40% of the market. Besides, we note that only estimates of negative binomial regressions are significant for the interaction term. These results support the theoretical result suggested in Proposition (3) that the positive effect of market concentration on the security level a firm provides is reduced when it highly dominates the market ($\partial^2 s_i^*/\partial n \partial m_i < 0$).

Lastly, to check the robustness of our main findings, additional estimations are carried out. First, we exclude the vendor dummies and only include *open_source* dummy and $software\_age$ as software and vendor specific characteristics. The estimation results for our main explanatory variables are qualitatively similar (see Table 8 in Appendix). Secondly, controlling for the data source of vulnerability

reports does not affect the results (see Table 9 in Appendix).[20] Also, not controlling for the vulnerability and software specific effects does not qualitatively affect our main results (see Table 10 in Appendix).

8 Conclusion

Our objective in this paper was to examine the impact of competition intensity on software vendor's security investment behavior. To answer empirically to this question, we took the case of the web browser market. Using a dataset of 586 web browser vulnerabilities identified and patched over a period of ten years, we examined the effect of market concentration and a significant market share of the web browser affected by the weakness, on publisher's promptness to release a security patch.

We find that market concentration is not necessarily harmful to security provision: less competition makes a web browser publisher more responsive in delivering security patches, although this effect is reduced when the publisher is highly dominant in the market. In our theoretical model, we explains this result by a particularity of the market we study: because companies compete in quality, they have more incentives to invest in quality when competition is less intense. Nevertheless, because security is costly and does not lead to market expansion, a dominant position in the market does not encourage the vendor to invest in security.

Academics and practitioners together have largely insisted on the danger of software monoculture. Our findings give an additional insight on this issue, suggesting that it is important as well to take account the vendor's actual incentives to invest in security, which is affected by the degree of competition in the market.

Furthermore, in line with some practitioners' opinion, we find that diverse externalities such as the severity of the vulnerability, information disclosure or open source licensing also significantly impact the vendor's incentives to release a patch more quickly.

References

F. Allen. Reputation and product quality. *The RAND Journal of Economics*, pages 311–327, 1984.

C. Argenton and J. Prüfer. Search engine competition with network externalities. *Journal of Competition Law and Economics*, 8(1):73–105, 2012.

A. Arora, J. P. Caulkins, and R. Telang. Research note-sell first, fix later: Impact of patching on software quality. *Management Science*, 52(3):465–471, 2006.

[20] Our observations are gathered from three different vulnerability research programs

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

A. Arora, C. Forman, A. Nandkumar, and R. Telang. Competition and patching of security vulnerabilities: An empirical analysis. *Information Economics and Policy*, 2010a.

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*, 2010b.

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.

Y. Beres, J. Griffin, S. Shiu, M. Heitman, D. Markle, and P. Ventura. Analysing the performance of security solutions to reduce vulnerability exposure window. pages 33–42, 2008.

K. Campbell, L. A. Gordon, M. P. Loeb, and L. Zhou. The economic cost of publicly announced information security breaches: empirical evidence from the stock market. *Journal of Computer Security*, 11(3):431–448, 2003.

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. Cohen and M. J. Mazzeo. Competition, product differentiation and quality provision: an empirical equilibrium analysis of bank branching decisions. 2004.

S. Domberger and A. Sherr. The impact of competition on pricing and quality of legal services. *International Review of Law and Economics*, 9(1):41–56, 1989.

S. Frei, M. May, U. Fiedler, and B. Plattner. Large-scale vulnerability analysis. pages 131–138, 2006.

E. Gal-Or and A. Ghose. The economic incentives for sharing security information. *Information Systems Research*, 16(2):186–208, 2005.

D. Geer, R. Bace, P. Gutmann, P. Metzger, C. P. Pfleeger, J. S. Quarterman, and B. Schneier. Cyberinsecurity: The cost of monopoly. *Computer and Communications Industry Association (CCIA)*, 2003.

L. A. Gordon, M. P. Loeb, W. Lucyshyn, and T. Sohail. The impact of the sarbanes-oxley act on the corporate disclosures of information security activities. *Journal of Accounting and Public Policy*, 25 (5):503–530, 2006.

C. Ioannidis, D. Pym, and J. Williams. Information security trade-offs and optimal patching policies. *European Journal of Operational Research*, 216(2):434–444, 2012.

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.

J. Lerner and J. Tirole. Some simple economics of open source. *The journal of industrial economics*, 50(2):197–234, 2002.

M. J. Mazzeo. Product choice and oligopoly market structure. *The RAND Journal of Economics*, pages 221–242, 2002.

M. J. Mazzeo. Competition and service quality in the us airline industry. *Review of industrial Organization*, 22(4):275–296, 2003.

G. McGraw and C. CTO. Exploiting software: How to break code. In *Invited Talk, Usenix Security Symposium, San Diego*, 2004.

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

S. Ransbotham, S. Mitra, and J. Ramsey. Are markets for vulnerabilities effective? *ICIS 2008 Proceedings*, page 24, 2008.

U. Ronnen. Minimum quality standards, fixed costs, and competition. *The RAND Journal of economics*, pages 490–504, 1991.

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.

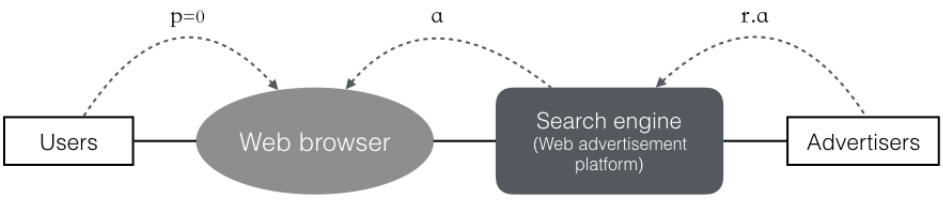
D. Waterman. Diversity and quality of information products in a monopolistically competitive industry. *Information Economics and Policy*, 4(4):291–303, 1990.

Appendix

Table 5: Comparison of the four most popular web browsers

Browser	Publisher	Rendering engine	License	Revenue model
Chrome	Google	Blink (fork of Webkit)	Proprietary software with open source rendering engine (GNU LPGL). An open source version of the browser is available (Chromium)	90% of ABC/Google's revenue come from search related ad.
Firefox	Mozilla	Gecko	Open source (MPL)	Built-in search engine royalties (> 90% of whole revenues, ≃$100m) and donations. Exclusive contract with Google until 2014 and with Yahoo Search since 2015
Internet Explorer	Microsoft	Trident and EdgeHTML since 2015	Proprietary	Revenues from other activities
Safari	Apple	Webkit	Proprietary software with open source rendering engine (GNU LPGL)	$1bn of built-in search engine royalties from Google (in 2014)

Figure 6: Today's web browser's revenue model



Users do not pay for the web browser thus $p = 0$. The web browser's owner derives revenue from the search engine (SE), which in turn obtains revenues from advertisers that pay for user traffic. Denoting by a the revenue the web browser publisher obtains per user, the SE earns $r \cdot a$ from advertisers, where $r > 1$.

Table 6: Description of variables

Variable	Description
Patching_time	Time spent by the editor to release a patch (unit: number of days)
-n	n represents the number of firms in the market at the time the vulnerability is discovered.
HHI	The Herfindahl-Hirschman Index at the vulnerability discovery date
Big_mshare	Equal to 1 if the affected web browser publisher's market share at the moment the vulnerability is discovered is greater than 50% (40%) of the market, 0 otherwise
$Vulnerability\_severity$	Vulnerability severity score (CVSS base score from 1 to 10)
$Vulnerabililty\_type$ dummies	Dummies for type of weakness such as 'cross site scripting', 'SQL injection', 'Information leak', etc. 12 types of weakness were associated to web browsers vulnerabilities.
$Software\_age$	Age of the software at the time when the vulnerability is discovered (unit: number of years).
Vendor dummies	Dummies associated to each web browser publisher except Microsoft
Open_source	Equal to 1 if the affected web browser's engine is open source, 0 otherwise
Disclosure	Equal to 1 if the vulnerability was publicly disclosed earlier than the conventional period of time of 90 days after being notified to the vendor, 0 otherwise.
qyear	The date at which the vulnerability was reported to the editor (unit: date in quarter).

Table 7: Summary statistics

Variable	Mean	Minimum	Maximum	Standard Deviation
Patching_time	100.2	0	302	52.6
-n	-6.8	-8	-4	1.02
HHI	0.309	0.246	0.575	0.639
Big_mshare ($m.s. \geq 0.40$)	0.13	0	1	0.33
Big_mshare ($m.s. \geq 0.50$)	0.09	0	1	0.28
$Vulnerability\_severity$	8.75	2.6	10	1.30
$Software\_age$	5.98	0	14.8	4.18
Open_source	0.26	0	1	0.44
Disclosure	0.53	0	1	0.50
qyear	2013q2	2007q1	2016q4	8.7

Number of observations: 586

Table 8: Detailed results for Table 4

Using as *Concentration*:	*-n*			
Big_mshare = 1 when the affected web browser's market share is:	≥ 0.40		≥ 0.50	
	OLS (coef.)	NB (IRR)	OLS (coef.)	NB (IRR)
Concentration	-5.361**	0.932**	-5.850**	0.914***
	(2.552)	(0.0265)	(2.602)	(0.0260)
Big_mshare	42.45	5.939***	34.01	22.02***
	(34.78)	(2.367)	(51.62)	(12.72)
Big_mshare × *Concentration*	8.942	1.298***	6.998	1.738***
	(5.447)	(0.0847)	(9.489)	(0.190)
Vulnerability specific effects				
Vulnerability_severity	-4.826**	1.069***	-4.896**	1.060**
	(2.031)	(0.0259)	(2.068)	(0.0266)
Vulnerability_type dummies				
*cwe*119	-8.377	0.896	-9.738	0.950
	(10.63)	(0.0992)	(10.61)	(0.106)
*cwe*17	-30.40	0.468*	-24.41	0.607
	(29.43)	(0.192)	(32.64)	(0.248)
*cwe*94	19.42	1.450***	15.78	1.487***
	(14.99)	(0.189)	(14.91)	(0.196)
*cwe*200	-21.03	1.202	-20.13	1.265
	(14.16)	(0.252)	(14.84)	(0.269)
*cwe*20	-9.467	1.041	-9.109	1.124
	(12.50)	(0.150)	(12.23)	(0.161)
*cwe*189	-25.05	0.745	-25.95	0.824
	(16.12)	(0.146)	(16.07)	(0.161)
*cwe*254	-10.85	1.268	-12.04	1.290
	(13.71)	(0.689)	(13.96)	(0.704)
*cwe*22	-73.81***	0.0127***	-53.22***	0.0213***
	(19.69)	(0.0144)	(12.55)	(0.0242)
*cwe*264	-18.69	1.114	-20.28	1.152
	(14.09)	(0.182)	(14.17)	(0.188)
*cwe*362	-24.13	0.841	-25.38	0.772
	(17.02)	(0.345)	(18.05)	(0.320)
*cwe*399	6.765	1.217*	4.015	1.316**
	(11.71)	(0.142)	(11.68)	(0.154)
*cwe*704	5.209	2.012	8.935	1.015
	(20.51)	(1.157)	(14.43)	(0.556)
Soft. and vendor specific effects				
Software_age	0.768*	1.005	0.855*	1.004
	(0.442)	(0.00571)	(0.449)	(0.00576)
apple	-9.522	1.246***	-11.71	1.263***
	(7.422)	(0.0969)	(7.720)	(0.0992)
google	-18.24	1.089	-13.06	0.910
	(17.30)	(0.140)	(14.96)	(0.104)
mozilla	-23.42**	1.006	-25.53***	1.020
	(9.179)	(0.0907)	(9.504)	(0.0930)
Disclosure effect				
Disclosure	-48.72***	0.601***	-48.27***	0.605***
	(3.557)	(0.0274)	(3.580)	(0.0277)
Time effect				
qyear	-1.451***	1.017***	-1.630***	1.017***
	(0.345)	(0.00133)	(0.392)	(0.00135)
Constant	444.2***	0.262***	482.0***	0.265***
	(88.49)	(0.0162)	(99.09)	(0.0164)
Observations	586	586	586	586
R-squared	0.394		0.389	
Wald chi-squared		241.40		246.04

Robust standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1
For OLS regressions coefficients are reported. For NB regressions AME or IRR are reported.

Table 9: Estimation results with *open_source* variable

Using as *Concentration*:	*-n*		HHI		*-n*	
Big_mshare = 1 when the affected web browser's market share is:					≥ 0.50	
	OLS (coef.)	NB (AME)	OLS (coef.)	NB (AME)	OLS (coef.)	NB (IRR)
Concentration	-5.142**	-4.129*	-81.99*	-102.0**	-5.016*	0.935**
	(2.476)	(2.293)	(43.28)	(41.79)	(2.668)	(0.0256)
Big_mshare					33.36	23.13***
					(46.37)	(12.86)
Big_mshare × *Concentration*					7.092	1.757***
					(7.722)	(0.180)
Vulnerability specific effects						
Vulnerability_severity	-5.568***	-5.239**	-6.049***	-6.086**	-5.333**	1.066***
	(2.126)	(2.520)	(2.279)	(2.549)	(2.162)	(0.0248)
Vulnerability_type dummies						
*cwe*119	-5.928	-2.201	-6.264	-1.533	-8.083	0.906
	(10.09)	(10.51)	(10.07)	(10.48)	(10.47)	(0.1000)
*cwe*17	-32.93	-41.59*	-32.54	-40.54*	-36.14	0.447**
	(30.11)	(22.23)	(30.02)	(22.57)	(30.46)	(0.178)
*cwe*94	19.77	22.65	18.26	21.75	18.58	1.457***
	(14.74)	(15.01)	(14.56)	(14.84)	(14.96)	(0.190)
*cwe*200	-24.02*	-18.19	-27.04*	-20.27	-23.88*	1.182
	(14.10)	(16.91)	(14.45)	(16.42)	(14.12)	(0.249)
*cwe*20	-7.878	0.220	-8.327	-0.200	-9.969	1.042
	(12.23)	(13.76)	(12.24)	(13.68)	(12.55)	(0.149)
*cwe*189	-25.71*	-24.61*	-25.28*	-22.49	-28.69*	0.696*
	(15.31)	(13.88)	(15.19)	(14.23)	(15.63)	(0.135)
*cwe*254	-10.85	-4.694	-10.81	-3.863	-12.39	1.275
	(13.59)	(49.48)	(13.56)	(49.79)	(13.88)	(0.694)
*cwe*22	-73.19***	-99.01***	-71.13***	-98.99***	-76.19***	0.0123***
	(11.68)	(2.743)	(11.42)	(2.751)	(12.70)	(0.0139)
*cwe*264	-19.05	-16.81	-20.69	-17.66	-20.99	1.097
	(13.90)	(13.29)	(14.06)	(13.08)	(14.27)	(0.178)
*cwe*362	-35.75*	-44.02**	-38.47**	-45.74**	-30.45**	0.751
	(18.87)	(21.89)	(17.46)	(21.13)	(14.42)	(0.305)
*cwe*399	8.023	15.30	7.415	16.18	5.712	1.209
	(11.42)	(11.78)	(11.37)	(11.80)	(11.80)	(0.140)
*cwe*704	8.962	18.33	10.84	23.47	21.74**	1.817
	(11.10)	(60.61)	(10.85)	(63.14)	(9.682)	(1.022)
Soft. and vendor specific effects						
Software_age	0.874*	1.022**	0.895**	1.030**	0.859*	1.007
	(0.447)	(0.520)	(0.451)	(0.518)	(0.455)	(0.00559)
Open_source	-17.53***	-22.46***	-18.40***	-23.64***	-16.10**	1.140**
	(5.352)	(5.327)	(5.295)	(5.253)	(6.466)	(0.0688)
Disclosure effect						
Disclosure	-49.59***	-49.56***	-49.87***	-50.19***	-49.64***	0.593***
	(3.563)	(4.449)	(3.580)	(4.465)	(3.545)	(0.0268)
Time effect						
qyear	-1.544***	-1.410***	-1.644***	-1.649***	-1.480***	1.017***
	(0.283)	(0.348)	(0.305)	(0.375)	(0.339)	(0.00134)
Constant	470.7***		557.3***		457.6***	0.264***
	(70.56)		(89.99)		(87.49)	(0.0163)
Observations	586	586	586	586	586	586
R-squared	0.384		0.383		0.386	
Wald chi-squared		232.37		235.06		239.15

Robust standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1
For OLS regressions coefficients are reported. For NB regressions IRR are reported.

Table 10: Estimation results without vulnerability and software/vendor characteristics control variables

Using as *Concentration*:	*-n*		HHI		*-n*	
Big_mshare = 1 when the affected web browser's market share is:					≥ 0.50	
	OLS (coef.)	NB (AME)	OLS (coef.)	NB (AME)	OLS (coef.)	NB (IRR)
Concentration	-5.039**	-5.475**	-74.94*	-106.0**	-7.138***	0.931***
	(2.443)	(2.339)	(40.02)	(41.33)	(2.664)	(0.0252)
Big_mshare					122.5***	54.18***
					(35.67)	(28.31)
Big_mshare × *Concentration*					20.72***	1.974***
					(5.760)	(0.192)
Disclosure effect						
Disclosure	-53.84***	-53.88***	-54.19***	-54.73***	-51.41***	0.621***
	(3.588)	(4.596)	(3.560)	(4.629)	(3.572)	(0.0299)
Time effect						
qyear	-1.536***	-1.504***	-1.569***	-1.640***	-1.250***	1.020***
	(0.248)	(0.279)	(0.264)	(0.301)	(0.266)	(0.000914)
Constant	422.6***		487.2***		344.6***	0.320***
	(46.77)		(65.14)		(55.01)	(0.0193)
Observations	586	586	586	586	586	586
R-squared	0.325		0.323		0.336	
Wald chi-squared		167.69		168.76		176.04

Robust standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1
For OLS regressions coefficients are reported. For NB regressions AME or IRR are reported.

Table 11: Regression of $Vulnerability\_severity$ by $Vulnerability\_type$ dummies

dependent var: $Vulnerability\_severity$	
$cwe119$	0.920***
	(0.160)
$cwe17$	-0.887
	(0.648)
$cwe94$	1.214***
	(0.194)
$cwe200$	-3.776***
	(0.314)
$cwe20$	0.585**
	(0.228)
$cwe189$	1.184***
	(0.302)
$cwe254$	-3.737***
	(0.904)
$cwe22$	-0.537
	(0.904)
$cwe264$	-2.324***
	(0.243)
$cwe362$	-2.987***
	(0.648)
$cwe399$	1.169***
	(0.167)
$cwe704$	-1.237
	(0.904)
Constant	8.037***
	(0.151)
Observations	586
R-squared	0.541

Standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1