Economics of information security and the market for software vulnerabilities

Arrah-Marie Jo

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2019IPPAT003

Thèse de doctorat

Essays on the economics of information security

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

École doctorale n°626 ED Institut Polytechnique de Paris (IP Paris)
Spécialité de doctorat : Economie

Thèse présentée et soutenue à Ville de soutenance, le le 25 Novembre 2019, par

ARRAH-MARIE JO

Composition du Jury :

Maya BACACHE-BEAUVALLET Professeur, Institut Polytechnique de Paris, Télécom Paris	Président
Thierry BURGER-HELMCHEN Professeur, Université de Strasbourg	Rapporteur
Thierry PENARD Professeur, Université de Rennes	Rapporteur
Rainer BOHME Professeur, University of Innsbruck	Examinateur
Thierry RAYNA Professeur, Institut Polytechnique de Paris, École Polytechnique	Examinateur
Marc BOURREAU Professeur, Institut Polytechnique de Paris, Télécom Paris	Directeur de thèse

Acknowledgement

I started my PhD without actually realizing what I was undertaking. I was getting out of four years of Big4's consulting life and so eager to have an intellectually more consistent experience, imagining that from now on, I would be able to contribute to a better society. I was charmed by the power of economic analysis and the discipline of Industrial Organization that I had discovered through the Master IREN's courses. I was so happy to be a student again and to learn new things of interest to me.

Doing a PhD was, of course, much harder and more serious than I expected. But most of all, it introduced me to an exciting profession: becoming a researcher. I feel so thankful to all the people who accompanied me through this learning process and who did not hesitate to play their part in it to orientate me on the right way. This section gives me the chance to express my gratitude to those who advised, listened to, and supported me throughout this period. I hope I can bring equal support to others that come after me and help them go through this fascinating journey.

First and foremost, I owe a great debt of gratitude to Professor Marc Bourreau, for having accepted to supervise my thesis. I thank him for his intellectual, moral, and financial support throughout these four years. I was fortunate to have a director who was always available, patient, and rigorous. He gave one of the most comfortable working environment for me to focus on my PhD. He also provided me with excellent opportunities to conduct my work, develop and share my work, and to learn under the best conditions.

I am honored to have as the referees of my thesis Thierry Burger-Helmchen and Thierry Pénard, as well as Maya Bacache, Rainer Böhme, and Thierry Rayna as members of my thesis committee. Thank you very much for having accepted to participate in the evaluation and the presentation of my work. I am also particularly grateful to Maya and Thierry (P.), who advised me at various stages of my research. Furthermore, I am truly thankful to Rainer's career advice and feedback for several papers presented in this thesis.

I am invaluably grateful to the people who encouraged me and helped me refine my papers during various conferences, seminars, and workshops. I wish to express my gratitude especially to Grazia Cecere, Daniel Ershov, Axel Gautier, Ulrich Laitenberger,

Patrick Legros, Doh-Shin Jeon, and Christine Zulehner, who provided me with such valuable comments and feedback, even at the most preliminary stages of my works.

I wish to express my gratitude to Alexandre De Cornière for receiving me so well during my research stay at Toulouse School of Economics. I am also grateful to Eric Brousseau who accepted me to master IREN, without which I would not have started this fantastic journey. I am thankful to Laurent Gille for his encouragement and valuable advice since master IREN. Thank you to all the colleagues from the Association Francophone de Recherche en Economie du Numérique (AFREN) for their encouragement and insightful advice.

I also had the great pleasure to spend four years with my amazing colleagues Adrien, Ambre, Alexis, Angela, Constance, Elie, Enrick, François, Jean-Marc, Jordana, Lukasz, Marie-Josée, Martin, Raphaël, Uli, Vicente, and Yann. I thank them for their good mood, their kindness, and openness. My research has greatly benefited from our discussions and their recommendations.

I thank my big and lovely family-in-law for their warm support. I thank my long-time friend Marie-Hélène, who often helped me with her multilingual talent to review several parts of my papers and who encouraged me remotely.

I am eternally grateful to my parents, who gave me the desire and the strength to complete this project successfully. They always put their trust in me, encouraged me, and advised me with the right words.

Last but not least, I thank my partner, who backed me and accompanied me all the way in this thesis as it was his own.

Contents

General Introduction

0.1 The economics of information security

Almost every day, a new headline reminds us that the use of technology is fraught with opportunities and risks. The advent of the Internet and other information and communication technologies has dramatically modernized and simplified our daily life and fostered economic growth. The transformation brought about by digitalization creates new dependencies; cyberspace has become an inseparable component of our economic, social, and political activities. Along with our increasing dependence to this interconnected digital environment, systems and networks are exposed to a growing number and a wider variety of threats. This rapidly evolving landscape brings new and growing challenges to cybersecurity.

Cyber risks are today at the top of the international agenda and are becoming a key business priority for an increasing number of companies.[1] Technical progress meets only partly the needs for cybersecurity. Engineers and security experts design security systems by guessing how an opponent would do to break the system. Modern software editors adopt secure coding practices and integrate security into the foundations of the development cycles. In the meantime, practitioners and academics progressively realize that approaching digital security risks solely from a technical perspective is not sufficient.[2]

[1] In a 2014 OECD survey (http://dx.doi.org/10.1787/9789264232440-en), governments identified cybersecurity as the second-highest priority area, while according to a survey run by ESG research, 40% of companies claim that cybersecurity is the top priority driving their technology spending (See https://www.esg-global.com/2019-technology-spending)

[2] Recent reports on how to manage digital risks insist on the importance of viewing digital risk management as an economic and social rather than purely technical challenge, which incorporates aspects of economics, human psychology, and other disciplines. See for instance the 2017 OECD report on Digital Security Risk Management, this article from Deloitte https://www2.deloitte.com/au/en/pages/risk/articles/cybercrime-tech-problem.html, or this article from Harvard Business Review https://hbr.org/2017/05/why-is-cybersecurity-so-hard

In particular, a number of researchers, mostly in computer science, started advocating that many of the existing security issues could be explained more clearly using the language of microeconomics (Anderson, 2001; Gordon and Loeb, 2002; Anderson, Böhme, Clayton, and Moore, 2008). This movement started forming an interdisciplinary research area called the economics of information security. It has become a growing and fast-moving discipline since about 2000, gathering practitioners and academic researchers of various backgrounds, from security engineers and economists to lawyers and sociologists. Publications in top journals in computer science, economics and management, the growing number of active researchers, the creation of conferences dedicated to the economics of information security (e.g., Workshop of Economics and Information Security[3]) testify the interest around the subject in the academic world.

One of the main lessons that economic analysis brought to information security is that security problems often occur because incentives are wrong. One seminal example is how banks' investment in security technology in different countries depended on their liability in case of online fraud or card loss affecting their customers (Anderson, 1993, 2007). The security of a system also often depends on the effort exerted by each individual, which in turn depends on their own benefits and costs, the efforts exerted by other individuals and their interdependencies. This makes information security a public good in many ways (Varian, 2004). In sum, a significant part of this literature builds on the analysis of externalities each agent suffers from and how to realign their incentives. Various questions are addressed, from modeling the interaction between attackers and defenders (Varian, 2004; Bier, Oliveros, and Samuelson, 2007; Bohme and Moore, 2010), examining the role of different liability and information disclosure policies (Kannan and Telang, 2005; Arora, Caulkins, and Telang, 2006a; Kim, Chen, and Mukhopadhyay, 2011; August and Tunca, 2011; Lam, 2016), risk sharing and coordination possibilities between vendors and users (August and Tunca, 2006; Cavusoglu, Cavusoglu, and Zhang, 2008; Kim, Chen, and Mukhopadhyay, 2009), how market structure and competition impact security investments (Gal-Or and Ghose, 2005; Arora, Forman, Nandkumar, and Telang, 2010a), to the effect of product differentiation on software security (August, Niculescu, and Shin, 2014; Dey, Lahiri, and Zhang, 2014). Although theoretical studies are much more numerous than empirical ones, this

[3]See `https://econinfosec.org`

literature contributes to a large number of applied areas related to information security like software vulnerability management and patching, security investment decisions and market insurance, network security, malware and botnets, or payment system security.

0.2 Some barriers to overcome

This field of research went through a rapid development during the last two decades. However, it seems that the enthusiasm and effervescence of the beginning have fallen, with a recent slowdown in scientific production after a relatively short flourishing period.[4] This slowdown can be attributed to two main factors. The first one is related to some inherent difficulties in scientific practice. Studies on academic disciplines and research process suggest that the domain-specific structure of disciplines plays an important role in explaining why interdisciplinary fields have hardly a long-term success. Indeed, disciplines tend to direct intellectual focus toward the center of their own field (e.g., Chubin, 1976; Jacobs, 2014). Specifically, Raasch, Lee, Spaeth, and Herstatt (2013) show that within very few years after the inception of an interdisciplinary field, there is a shift from interdisciplinary to multidisciplinary research, and from joint problem solving to parallel problem-solving. Researchers from different disciplines still study the same topics years after the creation of the field, but do so increasingly from their own disciplinary lenses.

A similar phenomenon is observed in information security economics. Computer scientists favor analysis that provide practical results, like concrete security solutions that can be applied in real situations. Consequently, many of the analytical works produced by computer scientists using economic theory focus on numerical simulations and measures applied to a real case. On the other side, economists usually prefer to provide a general causal relationship or an analytical result that depends on particular assumptions, which may be considered as too theoretical to computer scientists. Furthermore, for economists, the information security environment may be considered as a specific situation in which their models can be applied. Thus unless the information security framework gives a stunning new insight or unless some splendid data set becomes available, there is not much reason for a generalist economist to study the

[4] For instance, the number of publications in peer-reviewed academic journals is superior between 2009 to 2014 than recently.

particular case of information security.

The second barrier for the development of the field of economics of information security is the difficulty to obtain reliable, exhaustive, and exploitable data, which particularly hinders the development of robust empirical studies. For example, according to a survey conducted in 2016 by UK's Bank and Institute of Directors, nearly three-quarters of cyberattacks went unreported.[5] Others have suggested that between 60% and 89% of all cyber-incidents go unreported (Edwards, Hofmeyr, and Forrest, 2016). The main reason is the confidential and strategic nature of the information. Victims of cyber incidents are unwilling to share information about a breach, not only because it increases the risk that the information is used against them, but also for reputational concerns (Telang and Wattal, 2007). Another difficulty is that much of the available data is collected by parties with a vested interest in under or over-reporting (Anderson et al., 2008). The absence of a compulsory and common monitoring system also restricts the reliability of data. For instance, few EU Member States regularly collect official data on cyber-related matters, which hinders comparability.[6] Fortunately, institutions become increasingly aware of the necessity to impose mandatory reporting and to define a consistent reporting mechanism. A significant advance is the creation of security-breach reporting laws, implemented today in many US states starting from California in 2003 and in the EU since the implementation of the General Data Protection Regulation (GDPR) in May 2018.

*

Despite these obstacles, tools and insights from economic analysis have clearly a lot to contribute to a better understanding of the market failures in information security, helping to improve the management of security risk. What is all the more fascinating about information security economics is that the rapid evolution of the cybersecurity landscape and its relation with innovation and new forms of digital markets make it more challenging to provide valuable and robust analysis.

In the light of all these elements, the objective of this thesis is to contribute

[5]Source: https://www.cso.com.au/article/595298/most-cybersecurity-breaches-go-unreported-uninsured-despite-executive-concern-barclays

[6]Source: 2019 briefing paper by the European Court Auditors on Challenges to effective EU cybersecurity policy (https://www.eca.europa.eu/en/Pages/DocItem.aspx?did=49416)

empirically to the research field of information security economics, by referring to traditional tools and knowledge in economics and especially in Industrial Organization – e.g., the relationship between competition and quality provision, contract theory, belief updating. I focus on new and evolving elements in the cybersecurity environment such as the use of free software revenue models in digital markets (Chapter 1), the introduction of crowdsourcing mechanisms to improve software security (Chapter 2), or the increasing involvement of third parties in software security (Chapter 3). I mainly conduct empirical analysis using original data that I have personally collected and consolidated. The general question I want to investigate is to understand the incentives of major actors that contribute to software security, namely software vendors, the white-hat hackers, security firms, and other third parties.

0.3 The market for software vulnerabilities, platforms and open innovation

The economics of platforms and open innovation are two important streams of research that have not been extensively exploited in information security, despite their importance in the current digital security environment and their potential value for this discipline. These two streams of research are central in the three articles that constitute the three chapters of my thesis.

While it has been largely admitted that better information sharing in security would be socially beneficial, the problem in sharing vulnerability information is similar to the one of the market for ideas: unless the information is revealed, one cannot accurately assess its value (Arrow, 1962). In the same way as this friction makes it difficult for innovators to sell their ideas profitably, it discourages organizations to share security information with others. To overcome this problem, engaging a third party that could build a trustful relationship and guarantee the interest of each agent appeared as a reasonable solution. These third parties are often referred to as security firms, security infomediaries, or vulnerability brokers in the literature (Kannan and Telang, 2005; Böhme, 2006). Mostly private companies, these intermediaries buy security-related information and share it within a closed group of subscribers. Clients are both the software vendors to whom they communicate the vulnerability information so that

the vendors can fix the security issues, and the users of software who want to protect their systems. In other words, a market for software vulnerabilities has been developed, where information about a new vulnerability has become henceforth a product that is bought and sold.[7]

Besides, the development of collaborative web tools and infrastructures allowed the emergence of web platforms that collaborate with the crowd for all kinds of tasks, including those that require creativity and innovativeness. Organizations in cybersecurity started crowdsourcing benign vulnerability identifiers – the so-called white-hat hackers – to improve software security. The first bug bounty platforms were launched in late 2013. They gather on one side of the platform the companies who want to organize a Vulnerability Research Program (VRP) and on the other side hackers who want to get compensated for discovering new vulnerabilities.[8] Bug bounty contests already existed before the emergence of such platforms.[9] But the scale became beyond comparison: for example, HackerOne, a US-based platform which currently dominates the market, collaborates with around 250,000 hackers in the world and more than 1,200 organizations have launched a VRP on it, including the U.S. Department of Defense and big companies in various industries like General Motors, Goldman Sachs, Twitter, Qualcomm or Starbucks.[10]

As it is well documented by the rich literature on platform economics, the notable success of bug bounty platforms comes mainly from their two-sidedness. Firms launch their contest on the platform because they can get access to a large pool of hackers, while hackers benefit from accessing a large number of VRPs on a single platform, which additionally allows them to manage a profile that cumulates experience from the different VRPs. The platform combines intermediation, assembling, and knowledge management activities all at the same time and greatly benefits from cross-sided network

[7] Böhme (2006) distinguishes four concepts that are often included in the terminology of "vulnerability market": bug challenges, vulnerability brokers, exploit derivatives, and cyber-insurance. Here, we limit the vulnerability market on the activity of buying and selling vulnerability information, for which vulnerability brokers act as an intermediary. Bug challenges are included as one example of how this market takes place.

[8] The terms "Vulnerability research program", "Bug bounty program", "Bug bounty contest" all designate an event that crowdsources individuals and rewards them for finding vulnerabilities in a software or a system.

[9] Crowdsourcing started being employed in software security already a decade ago by software vendors themselves. For instance, Netscape was one of the first company offering a financial reward to those who found valuable bugs in the beta version of its web browser in 1995.

[10] See `https://www.hackerone.com/customers`

effects between the two sides (Brousseau and Pénard, 2007). As it is common in this type of markets, the platform uses an asymmetric pricing strategy, where hackers get free access to the platform, while companies pay a usage fee. Hackers can multi-home, while companies usually launch their contests on a single platform, as it is costly for them to manage multiple contests using distinct platforms. Capitalizing from the knowledge and the data it acquires, the platform also offers a range of associated services.

All in all, bug bounty platforms are a good example of the development of two-sided business models in cybersecurity. The increasing dependency of companies and organizations to these platforms and their predominance is not yet of a central concern to the security community. Nonetheless, some papers show that the presence of a monopolistic market-based infomediary is welfare-decreasing. For instance, Kannan and Telang (2005) show that a market-based mechanism for vulnerability disclosure increases the inequalities in users' security by exposing non-subscribers to more attacks.

The rich literature on platform economics is for now barely used in information security, although a number of theoretical works account for different network externalities in security to explain some market failures in information security (e.g., Arora et al., 2006a; Choi, Fershtman, and Gandal, 2010; Dey, Lahiri, and Zhang, 2012). In the first chapter of this thesis, I study one particular element that characterizes a two-sided business model often used in digital markets and see how it affects the security level provided by the platform owner. Specifically, I show that in a market where the revenue model is based on offering the software to end-users for free while charging the advertisers, market concentration is not necessarily harmful to provision of a good-quality security. In the second chapter, I use data collected from a dominant bug bounty platform to study the crowdsourcing mechanism used in software security.

Open innovation is another important stream of research for this thesis. The development of a market-based mechanism for fixing software vulnerabilities demonstrates how external actors are increasingly involved in improving software security (and more generally corporate cybersecurity). In addition to it, considering that finding new vulnerabilities consists essentially in finding an innovative way to penetrate a system that has not been thought of before, then the process happening in the market for vulnerabilities perfectly fits with the definition of open innovation, which puts the

emphasis on the collaboration of in-house resources and external partners for innovative purposes (Pénin, Hussler, and Burger-Helmchen, 2011). In fact, among other reasons, because open source projects were initially born from a software development environment, information security always had a strong relationship with users and especially open-source communities.[11] Users have become developers, innovators and consumers all at the same time, i.e., they have become "prosumers". In this context, users and software vendors have a collaborative and cooperative relationship rather than a simple buyer-seller relationship (Rayna and Striukova, 2016).

Thereby, the security of a software depends not only on the company that develops the software but also on external communities. Furthermore, the progressive implication of private companies in open-source projects and in the security community, along with the development of a market-based mechanism in vulnerability disclosure, illustrate how open innovation strategies are widely used in IT security. In summary, it is difficult to have a general understanding of the mechanisms in information security by only focusing on software vendors' behavior or on the vendor-user relationship.

However, for now, existing works in information security economics generally focus on the interactions between software vendors, users, and attackers, ignoring the implication of other third parties. My dissertation fills this gap by studying the behavior and the incentives of multiple actors actively involved in cybersecurity: the first chapter focuses on software vendors' behavior, the second chapter on crowdsourced individual researchers, and the third chapter on all the different actors that participate in the discovery of software vulnerabilities, including corporate users, security firms and public organizations.

0.4 Presentation of the three chapters

This thesis is organized in three chapters, each addressing a separate research question.

One of the main ideas in the literature of information security economics is that systems often fail because organisations that are able to defend those systems do not bear the full costs of failure. A significant part of the literature accuses software editors

[11] Von Hippel and Von Krogh (2003) recall that open source became a central feature of the "hacker culture". In communities of open-source programmers, the term 'hacker' designates initially a person who enjoys exploring the details of programmable systems and how to stretch their capabilities, as opposed to most users. It is a positive term applied to talented and dedicated programmers.

of negligence, showing how they may prefer to release a software early than to release it with fewer bugs, or how they delay the release of a patch unless the vulnerability is disclosed to the public (Arora et al., 2006a; Choi et al., 2010). Nevertheless, few works actually provide empirical evidence on software vendors' behavior in securing their products. In particular, within a digital environment where new forms of markets such as two-sided markets have become abundant, does competition affect their investment in security in the same way? This is where I focus my interest in the first chapter. I analyze the relationship between competition intensity and the security level provided by a software vendor, examining the case of a software at the center of Internet security, namely the web browser, in which the vendors derive their revenue from advertising and compete in quality. I find that a higher market concentration positively impacts the vendor's responsiveness in patching vulnerabilities, although this effect is reduced when the vendor is too dominant.

In a second chapter, I focus on the crowdsourcing mechanism of white-hat hackers through the vulnerability research programs (VRPs), which is representative of the market for vulnerabilities that capitalizes on third parties' contribution. A VRP is a typical example of an innovation contest run in a web environment. The main challenge in managing such contest is to attract enough participants, while limiting the low-quality ones. In this context, I study how the hackers' perception of the uncertainty to obtain a reward, determined by the level of information the contest provides about the contractual terms through its written policy, affects participations and therefore the outcome of the contest. I show that this self-selection process of participants leads to a trade-off between having a larger number of participation but attracting less performant participants and attracting higher quality participants but generating fewer participations.

Lastly, while the literature has mostly explored how to incentivize software vendors to better secure their software, a dearth of research exists on the role of third parties that contribute to software security. In a third chapter, I try to go beyond the limited framework of software vendor and users' relationship, and examine third parties. In particular, I examine how the disclosure of a critical vulnerability affects the contribution of third parties in discovering new vulnerabilities and compare it with the software vendors' reaction. I find that third parties' overall contribution in improving

software security is considerable and that their contribution is significantly affected by externalities such as the disclosure of a critical vulnerability.

Chapter One: The effect of competition intensity on software security - An empirical analysis of security patch release on the web browser market[*]

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 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 2009 to 2018. 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: competition, software quality, information security

[*]I thank my thesis advisor Marc Bourreau for his patient guidance and support, Rainer Böhme, Grazia Cecere, Jens Grossklags, Thomas Maillart and Tong Wang for valuable suggestions. I also thank the anonymous reviewers at WEIS 2017 (San Diego), audience of Toulouse School of Economics 2018 Digital Economics conference (Toulouse), AFSE 2016 (Nice), CRESSE 2016 (Rhodes), EARIE 2016 (Lisbon), and JEI 2016 (Palma).

1.1 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).[1] 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 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.[2]

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.[3] 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.[4]

In this paper, we propose to analyze in greater detail and with empirical evidence, the relationship between competition intensity and software vendors' patch release

[1] By IT monoculture, we refer to an IT environment where a large fraction of systems run the same software (Lala and Schneider, 2009).

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

[3] 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.

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

behavior, 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 positive impact of market concentration on the security investment choice becomes less clear: the larger the market share of the dominant firm, the larger its installed base of users and the lower its 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 874 web browser vulnerabilities, discovered and patched from January 2009 to June 2018. We find strong evidence that higher market concentration positively impacts the vendor's responsiveness in patching vulnerabilities. On the other hand, a dominant position in the market affects negatively the editor's promptness to release a patch and reduces the positive effect of having less competitors.

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.[5] 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 1.2, we review the relevant literature. Then we describe the specificities of the web browser market and its

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

revenue model in Section 1.3. Section 1.4 presents the theoretical model. We then describe in Section 1.5 the link between the security quality of a software and patching vulnerabilities, and the econometric model. Section 1.6 presents the data, estimation results follow in Section 1.7 and the final section provides some conclusions.

1.2 Literature review

This paper contributes to two main streams of research: the 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.

Analytical works on investment in software security deal with various questions, from welfare and investment implications of interdependent security risks (e.g. August et al., 2014; Acemoglu, Malekian, and Ozdaglar, 2016), the role of different liability and information disclosure policies (Kannan and Telang, 2005; Arora et al., 2006a; Choi et al., 2010; August and Tunca, 2011), to coordination between software vendors and customers in the patch management process (August and Tunca, 2006; Cavusoglu et al., 2008). 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 in information and cost sharing (Gal-Or and Ghose, 2005; Kim et al., 2009).

On the other hand, empirical work is globally limited to issues related to vulnerability information disclosure (Gordon, Loeb, Lucyshyn, and Sohail, 2006; Telang and Wattal, 2007; Arora, Krishnan, Telang, and Yang, 2010b; Mitra and Ransbotham, 2015), presumably because of the lack of available data to researchers. Among existing studies, several papers exploit similar data as ours. Arora et al. (2010b) investigate software vendor's responsiveness to vulnerability information disclosure and the impact of information disclosure on the frequency of cyber attacks. Ransbotham, Mitra,

and Ramsey (2012) 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 some of the elements studied by these papers – the patching time, vulnerability and software characteristics – but we are not interested in the effect of information disclosure. Temizkan, Kumar, Park, and Subramaniam (2012) also compile a similar data set as ours to analyze how characteristics specific to the vulnerability or the software affect the patch release behavior of software vendors.

To the best of our knowledge, only one paper has studied the impact of competition on software publishers' patching behavior (Arora et al., 2010a). Interestingly, our findings are opposed to their results. While they find that vendors are more responsive when there is more competition, our results show that vendors provide a better security quality when the market is more concentrated. Our approach actually 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 puts 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. By studying a market at the heart of internet security and which presents a revenue model increasingly used by web applications, our empirical study contributes to better evaluating the actual impact of "software monoculture" on software security. Lastly, we study a relatively long period of time – a ten-year-period, which strengthens the robustness of our study, while Arora et al. (2010a) consider a large panel of different software during a shorter period.[6]

In order to define our empirical hypothesis, 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

[6]Our data covers an 11-year-period from 2007 to 2018, while Arora et al. (2010a) consider a 4-year-period from 2000 to 2003

a topic of interest in the literature for a long time, essentially 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.

Dey et al. (2014) study the incentives for quality differentiation in the security software market. By incorporating a negative network effect that arises from a consumer's desire to free-ride when other consumers invest in security, they explain why in the security software market we observe relatively high quality products with high price and little quality differentiation. It is interesting to mention that this paper is one of the few that gives an insight on quality competition in software market, but the studied framework is distant from ours.[7] 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 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 legal services (Domberger and Sherr, 1989), airline industry (Mazzeo, 2003), supermarket (Matsa, 2011), and software (Arora et al., 2010a). All of these papers study markets with one sided business models and find that an increasing competition leads to higher quality provision. We consider the case of "free" products where firms compete in quality and our results show that on the contrary, market concentration leads to higher product quality.

1.3 The web browser and its revenue model

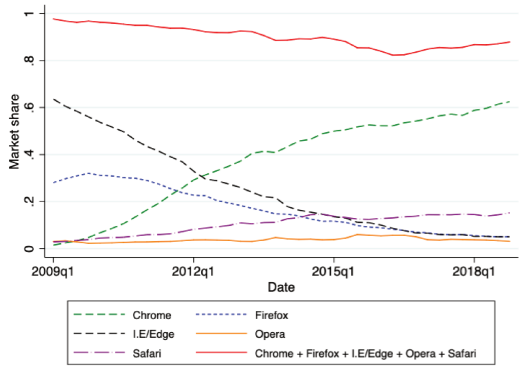
A web browser is a software that gives access to information resources on the World Wide Web (WWW). Its primary role is to retrieve and display content from remote web servers using the HyperText Transfer Protocol (HTTP).

[7] In Dey et al. (2014), it is consumers who buy a security product while in our paper we aim to study the investment behavior of firms – the software vendors –. Moreover, in their paper, products – security software – are not free of charge.

Today, the most popular web browsers are Google Chrome, Mozilla Firefox, Microsoft Internet Explorer, and Apple Safari. Figure 1.1a shows the evolution of their market shares from 2009 to 2018. 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 the studied 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 1.1b).

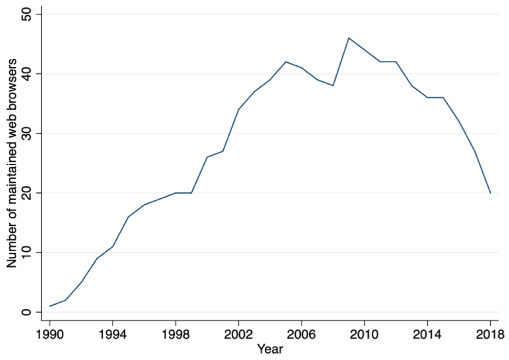
Figure 1.1

(a) Evol. of web browsers' market share



Source: Statcounter.com

(b) Number of maintained web browsers



Source: Wikipedia "History of web browser"

In the 1990's, web browsers were sold at a unit price or included in the application suite of an operating 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 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 1.9 and Figure 1.5 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. Conceivably, this is the reason why Google heavily communicates on its web browser's simplicity, performance and security for instance.[8] In this paper, we consider that the quality of a web browser corresponds to its security level.

Besides, 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.[9] 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.

1.4 A model of competition in security quality

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

[8] "Speed, simplicity and security are the key aspects of Google Chrome": When launching Google Chrome, the official press events and press release communicated about Chrome web browser using key words such as "lightweight", "fastest", "speed", "simplicity", "secure". Source: https://googleblog.blogspot.fr/2009/11/releasing-chromium-os-open-source.html

[9] 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.

consumers.

On the demand side, there is a unit mass of consumers, each of which acquiring one web browser. The market is fully covered.[10] We assume that consumers' utility depends only on the security quality level and that the revenue a software publisher derives per user is exogenous.

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., 2006a; 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:

$$\pi_i = a \left[b_i + (1 - B) \frac{s_i}{\sum_{j=1}^{n} s_j} \right] - \frac{\phi s_i^2}{2} \,. \tag{1.1}$$

In equation 1.1, b_i is a part of the demand already acquired by firm i which insures a gain of $a \cdot b_i$ regardless of the security level it chooses to provide. Conversely, $1 - B$ is the share of non loyal consumers for which all firms in the market compete for.

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

$$\frac{\partial \pi_i}{\partial s_i} = a(1 - B) \frac{\sum_{j=1}^{n} s_j - s_i}{(\sum_{j=1}^{n} s_j)^2} - \phi s_i = 0 \,. \tag{1.2}$$

[10] 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.

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 (1.2), we obtain the symmetric equilibrium security quality level for each firm:

$$s_i^* = \sqrt{(1-B)\cdot\frac{n-1}{n^2}\cdot\frac{a}{\phi}}\,. \tag{1.3}$$

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 1.3 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. Take 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, we set $B = b_i = \alpha_i m_i$, 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. Then 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 is in a dominant position, 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.

1.5 Empirical specification

In light of what the theoretical model suggests, we seek to examine empirically how the competition intensity affects software publishers' security investment level. For this purpose, we consider the responsiveness of a software publisher in fixing a *vulnerability* in its software as a measure of the security quality it provides.

In this section, we define what is a software vulnerability and we describe its lifecycle. This allows us to explain the rationale behind considering the responsiveness of a software vendor in releasing a security patch as a measure of its investment level in security. We then present in detail the econometric specification.

1.5.1 Software vulnerabilities and security quality

In information systems and computer security, a vulnerability eakness in the architecture, design, or code, which leaves the software or the system open to potential for exploitation.[11] 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 1.2 – Lifecycle of a vulnerability.

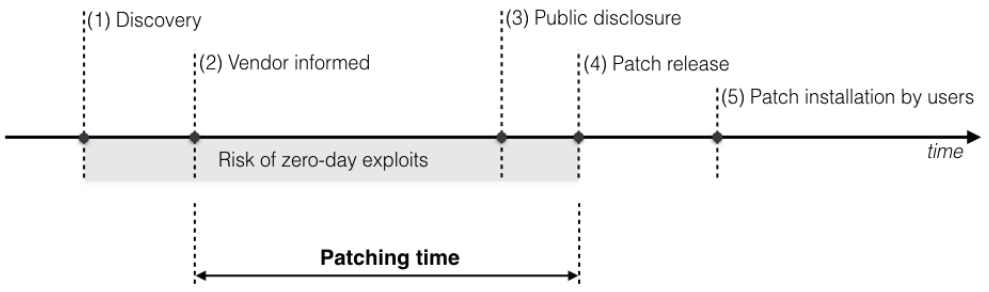


Figure 1.2 presents major events that may occur during the lifecycle of a software vulnerability, from its discovery, either by a malevolent or a well-intentioned actor, to

[11] Source: https://cwe.mitre.org.

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.

Software vulnerabilities are often 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 security flaw. 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. 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., 2006a).

In our empirical model, we consider the promptness of a software vendor to provide security patches as a proxy of its investment level, which determines the software's security quality. The security level of a software does not solely depend on the effort exerted in fixing the security flaws discovered ex-post product delivery. However, we consider that the ability of a firm to be responsive when it discovers a vulnerability represents its willingness to secure its product, i.e., the security quality it chooses to provide. Indeed, in order to be fast in delivering the security patches, the vendor should do considerable investment in security beforehand, such as having dedicated resources for software security or an internal organization that is able to deliver unexpected new updates.

At the same time, the vendor may decide to spend more or less resources to deliver a given patch according to the order of priority at the given moment. Specifically, we measure the duration between the moment the software publisher is informed about the existence of a vulnerability and when it releases a patch to fix it. This duration corresponds to the difference between time (4) and time (2) in the timeline in Figure

1.2, which we call the "patching time".

Time (2) corresponds to the moment when the vendor discovers the vulnerability. The vendor can either discover the flaw by himself or be notified by a third party. While it is indispensable to know the exact date of time (2) in order to measure the responsiveness of a software editor in securing its product, it is an information often kept confidential by the software vendor. In our paper, we use information that is disclosed by third parties who have discovered the vulnerabilities. Indeed, some private organizations pay individual researchers to identify security flaws in software that are used by their clients, and an increasing number of software vendors manage their own security research team or offer monetary incentives 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.

Two main considerations will affect the vendor's patching decision (4): the cost to develop the patch and the extent to which it internalizes 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 competitive pressure, the complexity to fix the flaw, the human resources available to develop the patch, or the impact on its reputation especially when the information is disclosed to the public. 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 is equal.

Note that in the case of web browsers, it is common that one update contains several security patches. Thus time (4) is equal for all security fixes delivered in a single update. However, this does not alter the idea that (4) − (2) reflects the security level provided by a vendor. Indeed, the shorter the period between when the editor has developed the patch and it actually releases the update package is, the more regularly the editor is likely to publish updates, which is more costly and which would mean in turn that the editor invests more in security.

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. Arora et al., 2006a). Vulnerability research organisms

generally apply a policy of "responsible disclosure", keeping the vulnerability information confidential during a period of time, after having notified the affected vendor. Usually, they also grant an additional period of confidentiality if the editor asks for it. Our study does not account for the impact of information disclosure.[12]

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 responsibility of software vendors in users' patch installation (Cavusoglu et al., 2008), but it is beyond the scope of this paper.

1.5.2 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 may have an impact on its responsiveness. Relying on the theoretical results in Section 1.4, we particularly focus on two measures 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:

$$Patching\_time_{ij} = \beta_0 + \beta_1 Concentration + \beta_2 X_i + \beta_3 X_j + \beta_4 Time\_effect + \epsilon\,, \tag{1.1}$$

where $Patching\_time_{ij}$ is the time spent by web browser publisher j in releasing a security patch for a given vulnerability i discovered at a given date. $Concentration$ is a measure of market concentration at the date the vulnerability was discovered. We use 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 parameter β_1.

$Patching\_time$ is a measure of the responsiveness of a vendor in securing its product and we consider that it reflects its investment level in security. Thus regressing $Patching\_time$ by $Concentration$ arises some reverse causality issue. Our objective

[12] We are not able to capture the effect of applying a responsible disclosure policy on the responsiveness of a vendor as the three programs from where our data come from are applying similar rules.

here is to examine the impact of market concentration on a firm's security investment but market shares and subsequently market concentration can also be affected by the security quality provided by the firm. We introduce an instrument to correct for this potential endogeneity problem. We identified the ratio between Windows and Mac OSX market shares in personal computer (PC) operating system (OS) market as a valid instrument. Indeed, the competition in the OS market has a direct impact on web browser market concentration (for instance, because of the presence of pre-installed web browsers on operating systems), but shares in these markets are not affected by the security level of web browsers. The same instruments are applied for the HHI. The detailed identification tests for the validity of the instruments are reported in Section 1.7.

Additionally, we control for a list of other factors that may have an impact on the responsiveness of a web browser publisher in releasing a security patch. First, X_i 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).[13] 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.[14] In our model, CVSS values are directly used as a variable ($Vulnerability\_severity_i$), while the CWE catalog is used as a vector of dummies.

Secondly, the vector X_j represents characteristics specific to the affected web browser and its publisher, such as the age of the software and publisher fixed effects. The $Software\_age_j$ 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

[13]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.

[14]The principal common weaknesses that characterizes web browser vulnerabilities are: improper restriction of operations within the bounds of a memory buffer, improper control of generation of code, information exposure, improper input validation, numeric errors, permissions, privileges, and access controls, concurrent execution using shared resource with Improper synchronization, ressource management errors, and user after free weaknesses.

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.[15] We also account for software and 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, $Time\_effect$ captures effects that are correlated with time, such as the growing awareness both among consumers and practitioners about security issues, which in turn, makes editors more responsive in patching vulnerabilities. For that, we use the quarterly date in which the vulnerability is discovered ($Qyear$). In alternative regressions, we use the total number of mobile broadband subscription in the world to account specifically for the effect of the rise of mobile use ($MobilevsBroadband$). 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:

$$\begin{aligned} Patching\_time_{ij} = \beta_0 &+ \beta_{1a} Concentration + \beta_{1b} Big\_mshare_i \\ &+ \beta_{1c} Concentration \cdot Big\_mshare_i \\ &+ \beta_2 X_i + \beta_3 X_j + \beta_4 Time\_effect + \epsilon \end{aligned} \tag{1.2}$$

In this equation, we added to the baseline model (1.1) the variable Big_mshare_i and an interaction term of this variable with $Concentration$. Big_mshare_i is a dummy equal to 1 if the affected web browser publisher's market share is greater than a certain percentage of the market at the moment the vulnerability is discovered. Following the definition of a dominant position given by the OECD, we use the threshold values of

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

40% and 50% for Big_mshare_i.[16] This variable captures the effect for a firm to have a relatively large installed base (of loyal consumers) compared to its competitors at the time it is informed about the security flaw. The coefficient of the interaction term β_{1c} represents the difference in the effect of market concentration between when the affected firm is in a dominant position in the market and when it is 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 weighted least squares (WLS) and weighted likelihood negative binomial (NB) regressions. To instrument *Concentration*, we use instrumental variable regression and control function approach (Wooldridge, 2015).

1.6 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 2009 to June 2018. 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 browsers and operating systems from Statcounter.com, (4) the evolution of the number of mobile internet subscription from the International Telecommunication Union (ITU), and (5) 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.[17] ZDI and iDefense are the two main players in the commercial vulnerability market, which financially reward security researchers in exchange for information about security flaws that were unknown before. Both ZDI and iDefense notify the affected vendors and communicate with them until they release a patch. Similar to these programs, Google Project Zero is a project initiated by Google in 2014, which focuses on vulner-

[16] https://stats.oecd.org/glossary/detail.asp?ID=3199

[17] iDefense was acquired by Accenture in 2017 but data we collect from iDefense dates from 2009 to 2017.

abilities 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. We excluded Google Chrome's vulnerabilities found by Google Project Zero from our data set in order to have the same configuration for every observation: vendors are notified about the discovery of a vulnerability by a third party. From these three programs, we compiled information related to web browser vulnerabilities they discovered. We consider only vulnerabilities specifically assigned to web browser publishers, i.e., that can be fixed by 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.

One observation consists in a pair of vulnerability i and the associated web browser publisher j. 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 between the date at which the vulnerability is reported to the vendor and the delivery date of the associated patch in number of days. 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 vulnerability research programs' record. These programs are actually some of the few sources that reveal 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) for each quarter from January 2009 to June 2018 using these data.[18] Specifically, the market share of a software vendor is the sum of the user

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

share of all the web browsers owned by this vendor. These market shares and market concentration measures are used as our main explanatory variables.[19] Market shares of operating systems that are used to build the instrument for our main explanatory variables were also collected from the same website. Specifically, we collected Apple Mac OSX and Microsoft Windows OS market shares in PCs from January 2009 to June 2018.

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).[20] Additionally, we use the yearly number of mobile broadband subscription in the world from 2009 to 2018 provided by the ITU, which is used as a control variable accounting for the rise of mobile use. Lastly, from each web browser's website, we collected information about each version in order to determine how old the affected versions affected by the vulnerability are. For a given vulnerability-browser pair, we use the oldest versions' age to build the $Software\_age$ variable.[21]

1.6.1 Representativeness of the studied sample

Table 1.1 – Comparison with NVD listed vulnerabilities

	Number of vulnerabilities			Sum of CVSS score		
Product	Our data	NVD	$\frac{ourdata}{NVD}$	Our data	NVD	$\frac{ourdata}{NVD}$
Google Chrome	33	1555	2.1%	254	9288	2.7%
Mozilla Firefox	68	1399	4.9%	606	8735	6.9%
Microsoft IE/Edge	580	1457	39.8%	4896	10344	47.3%
Apple Safari	193	890	21.7%	1398	5870	23.8%
Total	874	5301	16.5%	7154	34237	20.9%
Average CVSS score				8.19	6.46	

Our analysis is based on 874 observations consisting in web browser vulnerabilities identified from January 2009 to June 2018 by the three vulnerability research programs

[19] 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". Big_mshare is equal to 1 when the affected web browser's market share exceeds a certain amount; values of 40% and 50% are used.

[20] 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.

[21] Even though a vulnerability affects several versions of a given web browser, patches are always published the same day for all versions. Thus one observation in our data set includes all versions of a given web browser affected by one vulnerability.

mentioned earlier. The description of variables and summary statistics are reported in Table 1.11 and Table 1.10 of the Appendix.

Our data set represents 16.5% of the total web browser vulnerabilities referenced on NVD during the studied period in terms of number of observations.[22] Since the impact of a vulnerability on a software security depends on its severity, the representativeness of our data can also be measured by summing the severity of the observations. Table 1.13 reports the representativeness of our data compared to NVD depending on whether we consider the number of discovered vulnerabilities or the sum of their CVSS. Our data represents 20.9% of the mother population when we consider the severity of the discovered vulnerabilities.

One important issue we note by comparing our data set to NVD referenced vulnerabilities is that some web browsers are more represented than others in our data set. This may come from the fact that vulnerabilities in our data are choice-based: the programs that pay for vulnerability discovery and from where our data comes from may have some preferences in the software and the severity of vulnerabilities they are looking for. To overcome this problem, we apply a probability weight on each regression we estimate as suggested by Solon, Haider, and Wooldridge (2015). More precisely, we use Weighted Linear Regressions (WLS) instead of OLS regressions by weighting each contribution to the sum of squares by its inverse probability of selection and for Poisson regressions we maximise the log likelihood that weights each observation's contribution to the conventional log likelihood by its inverse probability of selection. We apply two types of probability weights and compare the results: the first one is based on the number of vulnerabilities yearly referenced in NVD for each web browsers (PW1), the second one is based on the sum of their severity (PW2).

1.6.2 Dependent variable

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

[22] NVD is sponsored by the US government and is one of the most comprehensive database for public vulnerability reports. We thus consider vulnerabilities listed on this database as the mother population.

to estimate our models.[23] Thus we perform WLS and Weighted Poisson regressions.[24] 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 1.2 – Mean value of *Patching_time*

	Mean	Std. Err.	Weighted Mean (PW1)	Lin. Std. Err.	Weighted Mean (PW2)	Lin. Std. Err.
All	95.3	1.61	91.3	10.4	88.4	11.4
Apple Safari	88.4	2.99	95.6	8.88e-16	92.3	4.44e-16
Google Chrome	85.4	14.0	107.3	3.15e-15	107.5	3.39e-16
Microsoft IE/Edge	100.7	1.89	98.7	4.86e-16	95.6	4.94e-16
Mozilla Firefox	74.9	6.34	63.2	7.42e-16	56.3	1.22e-15

Table 1.2 presents mean values of patching time for each browser during the studied period. We see that applying a probability weight modify the mean values as it corrects for the unbalanced representativeness of some web browsers compared to the mother population from NVD. For instance, the mean *Patching_time* for Chrome has increased by more than 20 days, from an average patching time of 85 days to 107 days. Nonetheless, regression results show that this bias does not affect qualitatively the results of our findings (See Section 1.7).

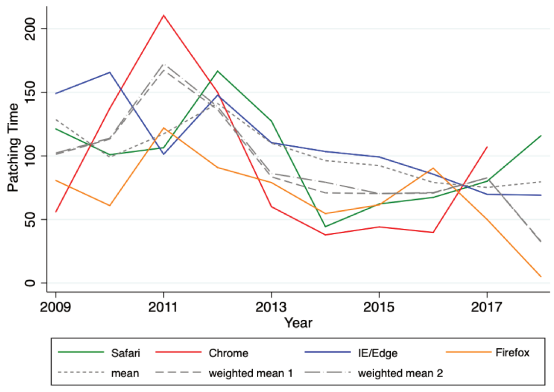


Figure 1.3 – Evolution of average *Patching_time* of web browser vulnerabilities from 2009 to 2018

Graph in Figure 1.3 plots the evolution of the average patching time from 2009 to 2018. We observe that over time, vendors become generally more responsive in securing their product. Moreover, overall, Mozilla takes less time than other editors to release a

[23] 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.

[24] Poisson regression is the baseline model for count data.

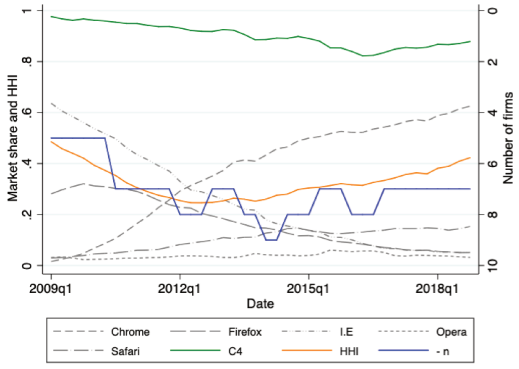
security patch. Otherwise, we do not observe particular trends from this graph.

1.6.3 Main explanatory variables

For the main explanatory variable *Concentration*, we use two different measures: the number of firms competing in the market and the HHI. Figure 1.4a shows the evolution of market concentration in the web browser market using different measures. The four-firm concentration ratio (C_4) is widely used in the economic literature, but rather because the HHI requires the complete distribution 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.[25] Moreover, C_4 is highly correlated to *Qyear* (see Table 1.3). We thus consider that it is not a better measure of concentration in our model.

Figure 1.4

(a) Measures of concentration



(b) Dominant web browsers

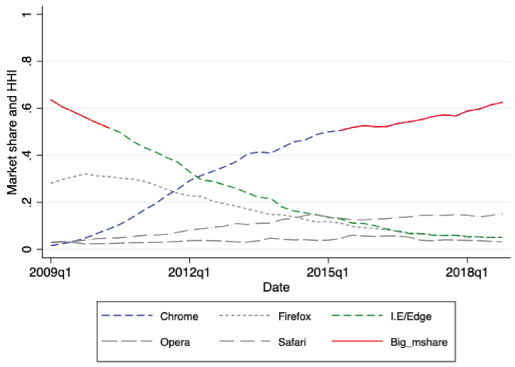


Table 1.3 – Correlation matrix of market concentration measures

Measures	*-n*	C_1	C_4	*HHI*	*Qyear*
-n	1.00				
C_1	0.43	1.00			
C_4	0.40	0.40	1.00		
HHI	0.72	0.88	0.01	1.00	
Qyear	-0.33	0.45	-0.93	0.06	1.00
Big_mshare (50%)	0.36	0.22	0.24	0.37	-0.29
Big_mshare (40%)	0.25	0.10	0.30	0.24	-0.33

In the second model we estimate, the main explanatory variable *Big_mshare* is a dummy variable equal to 1 if the affected web browser publisher's market share is

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

greater than 40% and 50% of the market at the moment the vulnerability is discovered. Values of 40% or 50% are set according to the definition of a "dominant position" given by the OECD, which corresponds in our case, to the dominance of Microsoft Internet Explorer and Google Chrome as illustrated in Figure 1.4b in Appendix.

1.7 Estimation results

In this section, we report the estimation results for the econometric models. We first show results for the baseline model (1.1), which estimates the effect of market concentration on the responsiveness of a publisher. We next estimate model (1.2), in which we account both for the number of firms competing in the market and for whether the web browser is "dominant" in the market at the time the security flaw is discovered.

1.7.1 Effect of competition intensity on the responsiveness of web browser publishers in patching security flaws, using the number of firms and the HHI as concentration measures

To begin with, we report in Table 1.4 regression results for model (1.1), which estimates the effect of market concentration on the patch release time. In each triplet of columns, we report both OLS and WLS results, using either a probability weight based on the number of vulnerability (PW1) in the NVD mother population, or on the sum of their severity (PW2). We use two different measures of concentration – the number of firms and the HHI. Consistent with our theoretical model, the coefficients of *Concentration* are negative. That is, web browser publishers are more responsive in patching security flaws when the market is more concentrated, i.e., when the market is less competitive. More precisely, a web browser publisher releases a security patch about 3 days earlier when there is one less competitor in the market and about 1 day earlier when the HHI increases by 0.01 point. We note that the results does not change qualitatively when we correct for the sampling issue by weighting the regressions. In the main tables thereafter, we only report results for weighted regressions. Additionally, using OLS or Poisson regressions produce very similar results (See Table 1.12 in Appendix).

Regarding other explanatory variables, the impact of vulnerability severity score

Table 1.4 – Results for model (1.1) using -n and HHI as concentration measures

Using as *Concentration* measure:	-n		
	OLS	WLS PW1	WLS PW2
	(1)	(2)	(3)
Concentration	-3.401**	-2.776*	-2.817*
	(1.726)	(3.386)	(4.553)
Vulnerability_severity	-2.530	-15.62	-14.73
	(1.706)	(9.533)	(9.003)
Vulnerability_type	✓	✓	✓
Software_age	1.597***	2.343*	2.028*
	(0.343)	(0.990)	(0.776)
Apple	-18.00***	-42.84	-37.22
	(4.965)	(19.03)	(18.68)
Google	-22.70***	-20.92	-14.54
	(8.174)	(11.49)	(12.61)
Mozilla	-43.33***	-60.05**	-55.17**
	(6.166)	(11.54)	(14.20)
Qyear	-1.789***	-3.395**	-3.253***
	(0.243)	(0.641)	(0.484)
Constant	474.3***	926.5**	883.1**
	(60.12)	(228.0)	(201.4)
Observations	874	874	874
R^2	0.197	0.333	0.347
VIF for *Concentration*	1.38		
Using as *Concentration* measure:	HHI		
	(4)	(5)	(6)
Concentration	-93.88***	-190.0**	-188.5**
	(32.01)	(59.27)	(66.42)
Vulnerability_severity	-3.005*	-16.28	-15.20
	(1.716)	(8.737)	(8.336)
Vulnerability_type	✓	✓	✓
Software_age	1.558***	2.386	2.082*
	(0.342)	(1.136)	(0.877)
Apple	-17.10***	-43.01	-36.93
	(4.927)	(18.59)	(17.66)
Google	-22.97***	-25.59	-18.35
	(8.151)	(11.10)	(12.13)
Mozilla	-42.54***	-60.58**	-54.21**
	(6.151)	(11.73)	(13.56)
Qyear	-1.670***	-3.458**	-3.179**
	(0.233)	(0.792)	(0.616)
Constant	506.4***	1,030**	954.1**
	(61.27)	(241.3)	(198.3)
Observations	874	874	874
R^2	0.201	0.354	0.365
VIF for *Concentration*	1.24		

Note: OLS regressions for Columns (1) and (4), Weighted Linear Regression using a probability weight based on the number of vulnerabilities yearly referenced in NVD for each web browsers for Columns (2) and (5), and again a WLS with a probability weight based on the sum of the severity of the vulnerabilities in NVD for Columns (3) and (6). As we use 8 distinct dummies to account for *Vulnerability_type* fixed effects, we do not report their coefficients in this table. Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1

($Vulnerability\_severity$) is negative, meaning that editors are likely to respond faster to more severe vulnerabilities. Besides, by regressing $Vulnerability\_severity$ by $Vulnerability\_type$ dummies, we note that there is a strong correlation between the type of the vulnerability and the severity score (see Table 1.7 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.

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 in investing on it. Moreover, the older the version is, the code is more likely to be longer and complex; therefore, it takes more time to identify and fix the flaw (see paragraph 1.5.2). As to vendor-specific characteristics, we note that coefficients for vendor dummies loose their significance when we correct for the endogenous sampling issue, except for Mozilla dummy. We find that Mozilla Firefox vulnerabilities are patched on average 45 to 50 days faster than vulnerabilities of Microsoft Internet Explorer and faster than Google Chrome or Apple Safari by 20 to 30 days, confirming the idea suggested by the literature about the benefits of open source projects.

Lastly, coefficients for $Qyear$ are significant and negative: each quarter, the patch release time for web browser vulnerabilities is reduced by around 3 days. We attribute this effect on trends that we cannot measure but which are correlated with time, such as the growing awareness of and interest in web application security. For instance, the increasing number of mobile broadband subscription seems to explain an important part of this effect (see Appendix 1.8) although we cannot attribute the "time effect" solely to the rise of mobile use.

We now turn into instrumental variable estimation to fix the simultaneity issue between $Concentration$ and $Patching\_time$. As a reminder, we instrument

Table 1.5 – Results for model (1.1) - instrumenting *Concentration* with *WinvsMac_PCs*

Using as *Concentration*:	*-n*			
	PW1		PW2	
	IV	FS	IV	FS
	(1)	(2)	(3)	(4)
Concentration	-26.13***		-24.89***	
	(8.600)		(8.898)	
Vulnerability_severity	-16.60***	-0.0169	-15.12***	-0.00224
	(5.877)	(0.0445)	(5.742)	(0.0405)
Vulnerability_type	✓	✓	✓	✓
Software_age	2.320**	-0.00560	1.887*	-0.0122
	(0.991)	(0.0114)	(1.000)	(0.00971)
Apple	-37.51***	0.321**	-29.60**	0.411***
	(12.17)	(0.139)	(12.56)	(0.122)
Google	-29.87**	-0.167	-20.25	-0.0826
	(13.79)	(0.139)	(14.30)	(0.138)
Mozilla	-57.19***	0.129	-49.65***	0.197
	(12.13)	(0.137)	(14.50)	(0.137)
Qyear	-4.329***	0.0800***	-3.854***	0.0903***
	(0.795)	(0.0124)	(0.747)	(0.0115)
WinVsMac_PCs (Instrument)		0.305***		0.310***
		(0.0236)		(0.0225)
Observations	874	874	874	
R^2	0.243	0.471	0.276	0.444
Partial R^2 (excluded instr.)		0.290		0.291
F stat (excluded instr.)		166.6		190.8
Using as *Concentration* measure:	*HHI*			
	(5)	(6)	(7)	(8)
Concentration	-274.2***		-264.6***	
	(87.64)		(91.48)	
Vulnerability_severity	-16.63***	-0.00172	-15.41***	-0.00131
	(5.535)	(0.00165)	(5.482)	(0.00146)
Vulnerability_type FE	✓	✓	✓	✓
Software_age	2.403***	-0.000231	2.096**	-0.000355
	(0.914)	(0.000486)	(0.947)	(0.000431)
Apple	-42.80***	0.0113**	-36.42***	0.0129**
	(11.08)	(0.00568)	(11.46)	(0.00527)
Google	-28.13**	-0.00952*	-20.18	-0.00751
	(12.53)	(0.00564)	(13.03)	(0.00531)
Mozilla	-60.67***	-0.000420	-53.54***	0.00386
	(12.05)	(0.00681)	(14.46)	(0.00667)
Qyear	-3.536***	0.0105***	-3.181***	0.0110***
	(0.638)	(0.000607)	(0.628)	(0.000541)
WinVsMac_PCs (Instrument)		0.0290***		0.0292***
		(0.00125)		(0.00113)
Observations	874	874	874	874
R^2	0.350	0.714	0.362	0.710
Partial R^2 (excluded instr.)		0.681		0.670
F stat (excluded instr.)		536.5		665.0

Note: 2SLS and Weighted 2SLS regressions. All the 8 distinct *Vulnerability_type* dummies are included in the regressions but we do not report their coefficients in this table. Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1

Concentration by the market share ratio between Microsoft Windows and Apple Mac OS in the desktop market. We consider that market shares in desktop OS may affect the market concentration (and thus the competition intensity) in the web browser market but do not affect directly the responsiveness of a web browser publisher in patching a vulnerability. Results are presented in Table 1.5. From columns 1 to 4, we report results for estimation using the number of firms as concentration measure, while columns 5 to 7 report results when using the HHI as concentration measure. First, results show stable coefficients for every variable between estimation 1 and 3 and 5 and 7, suggesting little influence of the sampling. First stage regressions show that the excluded instrument is correlated with the endogenous variable, meaning that the market structure of operating systems has a significant influence on the web browser market structure. Additionally, we have one instrument and the F statistics on the excluded instrument are much more greater than 16.38, which is the size of nominal 5% Wald test. The Stock-Yogo test is satisfied and we consider that our instrument is not weak. Estimation for the second stage shows that using an instrument for *Concentration* does not change the sign of its effect on the *Patching_time*. On the other hand, it enlarges significantly the magnitude of the effect. It is reasonable to find a greater positive effect of market concentration on the responsiveness of a vendor as we are considering that a reverse causality exists between the patch release time and concentration. Indeed, the reverse causality we are worried about here is the fact that the responsiveness of a vendor in releasing a patch, which is a proxy of the security quality it provides to users, has a positive impact on user shares and in turn on market concentration. That is, a longer patch release time would decrease the firm's market share and in turn decrease market concentration. Thus controlling for this reverse causality would enlarge the effect of a shorter patching time for higher concentration. All in all, IV estimation results strengthens our findings.

For another robustness check, we estimate model (1.1) without each of the control variables (Table 1.13 in Appendix). Not controlling for vulnerability characteristics ($Vulnerability\_severity$ by $Vulnerability\_type$ dummies) or software and vendor's characteristics ($Software\_age$, software vendor's dummies) does not qualitatively affect our main results.

To sum up, from this baseline model, we find that market concentration impacts

positively the responsiveness of web browser publishers in securing their product. 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 ($\delta s_i^*/\delta n < 0$).

1.7.2 The effect of being in a dominant position

Table 1.6 – Results for model (1.2), effect of *Big_mshare*

Big_mshare = 1 when the affected web browser's market share is:	≥ 0.50		≥ 0.40	
	PW1	PW2	PW1	PW2
Concentration	-28.48***	-28.90***	-29.76***	-30.70***
	(8.970)	(9.082)	(8.770)	(8.737)
Big_mshare	536.4***	520.9***	481.4***	474.6***
	(119.3)	(119.7)	(85.39)	(88.81)
Big_mshare × *Concentration*	81.18***	78.00***	71.08***	70.24***
	(18.71)	(18.60)	(12.07)	(12.53)
Vulnerability specific effects				
Vulnerability_severity	-9.981	-8.613	-12.17**	-10.23*
	(6.204)	(6.133)	(5.465)	(5.413)
Vulnerability_type FE	✓	✓	✓	✓
Software_age	0.744	0.416	-0.292	-0.313
	(0.927)	(1.047)	(0.913)	(1.024)
Vendor specific FE	✓	✓	✓	✓
Qyear	-2.913***	-2.630***	-3.096***	-2.869***
	(0.816)	(0.781)	(0.707)	(0.687)
Correction term for *Concentration*	29.35***	28.45***	25.02***	25.74***
	(9.881)	(10.37)	(9.097)	(9.421)
Correction term for *Big_mshare* × *Concentration*	-19.30***	-16.80***	-12.94**	-11.39*
	(6.104)	(6.079)	(5.683)	(6.023)
Observations	874	874	874	874
R^2	0.353	0.376	0.385	0.398

Note: Dependent variable is *Patching_time*. Weighted Linear Regressions. Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1

Table 1.6 reports estimations for model (1.2), which focuses on the change in the responsiveness of a publisher when the web browser publisher has a dominant position. In the two first columns, we report results for *Big_mshare* equal to 1 when the affected web browser's market share is more than 50% of the market; in the two other columns, estimation results for the threshold value of 40% are reported.

First, results for model (1.2) confirms the result obtained in the baseline model. Keeping everything else equal, we obtain qualitatively and quantitatively similar results for the effect of *Concentration* in the two models: a web browser publisher releases a security patch for about 29 days more quickly when there is one less competitor in the market.

Nevertheless, this "positive" effect is dependent to whether the affected web browser has a dominant position or not. First, both coefficients of Big_mshare and $Big\_mshare \times Concentration$ are positive, meaning that when a web browser is dominant, the vendor releases a security patches less rapidly than it would have released if it was not. Moreover, comparing the coefficients' magnitude when the threshold market share is ≥ 0.40 to ≥ 0.50, the higher the market share of the publisher, the larger is the negative effect of market dominance on the responsiveness of the vendor. This result corroborates Proposition (2) which suggests that the security quality chosen by a highly dominant firm decreases with respect to its market share ($\partial s_i^*/\partial m_i < 0$). Secondly, the interaction term $Big\_mshare \times Concentration$ presents a positive sign. That is, the positive effect of market concentration in reducing the patching time is weakened when the affected publisher has a dominant position. Specifically, for a given number of firms competing in the market, a web browser editor which has more than 50% of the market would release a patch 520 days later than if it was not in a dominant position. Note that we measure here the causal effect of market dominance on the responsiveness of the vendor, after having removed the effect related to reverse causality, which explains why we have such a big coefficient. Moreover, this negative effect is strengthened when we consider a higher threshold of market share for the Big_mshare dummy. More precisely, the effect of the interaction term is 1.14 times larger when we consider that being in a dominant position is having more than 50% of the market compared to 40%. These results support the theoretical result suggested in Proposition (3), which states that the larger the market share of the dominant firm, the more reduced is the positive effect of market concentration on the security level the firm provides ($\partial^2 s_i^*/\partial n \partial m_i < 0$).

All in all, our results show that less competition in the market leads to higher security quality provision by web browser publishers. However, when a web browser is dominant, we find that the negative effect of a significant dominance on the responsiveness of the software vendor is greater than the positive effect of having less competitors.

To recapitulate our identification strategy, first, we have compared our data set to a more comprehensive database (NVD) and we applied a probability weight for each observation according to (1) the number of vulnerabilities observed for each browser each year and (2) the sum of their severity. Weighting or not our estimations does not

qualitatively modify the results. Secondly, we used IV estimations to correct for the endogeneity of our explanatory variable. We used the fact that concentration in the web browser market is strongly correlated to market share ratio between Microsoft Windows and Mac OS in the operating system market. By instrumenting the endogenous variable, we obtain a greater positive effect of market concentration in reducing the patch release time. Third, we control for a number of effects that might influence the patch release time in order to correctly isolate the effect attributed to the competition intensity in the market. We control for vendor fixed effects, the characteristics of the vulnerability, the software age, and a time variable which represent effects correlated to the time such as the rise of mobile use and the increasing attention given to web security. Whether we control or not for all of these effects does not modify our main findings except for the time effect.

1.8 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 874 web browser vulnerabilities identified and patched over a period of ten years, we examined the effect of market concentration and being in a dominant position, on web browser publishers' 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. Through a theoretical model, we explain this result by a particularity of the market we study: because companies does not compete in price but 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 a firm to invest more in security(quality).

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, the software age or open source licensing also significantly impact the vendor's incentives to release a patch more quickly.

This study also provides policy makers with empirical evidence on whether competition acts in the same way on firms' security investment incentives in free software markets such as the web browser one, a case which is even more of interest as it is very common in today's digital markets.

1.9 Appendix

Table 1.7 – dep. var: *Vulnerability_severity*

*cwe*119	0.915***
	(0.158)
*cwe*94	1.895***
	(0.213)
*cwe*20	0.582**
	(0.256)
*cwe*189	1.494***
	(0.354)
*cwe*416	0.717*
	(0.374)
*cwe*399	1.741***
	(0.175)
*cwe*264	-1.287***
	(0.251)
*cwe*200	-4.054***
	(0.284)
Constant	7.390***
	(0.149)
Observations	874
R-squared	0.481

S.E in parentheses
*** p<0.01, ** p<0.05, * p<0.1

Table 1.8 – Model (1.1) using *MobilevsBroadband* as a control

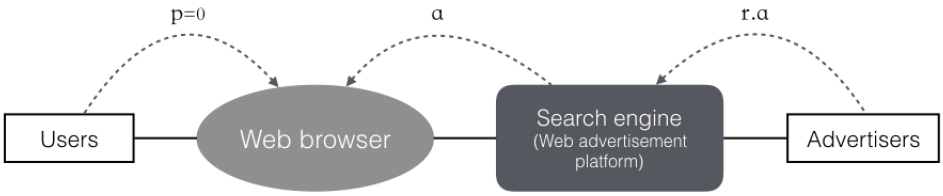
	-n		*HHI*	
	PW1 IV	PW2 IV	PW1 IV	PW2 IV
Concentration	-22.12***	-22.52**	-227.9***	-229.6**
	(8.153)	(9.138)	(83.43)	(91.76)
Vulnerability_severity	-16.53***	-15.26***	-16.40***	-15.32***
	(5.612)	(5.551)	(5.363)	(5.333)
Vulnerability_type FE	✓	✓	✓	✓
Software_age	2.442**	2.047**	2.459***	2.172**
	(0.990)	(1.014)	(0.924)	(0.959)
Vendor FE	✓	✓	✓	✓
MobilevsBroadband	-0.0368***	-0.0337***	-0.0304***	-0.0278***
	(0.00636)	(0.00621)	(0.00522)	(0.00522)
Observations	874	874	874	874
R^2	0.295	0.311	0.366	0.376

Note: Dependent variable is *Patching_time*. Weighted 2SLS regressions. Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1

Table 1.9 – 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 1.5 – 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 1.10 – Summary statistics

Variable	Weighted Mean	Lin. S.E.	Mean	S.D.	Min	Max
Patching_time	91.3	7.65	95.2	47.8	0	302
-n	7.20	0.0586	-7.31	0.99	-5	-9
HHI	0.317	0.0057	0.317	0.051	0.246	0.485
Big_mshare ($m.s. \geq 0.40$)	0.155	0.030	0.045	0.21	0	1
Big_mshare ($m.s. \geq 0.50$)	0.104	0.027	0.074	0.26	0	1
Vulnerability_severity	7.99	0.117	8.19	1.60	2.1	10
Software_age	7.13	0.604	7.05	5.19	0	22.9
Qyear	216.3	1.1	2014q3	9.7	2009q1	2018q2
Win_vs_MacOS	11.2	0.35	10.5	3.72	6.22	25.3

Number of observations: 874

Table 1.11 – 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. 8 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
$Qyear$	The date at which the vulnerability was reported to the editor (unit: date in quarter).
Win_vs_MacOS	Ratio between Windows and Mac OS's market share in the desktop operating system market

Table 1.12 – Results for model (1.1) with Poisson regressions

Using as *Concentration* measure:	*-n*			*HHI*		
	Poisson	PW1 Poisson	PW2 Poisson	Poisson	PW1 Poisson	PW2 Poisson
Concentration	-4.331***	-4.422	-4.726	-106.9***	-213.8***	-206.2**
	(0.379)	(3.972)	(5.083)	(6.847)	(81.25)	(89.78)
Vulnerability_severity	-2.235***	-14.48*	-13.02*	-2.788***	-14.59**	-13.06**
	(0.390)	(8.306)	(7.428)	(0.393)	(7.169)	(6.456)
Vulnerability_type FE	✓	✓	✓	✓	✓	✓
Software_age	1.384***	1.886*	1.546*	1.357***	1.851*	1.551*
	(0.0743)	(1.073)	(0.839)	(0.0744)	(1.111)	(0.859)
Apple	-14.62***	-32.76**	-27.34**	-14.05***	-33.03**	-27.19**
	(1.005)	(13.94)	(13.91)	(1.002)	(13.03)	(12.60)
Google	-19.87***	-20.94**	-14.98	-20.32***	-26.21***	-19.29*
	(1.569)	(10.43)	(11.05)	(1.562)	(9.858)	(9.994)
Mozilla	-35.85***	-50.12***	-46.80***	-35.52***	-51.17***	-46.62***
	(1.029)	(8.222)	(10.65)	(1.032)	(7.622)	(9.601)
Qyear	-1.794***	-3.347***	-3.175***	-1.679***	-3.492***	-3.154***
	(0.0551)	(0.492)	(0.390)	(0.0524)	(0.742)	(0.570)
Observations	874	874	874	874	874	874

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

Table 1.13 – Estimation results for model 1.1 with or without control variables

Using:						$-n$				
	Weighted IV (PW1)					Weighted IV (PW2)				
Concentration	-26.1***	-25.5***	-25.0***	-28.9***	-27.8**	-24.9***	-26.1***	-25.9***	-30.6***	-30.0***
	(8.60)	(9.53)	(9.64)	(10.5)	(11.3)	(8.90)	(9.64)	(9.72)	(10.8)	(11.6)
*Vendor*FE	✓					✓				
Software_age	✓	✓				✓	✓			
Vulnerability_type FE	✓	✓	✓			✓	✓	✓		
Vulnerability_severity	✓	✓	✓	✓		✓	✓	✓	✓	
Qyear	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Observations	874	874	874	874	874	874	874	874	874	874
R^2	0.243	0.186	0.188	0.103	0.061	0.276	0.226	0.227	0.128	0.091
Using:						HHI				
	Weighted IV (PW1)					Weighted IV (PW2)				
Concentration	-274***	-271***	-268***	-302***	-291**	-265***	-279***	-279***	-323***	-317***
	(87.6)	(99.7)	(101)	(112)	(120)	(91.5)	(101)	(103)	(115)	(122)
Vendor FE	✓					✓				
Software_age	✓	✓				✓	✓			
Vulnerability_type FE	✓	✓	✓			✓	✓	✓		
Vulnerability_severity	✓	✓	✓	✓		✓	✓	✓	✓	
Qyear	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Observations	874	874	874	874	874	874	874	874	874	874
R^2	0.350	0.279	0.279	0.216	0.165	0.362	0.310	0.310	0.238	0.197

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

Table 1.14 – Results for model (1.2), using OLS and Poisson regressions

Using as *Concentration*:	*-n*			
Big_mshare = 1 when the affected web browser's market share is:	≥ 0.50		≥ 0.40	
	OLS IV	Poisson IV	OLS IV	Poisson IV
Concentration	-22.22***	-21.99***	-27.42***	-27.24***
	(4.566)	(1.090)	(4.215)	(1.026)
Big_mshare	394.8***	319.9***	229.8***	208.2***
	(117.0)	(24.84)	(77.33)	(17.08)
Big_mshare × *Concentration*	65.96***	54.11***	34.59***	32.20***
	(22.27)	(4.814)	(13.28)	(2.980)
Vulnerability_severity	-1.012	-1.395***	-3.293*	-2.983***
	(2.121)	(0.479)	(1.855)	(0.431)
Vulnerability_type FE	✓	✓	✓	✓
Software_age	0.823**	0.741***	0.644*	0.598***
	(0.364)	(0.0805)	(0.352)	(0.0782)
Vendor FE	✓	✓	✓	✓
Qyear	-1.610***	-1.719***	-2.048***	-2.085***
	(0.333)	(0.0752)	(0.283)	(0.0655)
Correction term for *Big_mshare*	18.41***	17.58***	23.33***	22.84***
	(5.068)	(1.199	(4.671)	(1.127)
Correction term for *Big_mshare* × *Concentration*	-13.27	-6.612***	3.186	3.608**
	(9.446)	(1.951)	(6.495)	(1.438)
Observations	874	874	874	874

Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1
For Poisson regression average marginal effects are reported.

Table 1.15 – Results for model (1.2) with or without control variables

Using as *Concentration*:	-*n*				
Big_mshare = 1 when the affected web browser's market share is:	≥ 0.50				
Concentration	-27.45***	-31.05***	-30.64***	-32.94***	-31.32***
	(8.566)	(9.349)	(9.312)	(10.75)	(11.55)
Big_mshare	249.2*	360.1**	361.6**	263.2*	296.5**
	(145.8)	(145.8)	(145.2)	(136.4)	(143.5)
Big_mshare × *Concentration*	37.02*	50.25**	50.53**	37.40*	42.86**
	(22.27)	(21.96)	(21.88)	(20.40)	(21.54)
Software_age	✓	✓			
Vulnerability_type FE	✓	✓	✓		
Vulnerability_severity	✓	✓	✓	✓	
Qyear	-3.813***	-3.448***	-3.408***	-4.143***	-3.184***
	(0.874)	(0.839)	(0.833)	(1.135)	(0.995)
Correction term for *Concentration*	31.40***	31.64***	31.16***	32.02***	30.72**
	(9.701)	(9.857)	(9.620)	(11.50)	(12.10)
Correction term	-7.394	-5.534	-5.731	-3.088	-4.852
for *Big_mshare* × *Concentration*	(7.224)	(6.916)	(6.929)	(6.517)	(6.899)
Observations	874	874	874	874	874
R^2	0.377	0.331	0.331	0.249	0.203

Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1
All columns report Weighted IV estimation (PW1).

Chapter Two: Hackers' self-selection in crowdsourced bug bounty programs[*]

Abstract

A bug bounty program, also called Vulnerability Research Program (VRP), is a form of crowdsourcing increasingly employed by modern companies to improve their system's security. It consists in offering monetary rewards to individuals that find new security flaws in a software or a system.

One of the key challenges in the design of such contests is to attract enough participants while limiting the low quality participations. In this paper, we study how hackers' perception of the uncertainty to obtain a reward, determined by the level of information a contest provides about the contractual terms, affects the outcome of the contest both quantitatively (the number of participations) and qualitatively (participants' quality). Specifically, we examine how a hacker's choice to participate to a VRP depends on this level of information.

Using an unbalanced panel data set on 156 bug bounty programs run on a well-known bug bounty platform, we find that a more detailed contest policy and in particular more information about the compensation scheme attracts a greater number of participants. On the contrary, providing less detail induces less participation but attracts more performant and more experienced hackers. Hackers self-select whether to participate in a VRP according to the level of information about the contest's contractual terms, which leads to a trade-off between inducing more participation and attracting more valuable participants.

Keywords: bug bounty program, vulnerability research program, innovation contest, contract-related incentives, self-selection effect, hackers.

[*]I thank my thesis advisor Marc Bourreau for his support. I also thank Maya Bacache, Christine Zulehner, and the seminar audiences at 3EN 2018, EARIE 2018, DIF Lyon 2018, ZEW summer Workshop 2018, and at the TSE Digital Seminar for their useful comments on a previous version of the paper.

2.1 Introduction

Five years ago, a 17-year-old teenager in India discovered a serious vulnerability in several airline booking systems that allowed to get free plane tickets. Despite his effort to reach the airline companies and to alert them about the flaw, only one company took him seriously and reacted.[1] This is less true nowadays: an increasing number of companies seek to collaborate with benign vulnerability identifiers – the so-called white-hat hackers – to improve their systems' security. Methodical approaches have been developed to work with independent researchers, such as offering incentives in the form of a "bounty". Organizations started running *Vulnerability Research Programs* (VRPs) – also commonly called *Bug Bounty Programs* – which give monetary compensation to crowdsourced resources in exchange of information about vulnerabilities. Along with it, web platforms called "Bug bounty platforms" have emerged, hosting and managing these VRPs as a third party (E.g. HackerOne, BugCrowd, Yeswehack).

Launching a VRP has become a modern way to improve software and systems' security and is becoming accepted as a normal part of the software development lifecycle. However, managing a successful VRP is not an easy task. One of the key challenges companies face is to make the right balance between attracting enough participants and setting sufficiently high standards for participation in order to limit the proportion of low-value participations.[2] Indeed, a VRP intends to benefit from the diversity of participants, thus it is important to let a large pool of individuals to participate. Yet, each participation induces a cost, as it requires dedicated resources to sort the relevant participations out of the invalid ones and to communicate with participants.

One method applied by bug bounty platforms to reduce the rate of invalid participations is to allow only individuals that have been sufficiently efficient in the past to participate.[3] Unfortunately, one can also lose potentially valuable participations by applying such restrictive policies (Zhao, Laszka, and Grossklags, 2017). Apart from setting such a *minimum quality standard*, it is the contest's policy – i.e., the terms and rules of the contest – that may shape the outcome of the contest.

[1] Source: https://medium.com/@kanishksajnani/how-i-could-have-travelled-the-world-for-free-5bb10ac46ae5

[2] Source: https://www.hackerone.com/blog/signal-requirements

[3] For instance, HackerOne allows hackers to submit a report only when they meet a given level of valid to invalid submission ratio

The policy of a crowdsourced contest such as VRPs is comparable to an employment contract, as it defines the contractual relationship between the contest owner and participants, especially by specifying the compensation scheme the contest offers to participants and what it expects in exchange. The relationship between workers' performance and the attributes of an employment contract have been largely studied by economists. The literature distinguishes at least two types of effects. First, a compensation scheme can affect a worker's performance – for instance, through a pay-for-performance scheme – by inducing a certain level of effort from the worker, which may be referred to as *effort effects*. Secondly, it can have an effect on personal attributes. For example, a number of empirical works show that more productive workers systematically prefer a variable-pay to fixed-pay schemes (e.g., Dohmen and Falk, 2011). The possibility that agents with different individual characteristics feel attracted by different pay schemes and therefore self-select into particular forms of contracts may be referred to as *self-selection effects*.

In the same way as an employment contract, the characteristics of a contest's policy may affect both the level of effort provided by participants and the type of individuals that choose to participate in the contest. In this paper, we focus on the second aspect, that is, how the attributes of a VRP affects the decision of an individual to participate. In particular, we are interested in how the completeness of the contract offered by a VRP affects a hacker's choice to participate. By completeness of the contract, we mean how much information the VRP's policy provides about the compensation scheme and about what it expects as an outcome.

Crowdsourced innovation contests like VRPs present several important specificities that make it difficult to derive the answer to our question directly from the case of a standard employment contract. First, the number of participants – the number of brains that work on a problem – is an important factor that defines the effectiveness of a crowdsourced innovation contest (Terwiesch and Xu, 2008; Boudreau, Lacetera, and Lakhani, 2011). A contest policy should be thus applicable and attractive enough to a large panel of individuals. At the same time, it has to be accurate enough to provide adequate incentives and make the research process efficient. Secondly, in VRPs, participants are asked to find new security flaws that were unknown before. That is, they are asked to find an innovative way to penetrate a system rather than to carry out

predefined tasks like it is the case in standard crowdsourcing platforms (e.g., Amazon Mechanical Turk). Certain types of individuals may be more qualified to innovate. Solvers who have deep knowledge and experience in the problem domain could be most effective. Or, on the contrary, technical and social marginality could be an advantage to successfully solve the problem (Jeppesen and Lakhani, 2010). All in all, the success of an innovation contest largely depends on how to sort and to attract the right kind of solvers. Third, a VRP offers only a variable-pay: participants are compensated only if their contributions are relevant enough, regardless of the effort they have actually provided. Variable-pay schemes are likely to attract more productive workers than fixed-pay schemes, but they encompass also a sorting effect on other attributes such as the relative self-assessment or risk preference (Dohmen and Falk, 2011) that might alter the effectiveness of a contest.

The purpose of this study is to examine how the completeness of the contract proposed by a VRP affects its effectiveness. As the goal for a bug bounty program is to find as many relevant vulnerabilities as possible (i.e., it is interested in maximizing the sum across all outcomes opposed to a one-prize contest in which the goal is to maximize the value of the highest outcome), we examine the effect both on the quantity (the number of participations) and the quality of the participants.

Our analysis is based on publicly available data from the web platform HackerOne. We use an unbalanced panel data set on 156 bug bounty programs run from January 2015 to February 2019. VRPs on the platform we analyze are free to choose the level of information they provide about the contractual terms through a written policy. They can provide more or less detail about their compensation scheme, from fully specifying the payouts for each task to having a large degree of discretion about the monetary rewards and the targeted scope. They can also modify their policy over time. We consider that the level of information provided in the written policy reflects the degree of completeness of the contract the contest offers. We find that the more precise and detailed the policy is, the more participants it attracts. However, it also attracts participants with more heterogenous performance and it reduces the average quality of participants. On the contrary, leaving more uncertainty about the monetary rewards and the targeted scope generates fewer, more homogenous participants, but with higher quality in average.

This paper proceeds as follows. Section 2.2 reviews the relevant literature, Section 2.3 develops our hypothesis for analysis and Section 2.4 describes the data and estimation strategies. Section 2.5 reports the results, and Section 2.6 concludes.

2.2 Related literature

Our paper is closely related to three streams of research. The first one is on the economics of information security. The literature on the economics of information security aims at studying the potential market failures causing information systems insecurity. Vulnerability discovery is one of the topics at the heart of this field. We contribute empirically to this literature, first by studying a marketplace – a market for software vulnerabilities, where hackers sell vulnerability information to software vendors and companies – that has been barely studied for now because of its novelty, and secondly by studying how the design of a bug bounty program affects the contribution of the individual researchers.

Among the few papers that focus on VRPs, Finifter, Akhawe, and Wagner (2013) analyze two programs run by big pioneers in the vulnerability research community (Google and Mozilla) and examine whether running a VRP is economically profitable for a firm. Zhao et al. (2017) develop an analytical framework that compares different policies that aims at reducing the number of invalid reports.[4] To our knowledge, there are only two empirical studies that analyze data from bug bounty platforms.[5] Zhao, Grossklags, and Liu (2015) compare the currently biggest bug bounty platform HackerOne (run by a US-based company) to Wooyun, a well-known Chinese bug disclosure platform. They compare the trend in the discovered vulnerabilities in the two platforms, the different reward structures VRPs offer, and how offering monetary incentives attract more participants to a VRP. Maillart, Zhao, Grossklags, and Chuang (2017) also use data from HackerOne and show that the number of participations in a VRP is considerably reduced over its duration and that hackers strategically switch to new programs when new programs become available.

Along with these two papers, our paper is among the few ones that provides an

[4]They suggest that in VRPs, a penalty system for the invalid reports is more efficient than applying a minimum quality standard. Their model also shows that an increasing number of participants may decrease both the organization's and the hackers' utilities.

[5]See definition and description of a bug bounty platform in Subsection 2.4.1

empirical analysis about bug bounty platforms. Besides the principal characteristics of VRPs already examined by existing papers – such as the effects related to monetary incentives or the decreasing probability to find new vulnerabilities –, we identify an important mechanism that affects the effectiveness of a VRP. We are the first one that focuses on how the amount of information provided by a VRP about its contractual terms affects its effectiveness. We assimilate a VRP's policy to an employment contract proposed by a firm to workers and we study how the perception of the uncertainty to obtain a reward affects a worker's choice to participate in the contest. Our data set is also unique, in that it is a recent and large panel data set on VRPs run by diverse types of organizations while they are managed on a single platform. As we have a panel data set, we are able to account for the different fixed effects and identify in a robust way the effect attributed to a change in a VRP's policy. Moreover, we use both quantitative (the number of participations) and qualitative (participants' average quality and variance) information that defines the outcome of a contest.

Our paper is also related to the literature on innovation contests and tournaments. VRPs are a type of contest in which the organizer commits to reward the participants according to the rules and terms it defined, and participants spend resources in order to win the rewards. For each new security flaw, only the first finder is rewarded. It is thus close to an innovation contest in that the goal of the contest is to find a new idea – an innovative way to penetrate the system and secure it. Economists have studied from various angles the optimal design of contests, mainly about how to allocate the prizes (Archak and Sundararajan, 2009; Liu, Yang, Adamic, and Chen, 2014) and whether free entry or restricted numbers of participants yield to better outcomes (Terwiesch and Xu, 2008; Boudreau et al., 2011).

As Boudreau et al. (2011), we are interested in how the degree of uncertainty faced by participants affects the outcome of the contest. However, our scope and approach differ from them in several aspects. First, we study a type of contest increasingly used with the rise of crowdsourcing but barely studied for now. As mentioned earlier, in contests like VRPs, the goal is to maximize the sum of the outcomes, while "traditional" innovation contests like those launched on the well-known web platform InnoCentive aims at selecting one best solution.[6] Secondly, in Boudreau et al. (2011), the degree of

[6] Boudreau et al. (2011) use data from the web platform InnoCentive.

uncertainty is measured by the number of problem domains on which a given solution draws. They focus on the fact that participants exert less effort when they face more uncertainty to solve the problem. In our case, the uncertainty comes from the level of information provided by the contest and we are interested in how the preference for uncertainty attracts a given type of participants. Lastly, the results we obtain are different from the findings of Boudreau et al. (2011). In our work, we find that uncertainty attracts more performant participants, which has a positive effect on the overall outcome, while in their case, it is the number of participants that compensate the reduced effort of each individual due to problem uncertainty.

We also build our analysis on results from the rich body of literature on employment contracts. Specifically, in our paper, we are interested in the incentive schemes used by firms to attract specific types of workers, namely the self-selection effect as defined in Salop and Salop (1976) or in Chow (1983). Analytical works show that individuals with higher skills are more likely to choose a performance-based pay schemes than low-skill workers (e.g., Salop and Salop, 1976; Demski and Feltham, 1978; Lazear, 2000b; Jensen, 2003). The basic idea is that a worker evaluates the match between his self-perceived personal attributes and the perceived attributes of available employment contracts and selects the contract that maximizes his expected utility. This theory is supported by a number of empirical papers. Most of them are laboratory experiments (Chow, 1983; Waller and Chow, 1985; Cadsby, Song, and Tapon, 2007; Eriksson and Villeval, 2008; Dohmen and Falk, 2011), except from a field experiment by Fehrenbacher, Kaplan, and Pedell (2017) and studies of Lazear (2000a,b) based on a large data set on an auto glass company's workforce.

The originality of our work also comes from the fact that we investigate the mechanism of a crowdsourcing contest by referring to researches applied to a standard employment framework. In particular, our focus is on participants' self selection, while studies on contests are more concerned with the relationship between the design of the contest and the effort exerted by participants.

To our knowledge, Eriksson and Villeval (2008) is the unique paper that studied the self-selection effect in the context of tournaments. In a laboratory experiment, they show that when workers are allowed to choose between a performance-pay and a tournament, there is a considerable reduction in the variance of effort among contestants in the

tournament. They suggest that this is due to the fact that subjects self-select their payment scheme according to the degree of risk aversion. We also rely on their findings and see whether their arguments can be applied to a more general case where the degree of uncertainty perceived by the participants differs by the level of information the contest provides about the contractual terms. Our work confirms the findings of Eriksson and Villeval (2008) – which is based on a laboratory experiments on 120 students – in a more robust way, using a "natural setting" from 156 contests involving 184 participants in average. Additionally, most researches that study the self-selection effect, including Eriksson and Villeval (2008), compare variable-pay schemes to "less variable" pay schemes. Our approach is unique, as the uncertainty of being compensated is reflected through several measures that reflect the completeness of a contract: the amount of information provided in the policy, whether the VRP gives detailed information about the rewards, and about the scope. This allows us to have a greater granularity in the degree of uncertainty a contract represents.

2.3 Hypothesis development

The relationship between the number of participants in a contest and its efficiency have received considerable attention in the literature on tournaments and innovation contests. The literature identifies two opposing effects related to the number of contestants. The first one is on how the competition between participants affects the amount of effort exerted by each participant. Related to this effect, analytical works (e.g, Fullerton and McAfee, 1999; Moldovanu and Sela, 2001; Terwiesch and Xu, 2008) suggest that having many contributors working on a single issue leads to a lower equilibrium effort for each contributor because they face a greater risk of not being rewarded for the effort they are exerting. Empirical works have also provided evidence of an effort-reducing impact of an increased number of participants (Casas-Arce and Martínez-Jerez, 2009; Garcia and Tor, 2009).

The second effect is on the probability to find a solution by increasing the number of "brains" working on a subject. Terwiesch and Xu (2008) argue that a larger number of participants is not necessarily inefficient because it brings a more diverse set of solutions, which can outweight the negative effect from the underinvestment of each

individuals. Boudreau et al. (2011) empirically show that in the case of problems with high-uncertainty, adding additional participants increases the overall contest performance. In fact, this second effect – the positive effect of having a large number of solvers – is particularly applicable to innovation contests, in which the discovery of a solution and thus obtaining a reward presents a high degree of uncertainty.

Regarding the impact of uncertainty on the decision of an individual to participate, Eriksson and Villeval (2008) suggest that when participants are free to choose whether to participate or not in a contest, the uncertainty of success sorts the participants before they enter the contest. The idea is that facing uncertainty, some individuals would choose the minimum effort, that is, they drop out of competition, securing the loser's prize without bearing any cost. In sum, if there is too much uncertainty to win a reward or it is perceived as is by potential participants, the contest may lose some contributors that would have otherwise exerted an effort, even small.

Based on this discussion, we predict the following about the relationship between the degree of completeness of a VRP's policy and the number of participants.

Hypothesis 1. *A VRP that provides more detailed information about the contractual terms, i.e., less uncertainty about winning a reward, attracts a larger number of participants.*

According to Terwiesch and Xu (2008) and Boudreau et al. (2011), the uncertainty about getting rewarded may negatively affect the overall outcome as it reduces the number of participants that work on the problem. However, Eriksson and Villeval (2008) argue that the degree of uncertainty also sorts the type of participants that actually participate to the contest, hence leads to an ex-ante selection process. They show that when workers are given the possiblity to choose the degree of uncertainty of their payment scheme, they are sorted by their attributes. This results in a pool of participants with less variance in their performance in tournaments (contests), compared to piece-rate payment schemes.

Based on this reasoning, we predict the following:

Hypothesis 2. *Providing less details about the contractual terms attract participants more homogeneous in their performance.*

Finally, what Eriksson and Villeval (2008) argue is that when we give the possibility

to choose whether to participate or not in a contest, subjects who have a higher probability to win choose to enter the contest, which actually have a positive effect on the overall efficiency of tournaments.

More generally, the main idea suggested by studies on the self-selection effect in employment is that the ex-ante sorting effects contribute significantly to output difference between different incentive systems. For instance, Dohmen and Falk (2011) show in a laboratory experiment that the output in all variable-payment schemes (piece rate, tournament, or revenue-sharing schemes) is higher than the results under the fixed-wage regime and that this output difference is mainly attributable to productivity sorting. That is, when facing the alternative between more or less uncertain payments, more productive workers systematically prefer a payment scheme which is linked to their own performance even though it is more risky. Additionally, in payment schemes like tournaments in which the compensation depends on relative performance, they show that relative self-assessment plays an important role in sorting into tournaments. Considering that hackers who have participated more numerous times in other VRPs in the past (i.e., that have more experience in the platform) are more likely to have a better relative self-assessment, we hypothesize the following:

Hypothesis 3. *A VRP that provides less information about the contractual terms sorts the participants into more performant and more experienced participants.*

2.4 Data and empirical framework

2.4.1 Data

For the purpose of our analysis, we have collected publicly available data from HackerOne. HackerOne is a well-known web platform, created in November 2013, which intermediates companies that want to run a Vulnerability Research Programs (VRPs) and white-hat hackers who want to participate in such programs.[7] HackerOne is

[7]Bug bounty platforms take full advantage of their two-sidedness. They benefit from cross-sided network effects between VRPs and hackers: firms launch their VRP on the platform, because they can get access to a large pool of hackers, while hackers benefit from accessing a large number of contests on a single platform, which additionally allows them to manage a "hacker" profile that cumulates experiences from the different VRPs run on the platform. As it is common in these type of markets, the platform uses an asymmetric pricing strategy, where hackers get free access to the platform, while companies pay a usage fee. Besides, hackers can multi-home while companies usually launch their VRPs on a single platform, as it is costly for them to manage multiple VRPs using distinct platforms.

currently the most dominant bug bounty platform on the market. Bug bounty platforms such as HackerOne are different from innovation crowdsourcing platforms such as InnoCentive, as VRPs do not aim at choosing one winner that suggests the best solution but look for as many valuable contributions as they can get. That is, in VRPs, all the relevant submissions are rewarded. VRPs are also different from crowdsourcing platforms such as Amazon Mechanical Turk, in which workers are usually asked to carry out very simple tasks.

In order to launch a VRP, the organizing company first needs to define and publish its policy on the platform. The policy provides information about the contractual terms of the contest: the vulnerability types the VRP is looking for, the scope it is targeting, the reward structure, a vulnerability disclosure agreement, etc. On HackerOne, it is upon each VRP to define its policy and the level of information it provides within this policy. Some VRPs provide minimal description about their expectations, while others develop detailed sections describing the targeted scope of the VRP, the requirements for being eligible for a bounty or the detailed reward structure and amounts. In Figures 2.1 in Appendix, we provide an example of a VRP on HackerOne that gives only some basic information about the compensation and what it is looking for, while Figure 2.2 is an example of a VRP providing a greater amount of information about the compensation and about what they expect in exchange. A VRP can also modify its policy after the program has been launched: it can add new details, change the reward conditions, add some bonus rewards, etc.

In our study, we only consider VRPs that offer monetary rewards, that are publicly accessible and that do not pre-select their participants. Any hacker that has an account on the platform can participate to these public VRPs. Participating to a contest consists in submitting a report about a vulnerability and fixing the reported vulnerability in collaboration with the coordinators of the VRP. The submitted report has to respect the rules and the guidelines defined by the VRP. After a report is submitted, the VRP evaluates whether the submission is relevant and how valuable the submitted report is

Regarding the usage fee, HackerOne requests a fee proportional to the amount of rewards a VRP pays to hackers (HackerOne requires around 20 % of each reward, which also include the taxes). The basic service a company can subscribe to is the access to the web platform and a customized web interface to manage its VRP. The platform also offers a range of services companies can subscribe, related to the management of a VRP: they can help for the triage of the submitted reports, manage the whole program, etc.

(i.e., how critical and how important the discovered vulnerability is). If the report is evaluated as not relevant or as a duplicate, then it is rejected. For reports that are considered as relevant, the VRP validates the submission and starts exchanging with the participant to fix the vulnerability. When the vulnerability is fixed, the report becomes "resolved" and the VRP offers a reward to the participant. Then finally, the submitted report is closed.

The data we have collected concerns 156 active VRPs on HackerOne during the period from January 2015 to February 2019. As VRPs' policies evolve over time, we were able to build a panel data set. It is an unbalanced panel data set, as all the VRPs are not always active during the studied period. We have collected information about each VRPs each month, including the number of valid submissions, the performance indicators of each hacker at the date their submissions were accepted, the number of words used in the policy, whether a VRP presents a dedicated section that describes the scope of the contest, and whether it details the reward structure. We also collected data on how old a program is – i.e., the number of months since it has been launched–, and whether the program is managed by HackerOne or by the company itself. We also count the total number of active VRPs each month on the platform. Additionally, we have completed this database with information about the companies that run the VRPs from several other sources (Wikipedia, Crunchbase, Owler.com). We use the number of employees as a proxy for the size of the company and we categorized the companies into 11 different industries. We also account for the origin of the companies (they come from 24 distinct countries) and whether they have been acquired by another firm during the studied period.

2.4.2 Empirical specification

Our objective is to analyze how the level of information given by the contest through its written policy affects the outcome of the contest. As the goal of a VRP is to maximise the sum of the values of all the participations, we consider two complementary aspects that defines the outcome of a VRP and that we are able to measure: the quantity of participations and the quality of the participants.

First, to analyze the effect of the level of information on the quantity of participations,

we use the following baseline specification:

$$Nb\_participation_{it} = \beta_0 + \beta_1 Information\_Level_{it} + \beta_2 Prog\_Age_{it} + FE_i + FE_t + \epsilon_{it}, \tag{2.1}$$

where, *Nb_participation*, our dependent variable, is the number of submissions received by VRP i validated at period t. Our explanatory variable of interest is $Information\_Level$, which measures how detailed the policy of VRP i is in period t. Three alternative variables are used as $Information\_Level$. The first one is *Nb_words*, which counts how many words are used in a given VRP's policy in a given period. The second variable is *Reward*, a dummy which identifies whether the VRP provides detailed information about the structure and the amount of the rewards at period t. The third variable we use for $Information\_Level$ is the dummy *Scope*, which identifies whether the VRP's policy includes a dedicated section that describes in detail the scope targeted by the contest.

According to Hypothesis 1, we expect to obtain a positive coefficient for β_1. That is, we expect that the more information a VRP provides about its policy, the less uncertain participants feels about winning a reward, and the more participations it generates. *Prog_Age* measures how long it has been since the VRP has been launched (in number of month). FE_i and FE_t represent respectively VRPs and time fixed effects. In this specification, as we account for program and time fixed effect, we do not account for variables at the program level that do not vary over time. Lastly, ϵ_{it} is an error term.

Next, we use the following alternative specification:

$$\begin{aligned} Nb\_participation_{it} = {} & \beta'_0 + \beta'_1 Information\_Level_{it} + \beta'_2 Prog\_Age_{it} + \beta'_{3a} Managed\_by\_HO_i \\ & + \beta'_{3b} Firm\_Size_i + \beta'_{3c} Platform\_Growth_t + Industry\_FE_i \\ & + Country\_FE_i + \epsilon_{it}. \end{aligned} \tag{2.2}$$

In this specification, we include a range of variables specific to VRPs instead of accounting for VRP specific fixed effects. On the platform HackerOne, each VRP is run by a distinct company.[8] Thus each VRP is associated to one company. We account

[8]Theoretically, a company can run more than one VRP but this is not the case in our data set.

for the type of industry the company belongs to ($Industry\_FE$), the country where the companies come from ($Country\_FE$), a dummy which identifies whether the VRP is fully managed by HackerOne ($Managed\_by\_HO$), for how long the VRP has been launched ($Prog\_Age$), and how big the company is ($Firm\_Size$). For the $Firm\_Size$, we employ the definition of small and medium-sized and big company used by the World Bank. Specifically, we classify the companies into 4 categories according to the number of employees they have during the observed period. $Platform\_Growth$ measures the number of active VRP at period t on the platform. It accounts for the time trend. Lastly, ϵ_{it} is an error term. As in the first specification, we expect β_1' to be positive according to Hypothesis 1.

To construct our dependent variable $Nb\_Participation$, we use the number of participations that have been evaluated as "valid" participations. In other words, we ignore the number participations that have actually occurred but that have been rejected by the VRP. This can be an issue for our specification as the number of valid number of participations could also account for some qualitative effect, as valid participations may come from participants with higher Signal score (See the paragraph below describing the qualitative performance indicators we use). In other words, the valid number of participation can be represented as the product of the number of actual participations (rejected participations included) and the probability of each participants to submit a valid report. We perform additional regressions using this definition for the dependent variable, in order to test the robustness of our principal results.

Moreover, our estimation results can be biased because we ignore that the number of participations in period $t-k$ ($k>0$) can affect the decision of a VRP to modify its policy in period t. In other words, we might ignore the potential simultaneity between the choice of the hackers to participate to a VRP (the supply) and the demand of the VRP. Regarding this issue, statistics from Zhao et al. (2017) show that invalid submissions in HackerOne are relatively constant over time.[9] We could thus consider that ignoring the rejected participations does not have an effect on our estimation. Nonetheless, we instrument our main regressor $Information\_Level$ in order to account for the potential reverse causality.

[9]In Zhao et al. (2017), the percentage of valid reports in public VPRs remains relatively stable around 25% of the total submission during the total observed period of 2 years.

Specifically, we use two variables as our instruments. The first one is a dummy which identifies whether the company that owns the VRP (and that finances it) is being *Acquired* by another company at period t. The acquisition of the company could cause a change in the VRP's demand and thus on its policy while it does not affect directly the decision of a hacker to participate or not in the VRP. We thus consider that the exclusion restriction is satisfied. The second instrument we use is $PSD2$, which accounts for the effect of the second European Directive on Payment Services on cybersecurity investments. This directive came into force in January 2018 and one of its major goals is to improve the security of online and mobile payments and cross-border payment services. The variable $PSD2$ identifies whether the core activity of the company that owns the VRP i concerns the online or mobile payment sector, whether it is a European company, and whether period t is after January 2018. We consider that $PSD2$ satisfies the exclusion restriction as it may have some direct impact on the investment decision of a VRP especially for companies that are regulated (and thus on the choice of the $Information\_Level$ of a VRP's policy) but not on hackers' participation.

In order to analyze the effect of the level of information on the quality of participants, we use a set of dependent variables that reflects the average quality of the participants ($Av\_Reputation$, Av_Signal, and Av_Impact) and the standard deviation of these average quality ($Sd\_Reputation$, Sd_Signal, and Sd_Impact).

HackerOne provides three different performance indicators that reflect the quality of a hacker over time, which are the *Reputation* score, the *Signal* score and the $Impact$ score. These are the indicators we use to build our dependent variables for the quality of the participants. The *Reputation* score is a measure of a hacker's level of experience on the platform. It is an aggregate score of its contribution since the hacker has entered the platform (since it has created an account on HackerOne). It takes account both the number of valid reports the hacker has submitted and how valuable these contributions are. The *Signal* score is the average relevance of the reports submitted by the hacker. The lower the *Signal* score is, the higher the probability that the hacker submits an invalid report is (an invalid report is a report that is considered as not relevant enough or that another participant has already submitted to the VRP). The $Impact$ score is a measure of the average severity of the vulnerabilities reported by the hacker. It

represents the average value of the relevant reports submitted by the hacker. We compute the mean value of each indicators for participants in each VRP each month to construct our dependent variables *Av_Reputation*, *Av_Signal*, and *Av_Impact*.

The same specification as for *Nb_Participation* (Specification 2.1) is used to examine the effect of the $Information\_Level$ on the average quality of the participants. That is:

$$Av\_Quality_{it} = \beta_0'' + \beta_1'' Information\_Level_{it} + \beta_2'' Prog\_Age_{it} + FE_i + FE_t + \epsilon_{it}, \tag{2.3}$$

where we use *Av_Reputation*, *Av_Signal*, and *Av_Impact* as a measure of the average quality (*Av_Quality*). Following Hypothesis 3, we expect that β_1'' is negative, i.e., the more detailed a VRP's policy is, the lower the participants' average level of experience and level of performance are.

Lastly, we compute the standard deviation of each quality indicators to define the last series of dependent variables (*Sd_Reputation*, *Sd_Signal*, and *Sd_Impact*). These variables allow us to examine the effect of the level of contractual information provided by a VRP on the variance of participants' quality. Again, the same specification is used (Specification 2.1). As Hypothesis 2 specifies, we expect the coefficient of our main explanatory variable β_1'' to be negative, that is, a less detailed policy attracts participants more homogenous in their level of experience and performance.

2.4.3 Summary statistics

Table 2.5 to 2.8 in Appendix provide a range of information about the data set we use. Table 2.5 provides a description of the variables and Table 2.6 reports the summary statistics. In Table 2.7, we estimate the mean value of each of our dependent variables (*Nb_Participation*, *Av_Reputation*, *Av_Signal*, and *Av_Impact*) according to the total duration of a VRP. Lastly, Table 2.8 reports a correlation matrix for a list of variables.

Our data set is composed by 156 VRPs. During the observed period, a VRP is run during 25 months in average with a large standard deviation of 27 months. The per-month number of participations varies considerably according to the VRP, with a large proportion of VRP-month pair with 0 participations (25 % of the observations are equal

to 0). It is for this reason that we use a Poisson regression with robust standard errors for the specifications using the number of participation as the dependent variable.[10]

A VRP receives an average of 184 valid reports during the total duration of the program and an average of 6.4 valid reports in a month. Overall, for a given VRP, the number of participations decreases over time. Interestingly, the number of participations is not correlated to the average amount of reward a VRP offers. We also observe that VRPs which are run during a longer period of time have a higher per-month participation rate. On the other hand, it is not because a VRP is run during a longer period of time that participants are more performant in average (See Table 2.7).

Regarding our main explanatory variables, on average, a VRP's policy is composed of 514 words with large disparity. Policies evolve over time, with an average of difference of 654 words between the shortest and the longest version over time. As to the variables *Reward* and *Scope*, we observe from the correlation matrix (See Table 2.8) that they have a positive relation with Nb_Words, i.e., a longer policy has more probability to include a reward and a scope section. However the three variables are not strongly correlated. Additionally, VRPs are more likely to include *Reward* and *Scope* sections when they get "older", but it is not because a VRP is run for a longer duration in total that it presents a more detailed policy since the beginning of its launch. Lastly, we observe that the number of participations and the average quality of participants are very weakly correlated even though we observe a positive relationship between them.

2.5 Results

2.5.1 Effect of the level of information of a VRP's policy on the number of participations

Table 2.1 reports the regression results for specification 2.1 and 2.2 using the number of participations as the dependent variable. In each pair of columns, we have reported the results using the three different measures for our main regressor of interest $Information\_Level$, namely Nb_Words, *Reward*, and *Scope*. The first column of each pair of columns reports the estimation results for Specification 2.1 and the second

[10] We consider that Poisson regressions with robust standard errors are more reliable than Negative Binomial regressions when accounting for Fixed Effects (Silva and Tenreyro, 2010).

Table 2.1 – Estimation results for the effect of *Information_Level* on the *Nb_Participation* - Regressions without IVs

	(1)	(2)	(3)	(4)	(5)	(6)
	Poisson FE	Poisson	Poisson FE	Poisson	Poisson FE	Poisson
Nb_Words	0.00499***	0.00463***				
	(7.65e-05)	(0.000355)				
Reward			3.329***	2.215***		
			(0.0826)	(0.416)		
Scope					1.100***	0.161*
					(0.0801)	(0.406)
Prog_Age	0.00695**	0.0347***	0.0296***	0.0522***	0.0461***	0.0684***
	(0.00272)	(0.0124)	(0.00272)	(0.0140)	(0.00270)	(0.0138)
Platform_Growth		-0.0694***		-0.0280***		-0.0184***
		(0.00344)		(0.00335)		(0.00337)
Firm_Size		2.920***		3.392***		3.717***
		(0.0590)		(0.0606)		(0.0616)
Managed_byHO		-0.305		0.106		-0.177
		(0.421)		(0.426)		(0.407)
VRP FE	✓		✓		✓	
Time FE	✓		✓		✓	
Industry dummies		✓		✓		✓
Country dummies		✓		✓		✓
Observations	4,174	3,192	4,174	3,192	4,174	3,192
Wald χ^2	392.75	631.04	182.66	388.58	160.92	329.25
Number of VRPs	156		156		156	

Note: Dependent variable is *Nb_Participation*. Poisson FE and Poisson regressions with robust standard errors. Coefficients are Average Marginal Effects. Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1. The number of observations in columns (2), (4), and (6) are reduced because we do not have the information for the *Firm_Size* for every VRPs.

one for Specification 2.2. As expected, the coefficients for *Nb_Words*, *Reward*, and *Scope* – i.e., β_1 and β_1' in Specification 2.1 and 2.2 – are all positive and statistically significant. That is, the higher the level of information of a VRP's policy is, the more participations it generates. For example, according to the reported average marginal effects in column (3) and (5), when a VRP provides detailed information about the rewards, the number of participation is increased by three more participations per month, while giving detailed information about the scope of the contest attracts one more participant per month. This also shows that providing a reward section has a stronger effect than providing a scope section.

Regarding other control variables, we observe that *Prog_Age* presents a positive coefficient, meaning that everything else kept fixed, the number of participations increases over the duration of a VRP. The magnitude of the effect is larger when we do not account for VRP's fixed effects. This is because in column (2), (4) and (6) *Prog_Age* also accounts for the attractiveness specific to each VRPs and as we have

seen in the summary statistics, VRPs that are launched during a longer period of time are likely to generate more participations on the whole. We also observe that the larger the number of active VRPs on the platform ($Platform\_Growth$), the less participations a VRP receives. This may be due to the fact that new hackers are not entering the platform as fast as new VRPs are created and the amount of effort a hacker dedicates to a single VRP is reduced with the number of contests that are launched on the platform. As to coefficients for $Firm\_Size$, they are positive and always significant, showing that VRP of larger firms generate a larger number of participations.

Note that in these regressions we use simple Poisson regressions that do not include any instruments to deal with the potential reverse causality between the number of participations and the level of information of a VRP's policy. The next table reports the estimation results using IV regressions.

Table 2.2 – Estimation results of the effect of $Information\_Level$ on $Nb\_Participation$ - Regressions with IVs.

	(1)	(2)	(3)
Nb_Words	0.00627***		
	(0.000448)		
$Reward$		3.477***	
		(0.383)	
$Scope$			0.823**
			(0.388)
$Prog\_Age$	-0.00635	0.0290**	0.0494***
	(0.0105)	(0.0114)	(0.0126)
Corr. term for $Acquired$	-0.000176	-9.113***	-13.39***
	(0.00132)	(2.455)	(4.329)
Corr. term for $PSD2$	-0.00379***	8.442***	14.87***
	(0.00135)	(2.467)	(4.412)
Time FE	✓	✓	✓
program FE	✓	✓	✓
Observations	4,177	4,177	4,177
Wald Chi^2	304.56	141.81	47.70
Fstat (excluded instr.)	179.06	125.66	45.40
Number of VRPs	156	156	156

Note: Dependent variable is $Nb\_Participation$. Poisson IV regressions with robust standard errors, using a control function approach. Coefficients are Average Marginal Effects. Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1

Table 2.2 reports the estimation results when we include the instrumental variables in our regressions, using Specification 2.1. As specified in Subsection 2.4.2, we use two instruments: $Acquired$ and $PSD2$. The estimation results for the first stage is reported in Table 2.9 in Appendix. First stage regressions show that the excluded instrument is

correlated with the endogenous variable and the Stock-Yogo test is satisfied.[11] Table 2.10 in Appendix shows that the over-identification restrictions are also valid, thus we consider our instruments as valid.

As in Table 2.1, we report the results for the three different measures for $Information\_Level$ in each pair of columns. We observe that the coefficients for our main explanatory variables remain all positive and statistically significant. Additionally, the magnitude of these coefficients are greater than in Table 2.1 for Nb_Words and $Reward$, meaning that the actual causal effect is greater than when we do not correct for the simultaneity between hackers' participations and the VRP's demand for participation. Interestingly, this does not hold for the case of *Scope* (Column (3)). That is, providing detailed information about the scope generates a greater number of participations but a part of the effect is due to the reverse causality. Nevertheless, this effect is smaller than the causal effect of *Scope* on the number of participations.

All in all, our results confirm Hypothesis 1, showing that more information about the VRP's contractual terms and especially providing more details about the reward generates more participations.

2.5.2 Effect of the level of information of a VRP's policy on the quality of the participants

Table 2.3 reports the results for estimations that examine the effect of the level of information of a VRP's policy on the quality of participants, using as dependent variables $Av\_Reputation$, Av_Signal, and Av_Impact.

Each triplet of columns in Table 2.3 report regression results using the three different measures for $Information\_Level$ (Nb_Words, $Reward$, and $Scope$) and for each columns of each triplet, we use the three different measures of participants' quality $Av\_Reputation$, Av_Signal, and Av_Impact. As a reminder, $Reputation$ represents the experience level of a participant, $Signal$ reflects the probability that the participant submits a valid solution and $Impact$ reflects the average severity of the vulnerabilities that are found by the hacker, i.e., how valuable its participation is in average. We use a Poisson regression for $Av\_Reputation$ while we use linear regressions for Av_Signal

[11] We have two instruments and F statistics for the excluded instruments are much more greater than 19.93 which is the size of nominal 5% Wald test for 2SLS.

Table 2.3 – Estimation results for the effect of $Information\_Level$ on participants' quality - dependent variables are normalized.

Dep. var. is	*Av_Reputation*	*Av_Signal*	*Av_Impact*
	(1)	(2)	(3)
Nb_Words	-2.05e-05***	-2.06e-05**	-1.16e-05
	(3.50e-06)	(9.83e-06)	(8.13e-06)
Prog_Age	-0.000544***	0.00474***	0.00138
	(0.000118)	(0.00125)	(0.000928)
Time FE	✓	✓	✓
program FE	✓	✓	✓
Observations	2,943	2,946	2,946
VRPs	156	156	156
Wald χ^2 or R^2	78.24	0.296	0.275
	(4)	(5)	(6)
Reward	-0.0127***	-0.0495***	-0.0266***
	(0.00350)	(0.0125)	(0.0102)
Prog_Age	-0.000657***	0.00469***	0.00137
	(0.000122)	(0.00123)	(0.000913)
Time FE	✓	✓	✓
program FE	✓	✓	✓
Observations	2,943	2,946	2,946
VRPs	156	156	156
Wald χ^2 or R^2	65.90	0.299	0.276
	(7)	(8)	(9)
Scope	-0.00316	-0.00163	0.00225
	(0.00327)	(0.0116)	(0.00913)
Prog_Age	-0.000740***	0.00524***	0.00174*
	(0.000119)	(0.00126)	(0.000917)
Time FE	✓	✓	✓
program FE	✓	✓	✓
Observations	2,943	2,946	2,946
VRPs	156	156	156
Wald χ^2 or R^2	44.96	0.295	0.275

Note: Dependent variables were normalized (value rescaled from 0 to 1) in order to compare the magnitudes of the effects. We use Poisson regression for *Av_Reputation*, OLS regressions for *Av_Signal* and *Av_Impact* scores. For all regressions, VRP and time fixed effects are included. Coefficients are AME for Poisson regression. Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1

and *Av_Impact*. We use a normalized value of our dependent variables (i.e., values are rescaled from 0 to 1), so that one can compare the effect of the explanatory variables on the different measures of participants' quality (See Table 2.11 in Appendix for the estimation results using the original values of the dependent variables).

As expected, the coefficients for $Information\_Level$ are negative, meaning that less information about the contractual terms, i.e., more uncertainty about getting rewarded, attracts higher quality participants on average. Specifically, hackers who are more experienced, who find more critical vulnerabilities and who makes less mistakes are likely to self-select into VRPs with a lower level of information about the contractual terms. We also observe that the effect of the amount of information provided in the written policy or detailing the amounts of rewards are statistically significant while the effect of detailing the scope is not always significant. On the other hand, the magnitude of the effect does not vary a lot for the different quality indicators.

While the dependent variables in Table 2.3 is the measure of the quality itself, we report in Table 2.4 the regression results for the effect of the $Information\_Level$ on the variance of participants' quality. In Table 2.4, each triplet of columns reports the regression results using as dependent variables the standard deviation of *Av_Reputation*, *Av_Signal* and *Av_Impact*. The coefficients for $Information\_Level$ are all positive, corroborating Hypothesis 2. That is, providing more information about the contract increases the variance of participants' performance and experience, while providing less information attracts participants that are more homogenous in their attributes. The coefficients are also always statistically significant except for the case where we use *Scope* as a measure of the level of information. We note that the coefficients for *Prog_age* are always negative, meaning that participants' quality becomes more homogenous over the duration of a VRP. Lastly, from Table 2.3 and 2.4, we also observe that detailing the VRP's scope does not have a statistically significant effect neither on the participants' performance (*Signal* and *Impact* scores), nor on their variances.

2.6 Conclusion

The objective of this paper was to see how the main communication tool for a VRP, its written policy, may affect its efficiency. We particularly focused on the level of

Table 2.4 – Estimation results for the effect of $Information\_Level$ on the variance of participants' quality.

Dep. var. is	*Sd_Reputation*	*Sd_Signal*	*Sd_Impact*
	(1)	(2)	(3)
Nb_Words	0.221***	0.000255***	0.00119***
	(0.00159)	(7.31e-05)	(0.000232)
Prog_Age	-1.363***	-0.0162*	-0.0619**
	(0.0542)	(0.00923)	(0.0252)
Time FE	✓	✓	✓
program FE	✓	✓	✓
Observations	3,057	3,057	3,057
VRPs	156	156	156
	(4)	(5)	(6)
Reward	72.32***	0.225**	1.138***
	(1.620)	(0.0922)	(0.311)
Prog_Age	0.296***	-0.0203**	-0.0801**
	(0.0533)	(0.00922)	(0.0311)
Time FE	✓	✓	✓
program FE	✓	✓	✓
Observations	3,057	3,057	3,057
VRPs	156	156	156
	(7)	(8)	(9)
Scope	269.0***	0.107	0.455
	(1.572)	(0.0851)	(0.287)
Prog_Age	-0.890***	-0.0203**	-0.0819**
	(0.0528)	(0.00942)	(0.0318)
Time FE	✓	✓	✓
program FE	✓	✓	✓
Observations	3,057	3,057	3,057
VRPs	156	156	156

Note: Dependent variable is the Standard Deviation of participants' quality (*Av_Reputation*, *Av_Signal*, *Av_Impact* scores) during the month. Poisson regression for *Sd_Reputation*, OLS regressions for *Sd_Signal* and *Sd_Impact* scores. For all regressions, VRP and time fixed effects are included. Coefficients are AME for Poisson regression. Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1

information provided about the contractual terms. We consider that the level of information determines the uncertainty perceived by the participants about getting compensated and we used three variables that represent different aspects of the level of information: the length of the VRP's policy, whether it provides detailed information about the reward, and about the scope of the contest.

Using a data set covering 4 years of activities, we find that providing more information about the contest generates more participations but also reduces the overall quality of participants and enlarges the variance of their quality. It is interesting to note that revealing more information about the rewards does not attract higher quality participants. On the contrary, leaving more uncertainty about the rewards and

what is expected in exchange attracts more homogenous, more performant and more experienced participants. Furthermore, monetary aspects like precising the reward structure seem to have a more significant self-selection effect than providing detailed information about the scope of the contest.

Our results suggest that in the context of tournaments such as a VRP, the possibility that agents with different characteristics feel attracted by different types of contract leads the VRP owner to a trade-off between having a larger number of participation but attracting less performant participants and attracting higher quality participants but generating fewer participations. This ex-ante self selection process is all the more important for contest like VRPs since it reduces the cost induced to manage the non-relevant participations.

2.7 Appendix

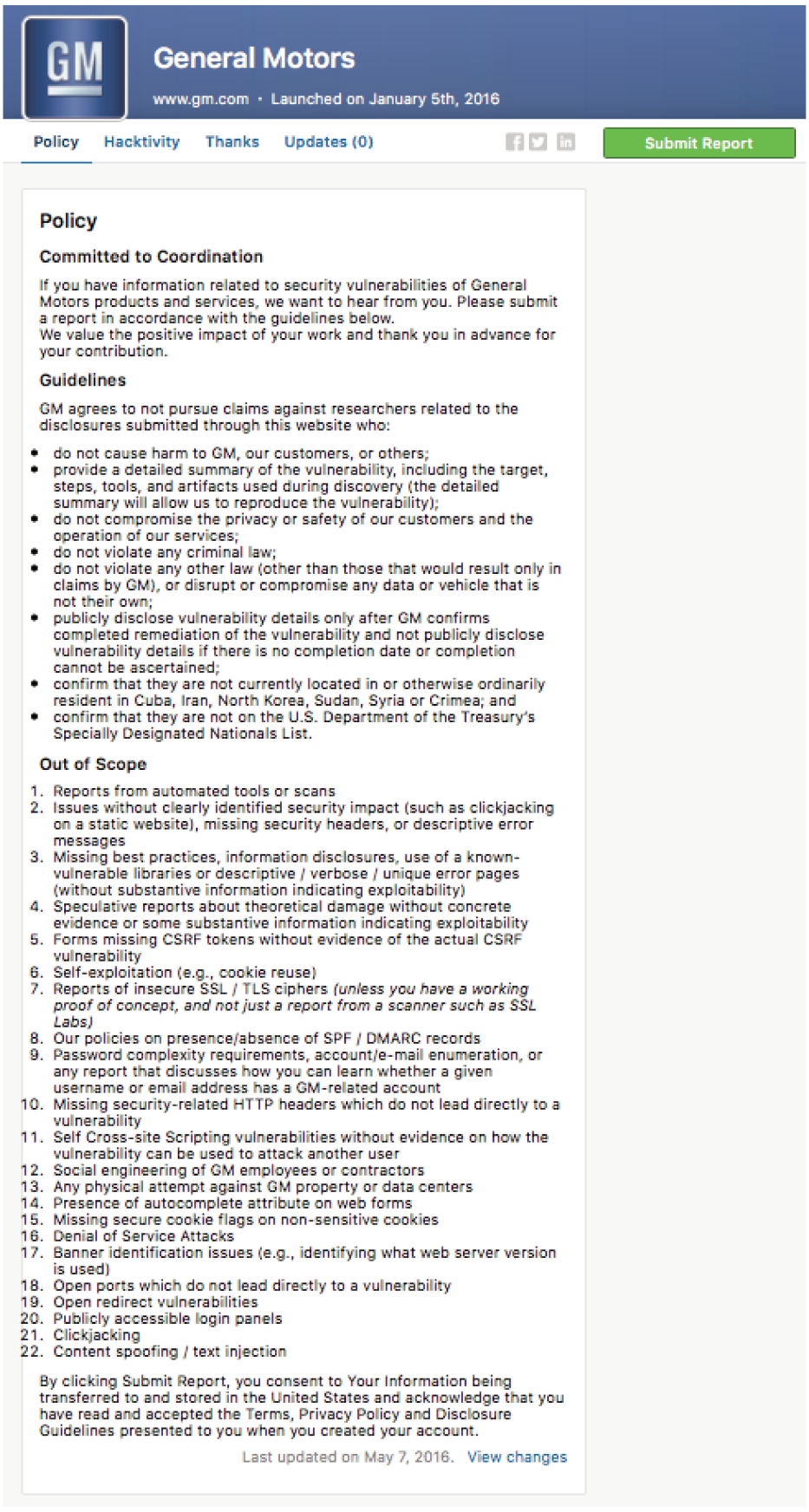
Table 2.5 – Description of variables

Variable	Description
Nb_Participations	The total number of valid reports the VRP processed during the month
Av_Reputation	The average Reputation score of the participants to a given VRP at a given month. The Reputation score is an aggregate score of a hackers' contribution in terms of vulnerability severity, relevancy, and quantity
Av_Signal	The average Signal score of the participants to a given VRP at a given month. The Signal score is the average relevance of the reports submitted by a hacker
Av_Impact	The average Impact score of the participants to a given VRP at a given month. Impact is a measure of the average severity of the vulnerabilities a hacker have reported.
Sd_Reputation	Standard deviation of *Av_Reputation*
Sd_Signal	Standard deviation of *Av_Signal*
Sd_Impact	Standard deviation of *Av_Impact*
Nb_Words	Number of words of the written policy at the observed period
Reward	A dummy which identifies whether the VRP provides detailed information about the structure and the amount of the rewards at period t
Scope	A dummy which identifies whether the policy includes a dedicated section that describes in detail the scope targeted by the contest.
Prog_Age	Number of month since the VRP has been launched
Firm_size	Categorization of the size of the company which owns the VRP by their number of employees (Small, Medium Size company, Big company)
Industry dummies	The type of industry the VRP's owner company belongs
Country dummies	Country of origin of the company that owns the VRP
Managed_by_HO	A dummy which identifies whether the VRP is fully managed by HackerOne
Platform_Growth	The number of active VRP on the platform at a given period
Acquired	A dummy which identifies whether the company that owns the VRP (and thus that finances it) is being *Acquired* by another company at period t
PSD2	A dummy which identifies whether the core activity of the company that owns the VRP i concerns the online or mobile payment sector, whether it is a European company, and whether period t is after January 2018

Table 2.6 – Summary statistics

Variable	Obs	Mean	Std. Dev.	Min	Max
Nb_Participation	4177	6.38	10.58	0	140
Av_Reputation	2943	2507.66	3079.62	8	36120
Av_Signal	2946	2.81	1.83	-4	7
Av_Impact	2946	14.35	5.67	0	38
Sd_Reputation	2386	2309.93	2542.78	0	25142.6
Sd_Signal	2391	1.88	.99	0	7.21
Sd_Impact	2391	5.65	3.91	0	25
Nb_Words	4177	513.91	475.74	0	2509
Reward	4177	.36	.48	0	1
Scope	4177	.43	.5	0	1
Prog_Age	4177	16.98	14.3	0	64
Managed_by_HO	4177	.245	.43	0	1
Firm_Size	3260	2.89	1.03	1	4
Platform_Growth	4177	66	15.51	19	89
Acquired	4177	.028	.17	0	1
PSD2	4177	.052	.22	0	1

Figure 2.1 – Example of VRP on HackerOne, providing minimal information about the contractual terms: General Motor's VRP



General Motors

www.gm.com · Launched on January 5th, 2016

Policy | Hacktivity | Thanks | Updates (0) | Submit Report

Policy

Committed to Coordination

If you have information related to security vulnerabilities of General Motors products and services, we want to hear from you. Please submit a report in accordance with the guidelines below.
We value the positive impact of your work and thank you in advance for your contribution.

Guidelines

GM agrees to not pursue claims against researchers related to the disclosures submitted through this website who:

- do not cause harm to GM, our customers, or others;
- provide a detailed summary of the vulnerability, including the target, steps, tools, and artifacts used during discovery (the detailed summary will allow us to reproduce the vulnerability);
- do not compromise the privacy or safety of our customers and the operation of our services;
- do not violate any criminal law;
- do not violate any other law (other than those that would result only in claims by GM), or disrupt or compromise any data or vehicle that is not their own;
- publicly disclose vulnerability details only after GM confirms completed remediation of the vulnerability and not publicly disclose vulnerability details if there is no completion date or completion cannot be ascertained;
- confirm that they are not currently located in or otherwise ordinarily resident in Cuba, Iran, North Korea, Sudan, Syria or Crimea; and
- confirm that they are not on the U.S. Department of the Treasury's Specially Designated Nationals List.

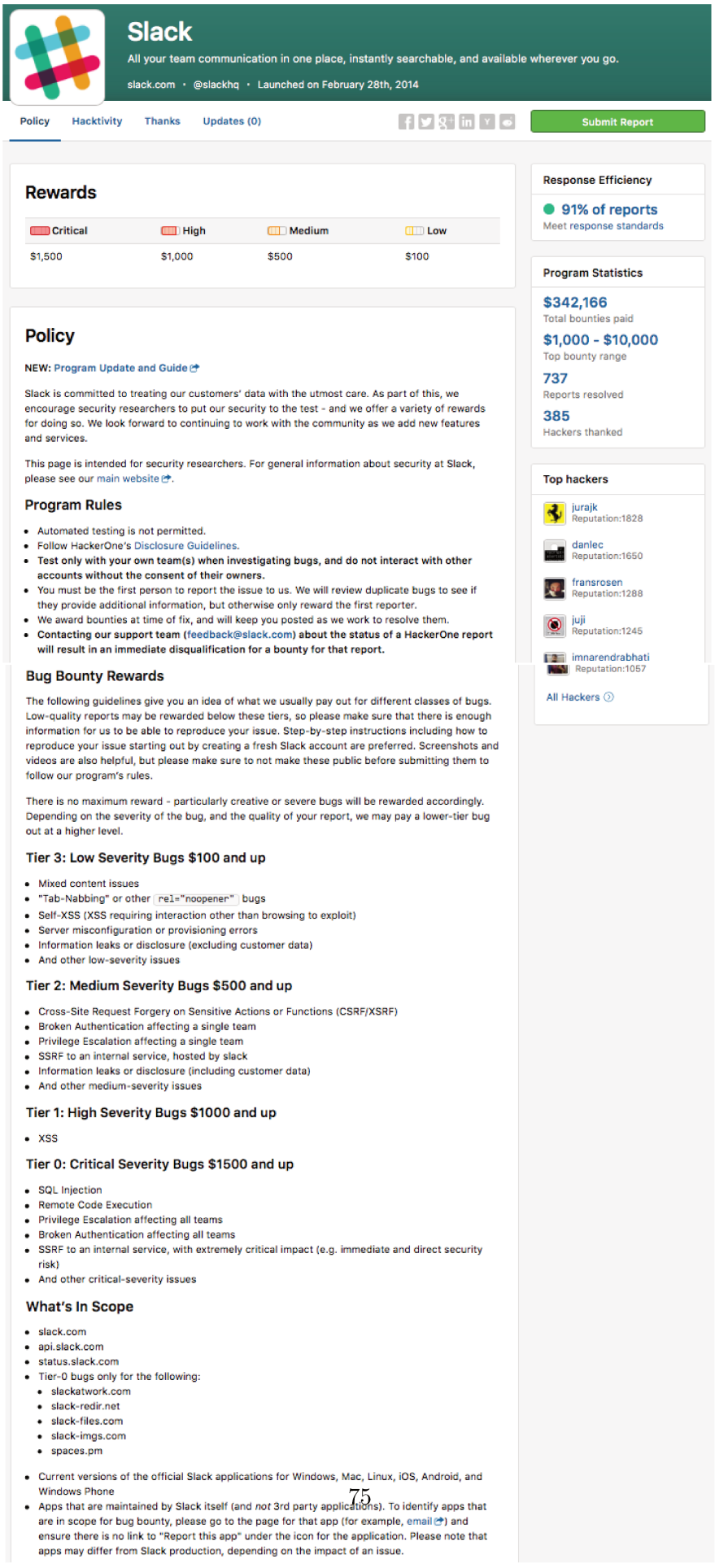
Out of Scope

1. Reports from automated tools or scans
2. Issues without clearly identified security impact (such as clickjacking on a static website), missing security headers, or descriptive error messages
3. Missing best practices, information disclosures, use of a known-vulnerable libraries or descriptive / verbose / unique error pages (without substantive information indicating exploitability)
4. Speculative reports about theoretical damage without concrete evidence or some substantive information indicating exploitability
5. Forms missing CSRF tokens without evidence of the actual CSRF vulnerability
6. Self-exploitation (e.g., cookie reuse)
7. Reports of insecure SSL / TLS ciphers *(unless you have a working proof of concept, and not just a report from a scanner such as SSL Labs)*
8. Our policies on presence/absence of SPF / DMARC records
9. Password complexity requirements, account/e-mail enumeration, or any report that discusses how you can learn whether a given username or email address has a GM-related account
10. Missing security-related HTTP headers which do not lead directly to a vulnerability
11. Self Cross-site Scripting vulnerabilities without evidence on how the vulnerability can be used to attack another user
12. Social engineering of GM employees or contractors
13. Any physical attempt against GM property or data centers
14. Presence of autocomplete attribute on web forms
15. Missing secure cookie flags on non-sensitive cookies
16. Denial of Service Attacks
17. Banner identification issues (e.g., identifying what web server version is used)
18. Open ports which do not lead directly to a vulnerability
19. Open redirect vulnerabilities
20. Publicly accessible login panels
21. Clickjacking
22. Content spoofing / text injection

By clicking Submit Report, you consent to Your Information being transferred to and stored in the United States and acknowledge that you have read and accepted the Terms, Privacy Policy and Disclosure Guidelines presented to you when you created your account.

Last updated on May 7, 2016. View changes

Figure 2.2 – Example of VRP on HackerOne, providing very detailed information about the contractual terms: Slack's VRP



Slack

All your team communication in one place, instantly searchable, and available wherever you go.

slack.com • @slackhq • Launched on February 28th, 2014

Policy | Hacktivity | Thanks | Updates (0) | Submit Report

Rewards

Critical	High	Medium	Low
$1,500	$1,000	$500	$100

Response Efficiency

91% of reports
Meet response standards

Program Statistics

$342,166
Total bounties paid

$1,000 - $10,000
Top bounty range

737
Reports resolved

385
Hackers thanked

Top hackers

jurajk Reputation:1828
danlec Reputation:1650
fransrosen Reputation:1288
juji Reputation:1245
imnarendrabhati Reputation:1057

All Hackers

Policy

NEW: Program Update and Guide

Slack is committed to treating our customers' data with the utmost care. As part of this, we encourage security researchers to put our security to the test - and we offer a variety of rewards for doing so. We look forward to continuing to work with the community as we add new features and services.

This page is intended for security researchers. For general information about security at Slack, please see our main website.

Program Rules

- Automated testing is not permitted.
- Follow HackerOne's Disclosure Guidelines.
- **Test only with your own team(s) when investigating bugs, and do not interact with other accounts without the consent of their owners.**
- You must be the first person to report the issue to us. We will review duplicate bugs to see if they provide additional information, but otherwise only reward the first reporter.
- We award bounties at time of fix, and will keep you posted as we work to resolve them.
- **Contacting our support team (feedback@slack.com) about the status of a HackerOne report will result in an immediate disqualification for a bounty for that report.**

Bug Bounty Rewards

The following guidelines give you an idea of what we usually pay out for different classes of bugs. Low-quality reports may be rewarded below these tiers, so please make sure that there is enough information for us to be able to reproduce your issue. Step-by-step instructions including how to reproduce your issue starting out by creating a fresh Slack account are preferred. Screenshots and videos are also helpful, but please make sure to not make these public before submitting them to follow our program's rules.

There is no maximum reward - particularly creative or severe bugs will be rewarded accordingly. Depending on the severity of the bug, and the quality of your report, we may pay a lower-tier bug out at a higher level.

Tier 3: Low Severity Bugs $100 and up

- Mixed content issues
- "Tab-Nabbing" or other `rel="noopener"` bugs
- Self-XSS (XSS requiring interaction other than browsing to exploit)
- Server misconfiguration or provisioning errors
- Information leaks or disclosure (excluding customer data)
- And other low-severity issues

Tier 2: Medium Severity Bugs $500 and up

- Cross-Site Request Forgery on Sensitive Actions or Functions (CSRF/XSRF)
- Broken Authentication affecting a single team
- Privilege Escalation affecting a single team
- SSRF to an internal service, hosted by slack
- Information leaks or disclosure (including customer data)
- And other medium-severity issues

Tier 1: High Severity Bugs $1000 and up

- XSS

Tier 0: Critical Severity Bugs $1500 and up

- SQL Injection
- Remote Code Execution
- Privilege Escalation affecting all teams
- Broken Authentication affecting all teams
- SSRF to an internal service, with extremely critical impact (e.g. immediate and direct security risk)
- And other critical-severity issues

What's In Scope

- slack.com
- api.slack.com
- status.slack.com
- Tier-0 bugs only for the following:
  - slackatwork.com
  - slack-redir.net
  - slack-files.com
  - slack-imgs.com
  - spaces.pm
- Current versions of the official Slack applications for Windows, Mac, Linux, iOS, Android, and Windows Phone
- Apps that are maintained by Slack itself (and *not* 3rd party applications). To identify apps that are in scope for bug bounty, please go to the page for that app (for example, email) and ensure there is no link to "Report this app" under the icon for the application. Please note that apps may differ from Slack production, depending on the impact of an issue.

Table 2.7 – Mean values of the dependent variables according to the duration of a VRP

Duration of a VRP (in months)	Mean value of *Nb_Participation*	Mean value of *Av_Reputation*	Mean value of *Av_Signal*	Mean value of *Av_Impact*
0 - 5	6.5	2588	3	13.1
5 - 15	4.7	2309	2.5	12.7
15 - 25	5.6	2776	2.8	14.4
25 - 35	4.5	2466	2.9	13.9
35 - 45	6.6	2475	2.7	14.5
45 - 55	7.2	2687	3.1	15.2
55 - 65	11.6	2296	2.8	15.3

Table 2.8 – Correlation matrix of the principal variables

	Nb_Participation	*Av_Reputation*	*Av_Signal*	*Av_Impact*	*Nb_Words*	*Reward*	*Scope*	*Prog_Age*	*Prog_Du.*
Nb_Participation	1								
Av_Reputation	0.04	1							
Av_Signal	0.15	0.31	1						
Av_Impact	0.09	0.25	0.49	1					
Nb_Words	0.24	-0.14	0.01	-0.1	1				
Reward	0.16	-0.1	-0.01	-0.06	0.56	1			
Scope	0.06	-0.04	0.02	-0.05	0.45	0.25	1		
Prog_Age	0.06	-0.13	5.0e÷3	-0.05	0.3	0.25	0.2	1	
Prog_Duration	0.12	0	0.03	0.12	0.1	0.16	0.05	0.67	1

Table 2.9 – First Stage regressions for Table 2.2

	(1)	(2)	(3)	(4)	(5)	(6)
Dep. var. is:	*Nb_Words*		*Reward*		*Scope*	
PSD2	1,605***		0.702***		-0.311***	
	(91.16)		(0.0778)		(0.0858)	
Acquired		478.2***		0.814***		0.638***
		(79.51)		(0.0650)		(0.0718)
Prog_Age	-19.99***	-21.95***	-0.00917***	-0.00799***	-0.0228***	-0.0200***
	(1.683)	(1.748)	(0.00144)	(0.00143)	(0.00158)	(0.00158)
VRP FE	✓	✓	✓	✓	✓	✓
Time FE	✓	✓	✓	✓	✓	✓
Observations	4,177	4,177	4,177	4,177	4,177	4,177
Number of VRPs	156	156	156	156	156	156

Note: Linear FE regressions. Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1.

Table 2.10 – Validity of over-identification restrictions

Included IVs:	*PSD2* & *Acquired*	Only *PSD2*	Only *Acquired*	*PSD2* & *Acquired*	Only *PSD2*	Only *Acquired*	*PSD2* & *Acquired*	Only *PSD2*	Only *Acquired*
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Nb_Words	0.00627***	0.00597***	0.00627***						
	(0.000448)	(0.000426)	(0.000448)						
Reward				3.477***	3.558***	3.385***			
				(0.383)	(0.381)	(0.380)			
Scope							0.823**	0.844**	0.751*
							(0.388)	(0.387)	(0.385)
Prog_Age	-0.00635	-0.00271	-0.00636	0.0290**	0.0274**	0.0290**	0.0494***	0.0484***	0.0493***
	(0.0105)	(0.0105)	(0.0105)	(0.0114)	(0.0114)	(0.0115)	(0.0126)	(0.0126)	(0.0126)
Corr. term for *PSD2*	-0.000176	-0.00353***		-9.113***	-1.135		-13.39***	1.150	
	(0.00132)	(0.000747)		(2.455)	(0.987)		(4.329)	(0.890)	
Corr. term for *Acquired*	-0.00379***		-0.00394***	8.442***		-0.284	14.87***		1.596*
	(0.00135)		(0.000738)	(2.467)		(1.014)	(4.412)		(0.904)
VRP FE	✓	✓	✓	✓	✓	✓	✓	✓	✓
Time FE	✓	✓	✓	✓	✓	✓	✓	✓	✓
Observations	4,177	4,177	4,177	4,177	4,177	4,177	4,177	4,177	4,177

Note: Dependent variable is *Nb_Participation*. Poisson IV regressions with robust standard errors, using a control function approach. Coefficients are Average Marginal Effects. Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1

Table 2.11 – Estimation results for the effect of *Information_Level* on participants' quality - dependent variables are not normalized

Dep. var. is	*Av_Reputation*	*Av_Signal*	*Av_Impact*
	(1)	(2)	(3)
Nb_Words	-0.741***	-0.000227**	-0.000442
	(0.126)	(0.000108)	(0.000309)
Prog_Age	-19.65***	0.0521***	0.0524
	(4.254)	(0.0138)	(0.0352)
Observations	2,943	2,946	2,946
VRPs	156	156	156
Wald χ^2 or R^2	78.24	0.296	0.275
	(4)	(5)	(6)
Reward	-457.8***	-0.545***	-1.012***
	(126.5)	(0.137)	(0.389)
Prog_Age	-23.71***	0.0516***	0.0519
	(4.405)	(0.0136)	(0.0347)
Observations	2,943	2,946	2,946
VRPs	156	156	156
Wald χ^2 or R^2	65.90	0.299	0.276
	(7)	(8)	(9)
Scope	-114.3	-0.0180	0.0856
	(118.2)	(0.128)	(0.347)
Prog_Age	-26.73***	0.0577***	0.0662*
	(4.297)	(0.0139)	(0.0349)
Observations	2,943	2,946	2,946
VRPs	156	156	156
Wald χ^2 or R^2	44.96	0.295	0.275

Note: We use Poisson regression for *Av_Reputation*, OLS regressions for *Av_Signal* and *Av_Impact* scores. For all regressions, VRP and time fixed effects are included. Coefficients are AME for Poisson regression. Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1

Chapter Three: Software vulnerability disclosure and third parties' involvement in security[*]

Abstract

Around the debate on software vulnerability disclosure, existing 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 disclosure and for the others. We find that the overall number of discovered vulnerabilities increases after the vulnerability disclosure. 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 third parties' action when analyzing the incentives to provide software security.

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

[*]I thank my thesis advisor Marc Bourreau for his patient guidance and support. I also thank Rainer Böhme, Ulrich Laitenberger, Patrick Legros, Julien Pénin, anonymous reviewers and audience at WEIS 2019, audiences at ACDD Strasbourg 2019, IOEA 2019, and 3EN 2019 for their useful comments on a previous version of the paper.

3.1 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, 2006b). On the other side, many 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 et al., 2010b). All of these studies consider that the disclosure of a vulnerability allows the attackers to exploit the disclosed information and in turn affects the users and the software vendors' behavior. Besides, a vulnerability disclosure may also impact the stakeholders' behavior without being specifically related to the disclosed vulnerability information. It may act as a signal, to third parties like consumers, investors, or security experts, that the actual security level of the affected software is lower than what it was considered until now. For instance, Telang and Wattal (2007) show that public vulnerability announcements lead to significant loss in the affected software vendor's market value. In our paper, we consider precisely this 'signaling effect' and study whether the disclosure of a specific vulnerability on a software gives an incentive to improve its overall security.

Specifically, we examine empirically how the public disclosure of a critical vulner-

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

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

ability with heavy media coverage affects the discovery of new vulnerabilities in the software that is subject to the disclosure. We study the case of three markets – by market, we mean a group of software that belong 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 generate significant attention from general media as well as from the security community, namely the web browser, 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 well-known 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 security effort exerted on the software affected by the disclosure compared to other unaffected software.

Theoretically, the answer to our question is not straightforward. First, regarding the effect on users, the vulnerability disclosure may reduce the perceived security quality of a software. If users are passive agents that do not contribute to the security level of the software, the disclosure of a vulnerability may just reduce the user demand. However if they can also choose to invest in security, they could be affected by the disclosure in a different way, depending on the available alternatives, the switching costs, and their investment capability in software security. The larger a business is – i.e., the larger and complex its information system is –, the less flexible it would be to switch from one software to another even though it realizes that the software it uses has a poor security quality. Companies who have large switching costs may prefer to actively collaborate with the software vendor and other third parties (security firms, individual security experts, public organizations, etc.) to improve the security of the software rather than waiting for the vendor to provide a better security. Secondly, the disclosure of a critical vulnerability may also act as an event that revises the belief – i.e., the subjectif probability – to find new vulnerabilities. Thus for actors that look for new opportunities to contribute to software security – such as security firms or individual researchers –, vulnerability disclosure may give an incentive to exert more

effort in discovering new vulnerabilities. However if the disclosed information is public and shared with everyone, the increased probability to compete with others can also deter them to exert an additional effort. All in all, the overall effect of vulnerability disclosure on the effort exerted by different actors to improve the affected software's security is uncertain.

Our analysis shows strong evidence that after the disclosure of a critical vulnerability, the vulnerability research activity on the software affected by the disclosure significantly increases compared to the control group of unaffected software. Interestingly, third parties are more affected by the disclosure than the software vendor itself. In particular, the number of vulnerabilities discovered by security firms and individual researchers increases significantly. This result is all the more important as our analysis shows that third parties 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 the potential contribution of third parties when studying software security.

The rest of the paper is structured as follows. In the next section, we review the relevant literature. Section 3.3 presents the identification strategy and the data. We present the estimation results in Section 3.4 and Section 3.5 concludes.

3.2 Related work

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. Various topics are handled in this literature, from modeling attack and defense (Varian, 2004; Bier et al., 2007; Bohme and Moore, 2010), liability policies (Kim et al., 2011; August and Tunca, 2011; Lam, 2016), risk sharing and coordination between vendor and users (August and Tunca, 2006; Cavusoglu et al., 2008; Kim et al., 2009), to product differentiation (August et al., 2014; Dey et al., 2014). In particular, some papers participate to the debate around whether promoting software vulnerability disclosure is socially desirable (Cavusoglu et al., 2007; Arora et al., 2008; Choi et al., 2010; Nizovtsev and Thursby, 2007). Their main finding is that, when vendors do not sufficiently internalize user loses, vulnerability disclosure provides an incentive for vendors to secure their product more quickly.[3]

[3]Besides, Choi et al. (2010) take also account the probability of an attack in relation with network effects and show that mandatory disclosure is not necessarily welfare improving.

Our paper tackles two aspects lacking in this literature. First, prior works mostly consider that users are passive agents, which mainly undertake damage control activities (patch installations, work-arounds) rather than actively engaging in preventive actions. However, 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. In our paper, users actively invest in software security by discovering and reporting vulnerabilities to the software vendor.

To our knowledge, only one paper considers the active involvement of users in security (Nizovtsev and Thursby, 2007). They consider the case of open source software where users can actively participate in finding and fixing vulnerabilities and show that the positive effect of vulnerability disclosure becomes greater when users are able to fix the software by themselves. Our paper is in line with the idea of Nizovtsev and Thursby (2007) that parties other than the software vendor may actively contribute in software security and we examine it empirically in markets that are not exclusively open source.

Secondly, a dearth of research exists on the role of third parties, while in practice, they are often actively involved in the lifecycle of a software and in improving security. Indeed, 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 each other and with the outside network through various protocols whose guidelines are maintained by public entities. Thus, the security of a company's system depends on a multitude of actors that in turn are dependent 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, etc. Since 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. This paper tries to fill the gap in the literature by analyzing not only the behavior of the software vendors but also of other third parties that actively contribute to software security, like downstream and upstream software vendors, security firms and individual researchers, public organizations and competitors.

Several empirical studies examine the impact of vulnerability disclosure on security related activities such as on the attack frequencies (Arora et al., 2006b), on software vendors' market value (Telang and Wattal, 2007), and on software vendors' patch release behavior (Arora et al., 2010b). We are also interested in whether a vulnerability disclosure gives an incentive to improve the security of the affected software. However our approach differs from them at least in two ways. First, we consider the disclosure of a vulnerability as a signal that updates the perceived security quality of the affected software rather than considering the effect that is directly related to the disclosed information. Secondly, we are interested in the impact of vulnerability disclosure on an activity – vulnerability discovery – that is specifically related to security spending. This is in line with the idea that the expected cost to breach a system, i.e., finding a vulnerability that was unknown before, can be a reasonable measure of the strength of a system (Schechter, 2002). Further, some argues that the market price to find an additional vulnerability may be a practical measure of the security level of a system (Camp and Wolfram, 2000; Schechter, 2002; Ozment, 2004).

Our work is also related to the recent economic literature on open innovation. A large number of researches document how users have been contributing efficiently to improve products and processes. Besides, open innovation is not only limited to the involvement of user communities; it encompasses all the "inflows and outflows of knowledge" that contribute to innovation (Chesbrough, Vanhaverbeke, and West, 2006). That is, it is most of all about an ecosystem of partners with whom to collaborate. In that sens, this stream of research is in tight relation with the industry platform

literature, which focus on strategies that platforms employ to influence and stimulate collaboration on complementary products and services from third parties (Gawer and Cusumano, 2014). One of the main goal of these streams of research is to examine the means and conditions that induce a more efficient participation from external resources. In our paper, we also study how a particular event affects third parties' incentives to exert an additional effort.

Furthermore, many researches insist on the fact that the notion of collaboration within the context of open source project and in business context is different (Boudreau and Lakhani, 2009; Pénin et al., 2011). In contrast with this observation, we study cases in which there is a mix between the contribution of user communities and a market mechanism. Indeed, by third parties, we designate both private actors that benefit from the market for software vulnerabilities as well as public actors and user communities. Moreover, the delimitation between private business purposes and the security community's goal is blurred. For instance, private actors like security firms are part of the security community and are even major actors that actively contributes to it. All in all, the effort exerted by an actor to improve software security can be either motivated by direct benefits from improving the security of the software because it is dependent to it, because of some intrinsic motivation to contribute to global security, or because of monetary incentives and other extrinsic motivation (reputation feedback, reciprocity).

3.3 Data and empirical Strategy

Our goal is to examine how a critical vulnerability disclosure on a software affects the effort exerted by various actors to improve its security. For that, we consider the effort exerted by an actor to discover new security flaws as a measure of its effort to improve the overall security of the software.[4] 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, the mobile OS and the desktop OS markets.

[4]The idea that the cost-to-break, i.e., the cost to find new vulnerabilities is an effective metric to measure the security level of a software has been defended by a number of researchers (Camp and Wolfram, 2000; Schechter, 2002; Ozment, 2004).

In order to study the causal relationship between a vulnerability disclosure and the effort of each actor 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 actor, before and after the extensive media coverage of a security flaw on the targeted software (the treatment group) and other software (the control group).

The main data set we use comes from Security Focus Bugtraq, which is a public database on software vulnerabilities. From this database, we collected information about all the vulnerabilities that affect any web browsers and operation systems and published from January 2009 to December 2018.[5] The raw data set provides the disclosure date of the vulnerability, which software it affects and who discovered it. We categorize the discoverers of the vulnerabilities into 6 types of actors: the software vendor, users (including companies that use the software or provide a service related to the affected vulnerability, as well as downstream and upstream vendors), security firms, individual researchers who do not precise their affiliation to a company or an organization, academics and public organizations, and the competitors of the affected software. The dependent variables for each specifications are built by consolidating this raw data set in several ways (see Subsection 3.3.3). The treatment is identified using Google Trends data. More precisely, we identify a particular vulnerability disclosure that has generated a spike in media coverage compared to all other vulnerability disclosures in a market (i.e., in web browsers, mobile OS or desktop OS).[6] We then examine how this shock affects the number of vulnerabilities that are discovered for each software belonging to the market.

We detail the econometric specification in the next subsection, then Subsection 3.3.2 presents the raw data set. Subsection 3.3.3 describes how we build our dependent variable for the three different specifications we use, then follows Subsection 3.3.4 where we discuss the identification of our treatment variable. Lastly, we verify the parallel trend assumption and detail the descriptive statistics in Subsection 3.3.5.

[5]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.

[6]For the web browser market, the data set is composed of vulnerabilities that affect only web browsers, and for operation system market only vulnerabilities that affect operation systems.

3.3.1 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 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:

$$y_{it} = \beta_0 + \beta_1 A_i \cdot P_t + FE_i + FE_t + \gamma X_{it} + \epsilon_{it}, \tag{3.1}$$

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, 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, and to zero outside this period. 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 ($post$).[8] FE_i and FE_t are software and time fixed effects.[9] 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, ϵ_{it} is the 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 – our treatment – between the treatment group and the control group. We expect that the sign of the coefficient β_1 is positive, i.e., a critical

[7] In our data set, the treatment group is the software that suffers from the disclosure and all other software in the same market – unaffected by the disclosed vulnerability – belongs to the control group.

[8] We consider that periods of more than 6 months are appropriate because the discovery of new vulnerabilities on a software (which differs for instance to discovering a simple bug on a functionality) does not happens more frequently.

[9] Thus the effect of A_i and P_t are included in the fixed effects

vulnerability disclosure would increase the global effort made in securing the software that suffers from the vulnerability disclosure.

Next, we estimate the following equation:

$$\begin{aligned} y_{ijt} =& \beta_0 + \beta_1 A_i \cdot P_t + \beta_2 A_i \cdot P_t \cdot ThirdParty_j + \beta_3 A_i \cdot ThirdParty_j \\ &+ \beta_4 P_t \cdot ThirdParty_j + \beta_5 ThirdParty_j + FE_i + FE_t + \gamma X_{it} + \epsilon_{ijt}, \end{aligned} \tag{3.2}$$

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. 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 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 actor separately:

$$\begin{aligned} y_{ijt} =& \beta_0 + \beta_1 A_i \cdot P_t \\ &+ \sum_{K_j \in Identifier\_Type} \beta_2^{K_j} A_i \cdot P_t \cdot K_j + \beta_3^{K_j} A_i \cdot K_j + \beta_4^{K_j} P_t \cdot K_j + \beta_5^{K_j} K_j \\ &+ FE_i + FE_t + \gamma X_{it} + \epsilon_{ijt}, \end{aligned} \tag{3.3}$$

where $Identifier\_Type = \{Users_j, Sec\_firms_j, Individuals_j, Public\_org_j, Competitors_j\}$ 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.

3.3.2 Raw data set

The raw data set consists in all the vulnerabilities reported from January 2009 to December 2018, associated to one or multiple web browsers or operation systems. The data is collected from Security Focus Bugtraq, which is a well-known public database on vulnerabilities. We use this database because it is the unique public database that provides information about who discovered – i.e., who first reported – a given vulnerability. An example of the raw information on Security Focus Bugtraq is presented

in Figure 3.9 in Appendix.

Table 3.1 – Number of observations in the raw data set

Type of software	Total number of vulnerabilities	Number of considered software	List of considered software
Web browser	2 651	6	Apple Safari, Google Chrome, Microsoft Internet Explorer or Edge, Mozilla Firefox, Mozilla Seamonkey, and Opera.
OS	12 539	16	
of which:			
Desktop OS	8 864		Apple MacOS, Debian Linux, Microsoft Windows, Oracle Linux, Red Hat Linux, SUSE Linux, Ubuntu Linux
Mobile OS	1 143		Apple iOS, Google Android
Other Unix like OS	1 625		FreeBSD, GNU, Oracle Solaris
Server OS	297		CentOS
Router/Switch OS	610		Cisco IOS, Juniper Junos

All the vulnerabilities in our data set have a patch at the date they are disclosed to public.[10] For vulnerabilities that affect more than one product, we have duplicated the observations in order to take the vulnerability into account for each software. Table 3.1 presents the number of observations we have in the raw data set for each market we consider. Software that do not present sufficient number of disclosed vulnerabilities during the observed period of time, i.e., that have less than one vulnerability discovered each month, are ignored. For instance, mobile OS such as RIM Blackberry, Nokia mobile or Windows mobile are not considered in our data set.[11]

Figure 3.8 (in Appendix) shows the evolution of the total number of vulnerabilities affecting web browsers and operation systems during the studied period. We observe that there is a slight increase in the number of vulnerabilities over time. Note that our specifications include a time fixed effect.

3.3.3 Dependent variable

Our dependent variable for the first specification (see Subsection 3.3.1 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 actor.

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

[11] This is why for mobile OS, only Apple iOS and Google Android are considered. Besides, in our regressions, we consider the operation systems as a whole and not the mobile OS separately.

In the raw data set, the discovery of each vulnerability is credited either to an individual, an organization, or a group of individuals and organizations. For each vulnerability, we code the type of actor the discoverers belong to.[12] 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 one over the number of contributors. Then the 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 presented in Subsection 3.3.1. For the first data set, we count the number of vulnerabilities affecting each software each month without considering who are the actors that have contributed. In the second data set, we define a dummy variable which indicates whether the identifier of the vulnerability is the software vendor or a third party (*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 actor. An example of how we have built our dependent variable is given in Table 3.6 in Appendix.

3.3.4 Treatment variables

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 Table 3.7 for summary statistics). Measuring the effect of all of these vulnerability disclosures 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 has raised particularly large media attention compared to other vulnerabilities. By considering a vulnerability that has been particularly critical and highly publicized compared to others, we claim that the effects that we identify are due to this disclosure rather than to 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

[12] As mentioned before, we categorize the vulnerability discoverers into 6 types of actors. Table 3.6.1 in Appendix describes how each actor may value the externality caused by the security of a software and thus would be affected by the disclosure of a critical vulnerability.

to visualize the relative evolution of a given search term on Google Search compared to other search terms. 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 web browsers, we compare the search trends for the terms "Internet Explorer vulnerability", "Chrome vulnerability", "Safari vulnerability", "Firefox vulnerability" and "Opera vulnerability" (see Figure 3.5 in Appendix). Then, for each of the three markets we study, we identify a peak search volume from the graphs obtained from Google Trend, which actually corresponds to the disclosure of a critical vulnerability. For instance, in Figure 3.5, we observe a peak search volume in mid 2014 for the search term "Internet Explorer vulnerability". This peak corresponds to a critical vulnerability affecting Internet Explorer, disclosed to public in April 26th 2014. This vulnerability was discovered by an independent security firm FireEye who reported it to Microsoft. It is a zero day vulnerability, that is, a vulnerability that is all the more critical because it did not have a security patch at the time it was disclosed. For the second case, the mobile operation systems, the identified critical vulnerability disclosure event corresponds to the disclosure of Android StageFright vulnerability in July 2015. As for the Zero Day in Internet Explorer, this critical and well-known vulnerability was also discovered by an independent security firm. Lastly, for desktop operation systems, the peak search volumes corresponds to the famous WannaCry ransomware attack happened in May 2017. It was the NSA that previously warned Microsoft about the possible theft of EternalBlue, which is the exploit used in WannaCry. Each of these events can be considered as unexpected, except by the software vendor. More details about the three vulnerability disclosure are presented in Figure 3.9 in Appendix.

These three vulnerability disclosures are our treatments in each market. We consider the software that is targeted by the vulnerability disclosure as the treatment group while other software in the same market belongs to the control group. With regard to the treatment period, we consider that a "treatment" begins at the date of the peak search volumes. Four alternative treatment periods are used, corresponding to a 6 months to 2 year-period after the disclosure.[13]

[13]Specifically, we consider the first 6 months after the disclosure, the first year, the first two years,

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 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 around the disclosure date. Indeed, most organizations apply these days a disclosure policy of 90 days.[14] 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.

3.3.5 Descriptive statistics

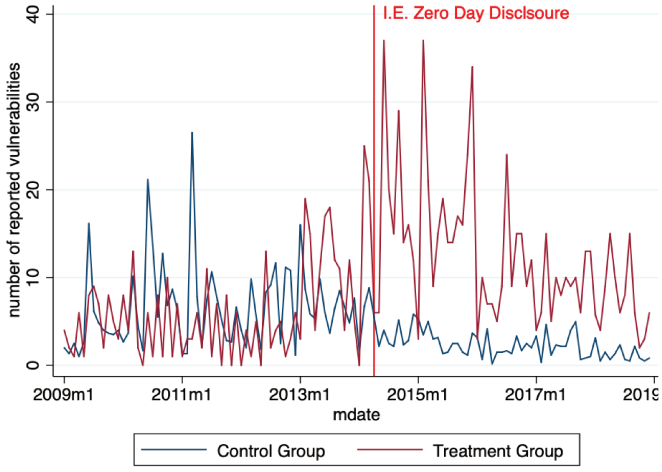
In this subsection, we provide some descriptive statistics related to the dependent variable and the impact of the treatment. Then we discuss the distribution of the different actors' contribution in our data set.

In Figures 3.1a, 3.1b, and 3.1c we first plot the average number of reported vulnerabilities over time, for the treatment group and the control group. 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. Then, for each cases, we visualize a significant increase in the number of vulnerabilities on the treatment group after the year the public announcement of a critical vulnerability occurs. We note that for Figure 3.1a and Figure 3.1c, 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 3.3.4, the sudden increase in the number of vulnerabilities could reflect the software vendor's behavior that suddenly publishes

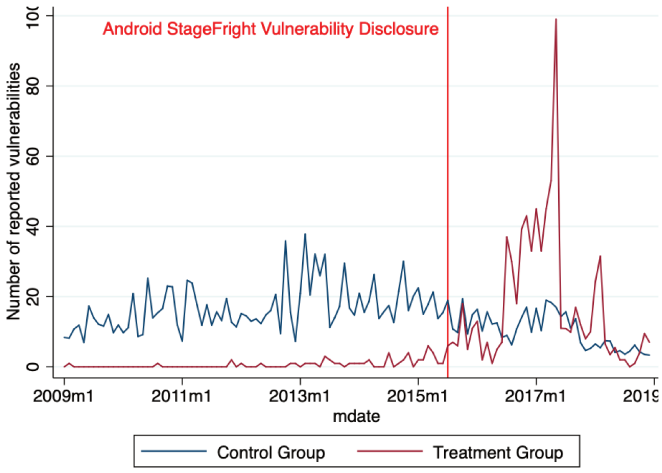
and the whole period after the disclosure as the alternative treatment periods.

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

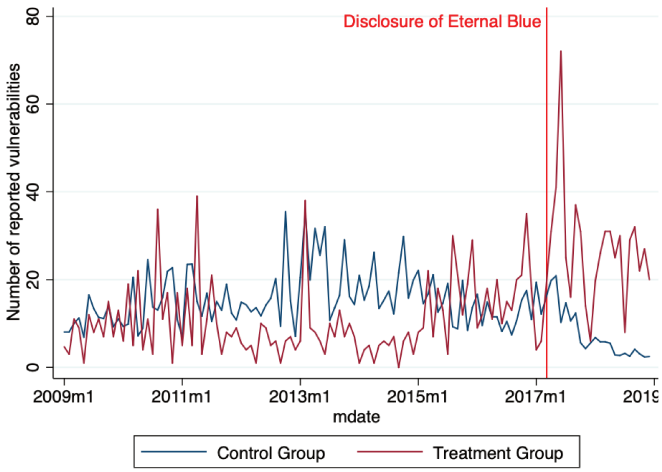
Figure 3.1 – Dependent variable for Specification 3.1



(a) Internet Explorer Zero Day case



(b) Android StageFright case



(c) Windows WannaCry and EternalBlue case

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.

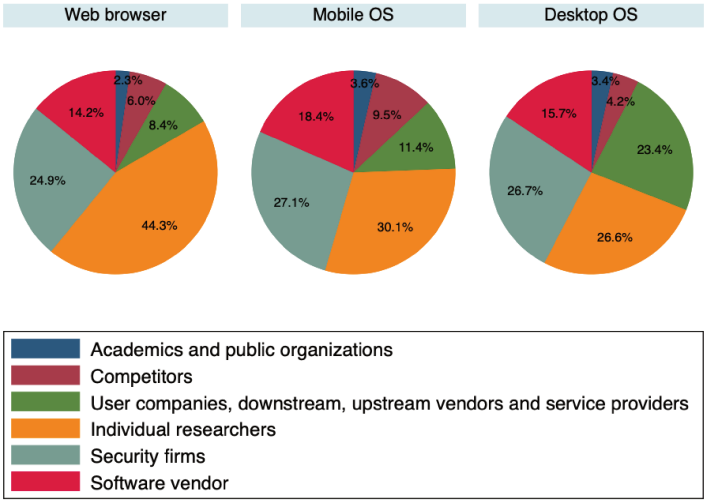


Figure 3.2 – Distribution of vulnerability identifiers

Figure 3.2 shows the distribution of vulnerability identifiers in the markets we study and Figure 3.10 (in Appendix) 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 parties 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. 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. We also observe that security firm also contributes more than software vendors in general. As to business users' contribution, we note that 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.

3.4 Results

Table 3.2 reports the estimation results for our baseline specification (equation 3.1), 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 ($post$).

Table 3.2 – Effect of a critical vulnerability disclosure on the number of discovered vulnerabilities.

treatment period is:	(1) $post6m$	(2) $post12m$	(3) $post24m$	(4) $post$
	Case of Internet Explorer Zero Day vulnerability disclosure			
$A \cdot P$	0.808*	0.712**	1.057***	1.775***
	(0.481)	(0.357)	(0.273)	(0.220)
Observations	819	819	819	819
Number of software				
Wald χ^2	432.85	433.99	445.66	490.81
AME of $A \cdot P$	4.179*	3.673**	5.372***	8.752***
	(2.510)	(1.867)	(1.449)	(1.271)
	Case of Android Stagefright vulnerability disclosure			
$A \cdot P$	0.747	0.611*	2.194***	3.332***
	(0.493)	(0.369)	(0.270)	(0.265)
Observations	1,872	1,872	1,872	1,872
Number of software				
Wald χ^2	646.61	646.95	720.50	814.01
AME of $A \cdot P$	4.123	3.369*	12.00***	18.23***
	(2.729)	(2.045)	(1.601)	(1.653)
	Case of Windows Eternal Blue vulnerability disclosure			
$A \cdot P$	0.686	0.648*	1.320***	
	(0.476)	(0.348)	(0.283)	
Observations	1,856	1,856	1,856	
Number of software				
Wald χ^2	626.02	627.50	648.96	
AME of $A \cdot P$	3.676	3.464*	7.066***	
	(2.574)	(1.881)	(1.603)	

Note: Negative binomial regressions. Dependent variable is number of reported vulnerabilities. Product fixed effects and time fixed effects are included in all specifications as well as other controls ($SoftwareAge$, $EndofLife$) and a constant. For Eternal Blue case, $post24m = post$. Standard errors in parentheses. *** p<0.01, ** p<0.05, * p<0.1

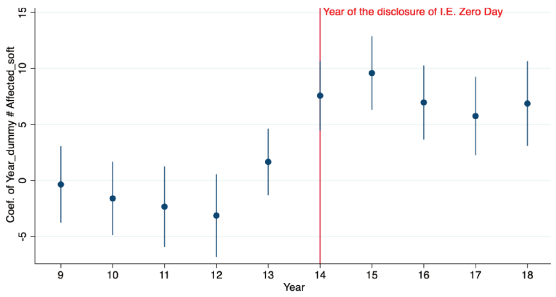
We have count data and given the presence of significant over-dispersion of the dependent variable with standard deviation superior to the mean (see Summary Statistics in Appendix Table 3.7), we use a negative binomial regression to estimate the equations. To facilitate the exposition, we only report 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 $post24m = post$).

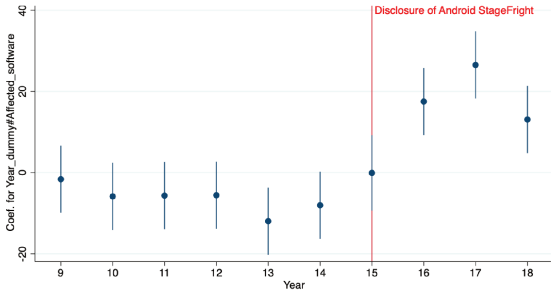
As expected, all the coefficients for $A \cdot P$ are positive for each of the three markets we study. That is, after the disclosure of a critical vulnerability, the total number of vulnerabilities discovered on the software concerned by the disclosure increases. Specifically, 5 more vulnerabilities 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 unaffected web browsers. In the same way, the disclosure of Android Stagefright vulnerability disclosure has increased the number of vulnerabilities discovered in Google Android by 12 additional vulnerabilities each month during the first two years following the disclosure, while the number of vulnerabilities found in Microsoft Windows has increased by 7 more vulnerabilities each month after the disclosure of Eternal Blue. We also observe that the effect of the disclosure 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 treatment period. This suggests that the vulnerability disclosure does not have an immediate effect on the behavior of the vulnerability discoverers but rather a gradual effect over time. Two explanations can be advanced. First, discovering new security flaws in a software is not a trivial task; it is not because one puts 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 vulnerability was 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.

Additionally, Figure 3.3a to 3.3c 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 3.1. Each graph shows that the difference between the treatment and the control groups is not varying significantly over time during the non-treated period. This visual inspection allows us to check the validity of the parallel trend assumption and to visualise the timing of the effect.

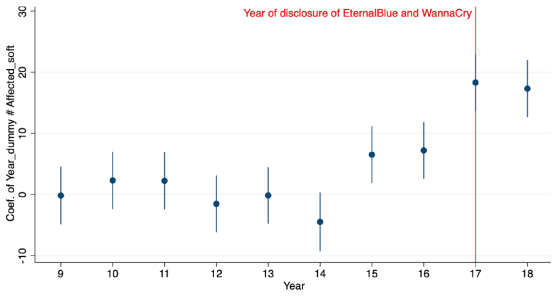
Figure 3.3 – Difference between treatment and control group's outcome, all actors confounded.



(a) Internet Explorer Zero Day case



(b) Android StageFright case



(c) WannaCry and Eternalblue case

Next, Table 3.3 reports the estimation results using our second specification (equation 3.2), in which we examine the effort exerted either by the software vendor and other third parties 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 exposition, although all the regressions include the other interaction terms we specified in Specification 3.2 (the terms $A \cdot ThirdParty$, $P \cdot ThirdParty$), 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 cases we study, meaning that in general, third parties contribute more in finding new security flaws than the software vendor. The positive sign of the coefficient is

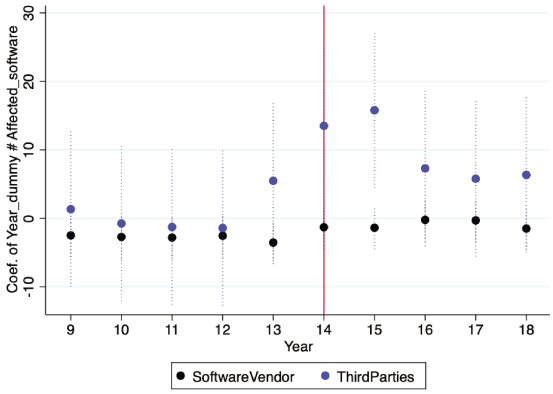
Table 3.3 – Effect of a highly publicized vulnerability disclosure on the software vendor vs. third party identifiers' behavior

treatment period is:	(1) $post12m$	(2) $post24m$	(3) $post$
	Case of Internet Explorer Zero-day vulnerability disclosure		
$A \cdot P$	-0.556	-1.051**	-0.619*
	(0.746)	(0.468)	(0.333)
$A \cdot P \cdot ThirdParty$	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 χ^2	992.48	998.28	1027.34
	Case of Android Stagefright vulnerability disclosure		
$A \cdot P$	0.424	0.627	1.499***
	(0.661)	(0.488)	(0.369)
$A \cdot P \cdot ThirdParty$	-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 χ^2	2028.30	2036.17	2119.79
	Case of Windows Eternal Blue vulnerability disclosure		
$A \cdot P$	0.675	0.121	0.549
	(0.629)	(0.463)	(0.378)
$A \cdot P \cdot ThirdParty$	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 χ^2	2016.87	2038.32	2093.08

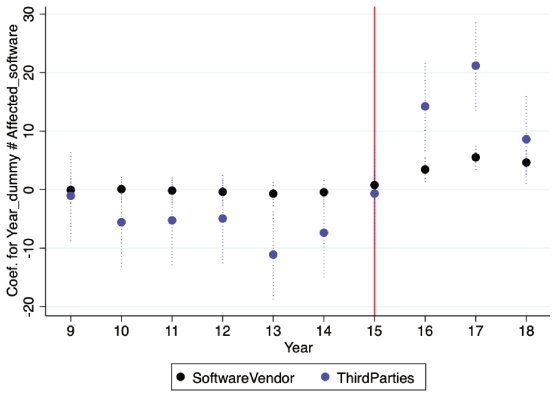
Note: Dependent variable is the number of reported vulnerabilities. Negative binomial regressions. $A{\cdot}ThirdParty$, $P{\cdot}ThidParty$, Product fixed effects and time fixed effects are included in all specifications as well as other controls ($SoftwareAge$, $EndofLife$). For Eternal Blue case,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

as expected, as the distribution of each actors' contribution (see Subsection 3.6.1) 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 $A \cdot P$, the coefficient is not systematically significant, but it is always positive. That is, overall, the behavior of third parties 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 type of actor could be different from another, but as we encompass all the different actors in one category – the "$ThirdParty$" –, the significant positive effect on some type of actors could be mitigated by a less significant or negative effect on others.

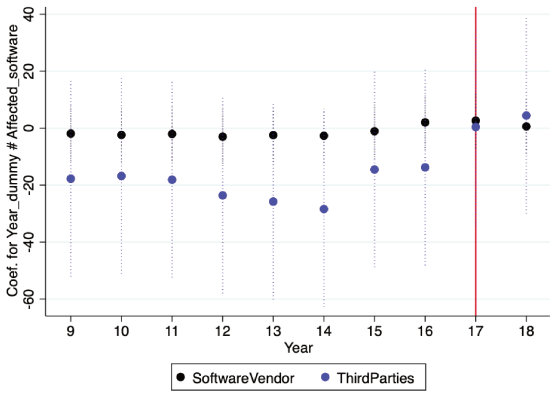
Figure 3.4 – Difference between treatment and control group's outcome, comparison of the software vendor's contribution and third parties' contribution.



(a) Internet Explorer Zero Day case



(b) Android StageFright case



(c) WannaCry and Eternalblue case

In Figure 3.4a to 3.4c, 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 parties' effort (in blue dots). The graphs clearly show that in each of the three cases, third parties' contribution increases significantly after the critical vulnerability disclosure occurs, while the change is less significant for the software vendor's contribution.

Table 3.4 – Effect of a highly publicized vulnerability disclosure on each actors ($post24m$ as the treatment period).

	(1) Web browser case	(2) Mobile OS case	(3) Desktop OS case
$A \cdot P$	-0.475	1.321***	0.515
	(0.350)	(0.364)	(0.371)
$A \cdot P \cdot Public\_org$	0.455	1.042	2.232***
	(0.960)	(0.775)	(0.663)
$A \cdot P \cdot Competitors$	0.916*	-1.789	-15.74
	(0.556)	(28,494)	(2,417)
$A \cdot P \cdot Users$	0.264	1.179**	1.665***
	(0.664)	(0.549)	(0.522)
$A \cdot P \cdot Individuals$	0.826*	1.126**	1.017*
	(0.472)	(0.512)	(0.524)
$A \cdot P \cdot Sec\_firms$	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: Dependent variable is the number of reported vulnerabilities. Negative binomial regressions. A, P, $A \cdot K$ and $P \cdot K$ with $K \in Identifier\_type$, product fixed effects and time fixed effects, $SoftwareAge$, $EndofLife$ and a constant 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

Lastly, Table 3.4 reports the effect of a critical vulnerability disclosure on each actors' behavior. To facilitate the exposition, 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 are reported in Table 3.8 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 but they contribute more than their competitors or the security firms. Note that the coefficient for $Users$ dummy is negative for the web browser case, meaning that contrary to the case of operation systems, the contribution of the 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 $A \cdot P$ and the identifier's dummy, we observe that most of the coefficients are positive except for the term $A \cdot P \cdot Competitors$. That is, our estimation results suggest that third parties 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.

3.5 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 parties 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) existing works on software security focus much more on the behavior of software vendors in providing security than on the contribution of other third parties, while (2) we show that third parties' 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 parties 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 others. 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, in a future version of the work, in order to obtain more robust results, I intend to examine the correlation between some

particular security investment events and the contribution of third parties. Indeed, one concern regarding the result we obtain is that third parties' contribution could actually be affected by the launch of particular vulnerability research programs sponsored by the software vendor itself or a specific group of users. Secondly, our findings rely on three specific cases on three markets that present some similarities each other. The study of an additional case may strengthen the reliability of our results. 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.

3.6 Appendix

3.6.1 Categorization of the actors

The public announcement of a vulnerability may affect each type of actors for different reasons and in different degrees. 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 updates the probability to find additional vulnerabilities in the affected software. 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. At the same time, as the disclosed information is shared with all the other third parties, competition can also reduce the effort they may exert.

- Academic researchers, public CERTs, and public organizations:[15] 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.

- 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.

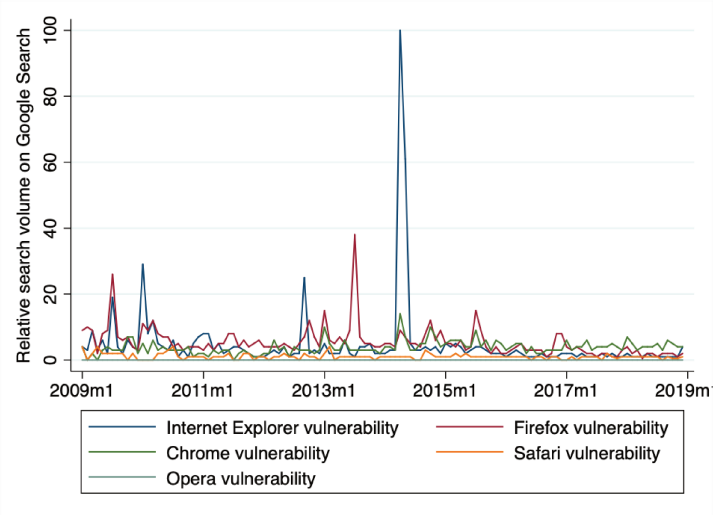
This categorization suggests that the public announcement of a vulnerability may affect each type of actors for different reasons and in different degrees.

[15] Private CERTs are accounted as a private company

3.6.2 The three critical vulnerability disclosures

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

Figure 3.5 – Google Search trend for web browser vulnerabilities from 2009 to 2018



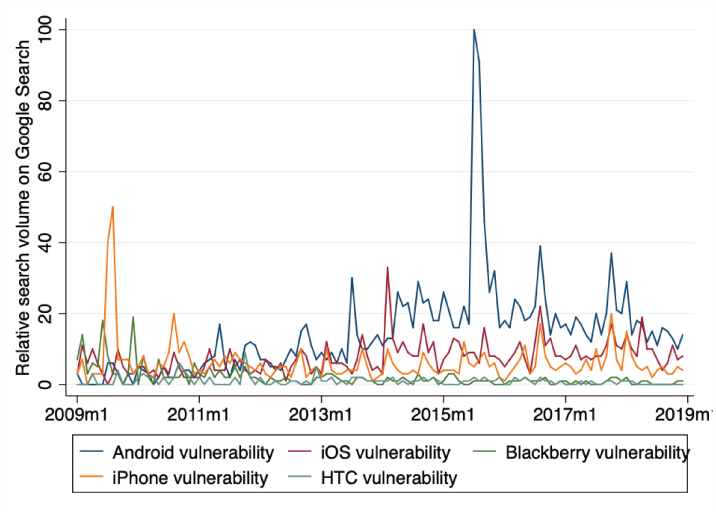
In Figure 3.5 we observe a peak search volume in mid 2014 for the search term "Internet Explorer vulnerability". This corresponds to a vulnerability that was announced in April 26th 2014 by Microsoft and the security firm FireEye. It is a zero day vulnerability – i.e. a vulnerability that did not have 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 existing versions of Microsoft Internet Explorer.[16] 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.[17]

[16] 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*

[17] *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-explorer-versions-9-through-11-identified-in-targeted-attacks.html

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

Figure 3.6 – Google Search trend for mobile OS vulnerabilities



The peak search volume we visualize in Figure 3.6 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.[18] 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.[19]

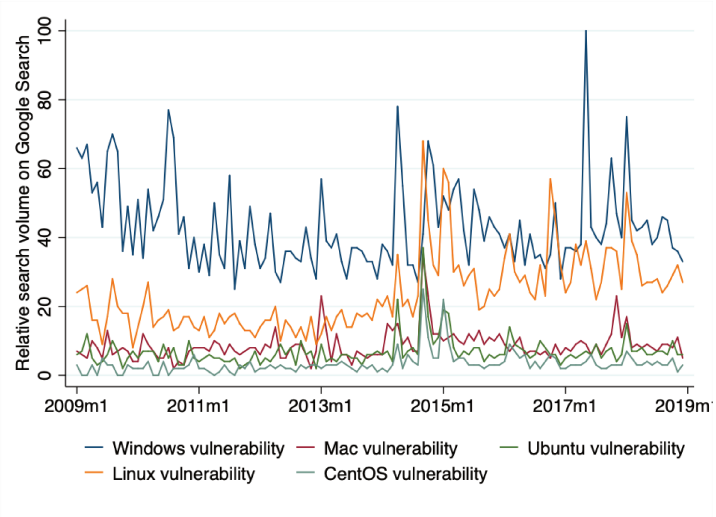
- Treatment for the case of desktop OS: Microsoft Windows Eternal blue and the famous *Wannacry* malware

Figure 3.7 plots the relative search volumes for terms that are the most popular in

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

[19] https://www.theverge.com/2016/9/6/12816386/android-nougat-stagefright-security-update-mediaserver

Figure 3.7 – 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.[20] 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.[21] Microsoft released a patch event for Windows XP which support ended in 2014.

[20]The vulnerability is denoted by entry CVE-2017-0144 in the Common Vulnerabilities and Exposures (CVE) catalog.
[21]source: Wikipedia

3.6.3 Other tables and figures

Figure 3.8 – Evolution of the total number of vulnerabilities

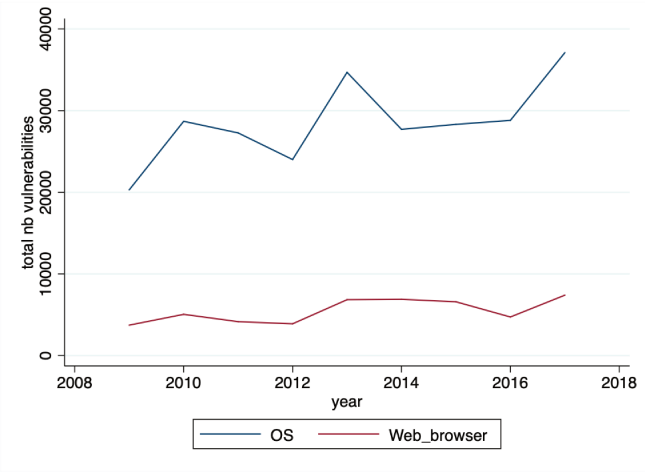


Figure 3.9 – An example of the raw data we collect from Security Focus Bugtraq

info | discussion | exploit | solution | references

Microsoft Windows Kernel 'Win32k.sys' CVE-2016-7255 Local Privilege Escalation Vulnerability

Bugtraq ID:	94064
Class:	Unknown
CVE:	CVE-2016-7255
Remote:	No
Local:	Yes
Published:	Nov 08 2016 12:00AM
Updated:	Sep 25 2017 08:00PM
Credit:	Neel Mehta and Billy Leonard of Googleâ??s Threat Analysis Group Feike Hacquebord, Peter Pi and Brooks Li of Trend Micro
Vulnerable:	Microsoft Windows Vista x64 Edition Service Pack 2 0 Microsoft Windows Vista SP2 Microsoft Windows Server 2016 for x64-based Systems 0 Microsoft Windows Server 2012 R2 0 Microsoft Windows Server 2012 0 Microsoft Windows Server 2008 R2 for x64-based Systems SP1 Microsoft Windows Server 2008 R2 for Itanium-based Systems SP1 Microsoft Windows Server 2008 for x64-based Systems SP2 Microsoft Windows Server 2008 for Itanium-based Systems SP2 Microsoft Windows Server 2008 for 32-bit Systems SP2 Microsoft Windows RT 8.1 Microsoft Windows 8.1 for x64-based Systems 0 Microsoft Windows 8.1 for 32-bit Systems 0 Microsoft Windows 7 for x64-based Systems SP1 Microsoft Windows 7 for 32-bit Systems SP1 Microsoft Windows 10 Version 1607 for x64-based Systems 0 Microsoft Windows 10 Version 1607 for 32-bit Systems 0 Microsoft Windows 10 version 1511 for x64-based Systems 0 Microsoft Windows 10 version 1511 for 32-bit Systems 0 Microsoft Windows 10 for x64-based Systems 0 Microsoft Windows 10 for 32-bit Systems 0

Table 3.5 – 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)

Figure 3.10 – Evolution of the distribution

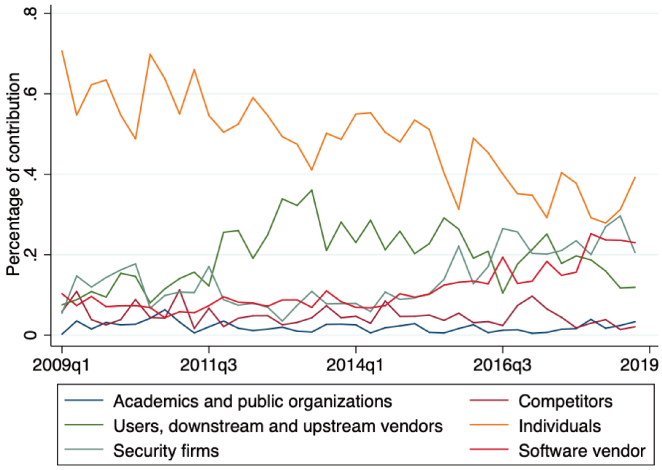


Table 3.6 – Example illustrating how we have built the three data sets

Exemple of raw data set

id	software	Disclosed date	Credit	Explanation
49732	Android	12/4/2019	the software vendor and Individual α	Two types of actors have contributed in finding this vulnerability. Thus we consider that each of the two actors (the software vendor and the individual researchers) have discovered $\frac{1}{2}$ of the vulnerability.
49900	Android	27/4/2019	Individual β and a security firm	$\frac{1}{2}$ vulnerability is attributed to the type of actor "Individual researchers" and the other $\frac{1}{2}$ is attributed to the software vendor.
49999	Android	01/5/2019	Individual γ	
50206	Android	01/5/2019	Downstream vendor, Individual α and the software vendor	
58326	Android	29/5/2019	Public organization	

Aggregated data set for Specification 3.1

y_{it}	$software$	monthly date	Explanation
2	Android	2019m4	A total of 2 vulnerabilities were found in April 2019 (id 49732 and id 49900)
3	Android	2019m5	A total of 3 vulnerabilities were found in May 2019

Aggregated data set for Specification 2

y_{ijt}	$software$	$mdate$	$ThirdParty$	Explanation
0.5	Android	2019m4	0	In April 2019, the software vendor discovered one vulnerability (Id 49732) with an individual so it has found $\frac{1}{2}$ vulnerability
1.5	Android	2019m4	1	
0.33	Android	2019m5	0	
2.67	Android	2019m5	1	

Aggregated data set for Specification 3

y_{ijt}	$software$	$mdate$	$Identifier\_Type$	Explanation
0.5	Android	2019m4	Software vendor	
1	Android	2019m4	Individuals	In April 2019, "Individual researchers" have participated in the discovery of two vulnerabilities (of Ids 49732 and 49900) and for each vulnerability they discovered it with another type of actors. In sum, they discovered $\frac{1}{2} + \frac{1}{2}$ vulnerabilities.
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 3.7 – 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 3.8 – Effect of a critical vulnerability disclosure on each actors (Other treatment periods)

	Case 1			Case 2			Case 3		
Treatment Period is:	*post*6*m*	*post*12*m*	*post*24*m*	*post*6*m*	*post*12*m*	*post*24*m*	*post*6*m*	*post*12*m*	*post*24*m*
Public_org	-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)
Competitors	-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)
Users	-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)
Individuals	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)
Sec_firms	-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)
$P \cdot A$	-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)
$Public\_org \cdot P \cdot A$	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)
$Competitors \cdot P \cdot$ 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)
$Users \cdot P \cdot A$	-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)
$Individuals \cdot P \cdot A$	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)
$Sec\_firms \cdot P \cdot A$	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

General Conclusion

Dealing with cybersecurity issues has become as vital as technology itself to our modern society. In response to this need, computer scientists started working with practitioners and researchers from multiple fields to revisit the way to approach IT security. Following the idea that economic analysis can bring valuable insights to better understand and manage security risks, this thesis aims to contribute to the field of economics of information security.

In this thesis, we have conducted three empirical studies by collecting and building three original data sets related to software vulnerability discovery and patching. We have considered several ways to measure the effort exerted by an actor that contributes to software security, from measuring the responsiveness of a software editor to release a security patch to considering how many new vulnerabilities a third party reports to the software vendor. We examined how the security of a software is affected by the incentives of the actors that depend on it through various standpoints: the security level of a software can be considered as the product quality chosen by a vendor that maximizes its profit according to the competition intensity in the market; the effort exerted to improve a software security can depend on the disclosure of a critical vulnerability; or, the security can be improved through a crowdsourced contest in which the level of information provided to the participants determines their performance.

In the present section, we summarize our results and present relevant future research directions.

In Chapter 1, we examined the impact of competition intensity on software vendors' security investment behavior. While both academics and practitioners have insisted on the danger of software monoculture, we find that market concentration is not necessarily harmful to security provision: higher market concentration positively impacts vendors' responsiveness in patching vulnerabilities, although this effect is reduced when a vendor is too dominant. Our findings challenge conventional understanding about the effect of competition in security. It also provides policy makers with empirical evidence on the impact of competition on firms' security investment incentives in free software markets such as the web browser one, a case which is even more of interest as it is very common in today's digital markets. Furthermore, in line with some practitioners' claims, we find that diverse externalities such as the severity of the vulnerability, the software age or open source licensing also significantly impact the vendors' incentives to release a patch quickly. This chapter presents a number of limitations. One important limitation is that I focus on a specific market that presents a unique position on the web environment, with some vendors that are omnipresent in the digital market and have a predominant position such as Google. Another limitation is that security is not always the primary feature valued by a regular web browser user. Possible future work would be to look at the effect of competition in a market where security is the most central feature of the product and in which we can assume that all users are able to correctly assess the security quality. For instance, security software used by corporate users fit into these criteria.

In Chapter 2, I study how hackers' perception of the uncertainty to obtain a reward, determined by the level of information a contest provides about the contractual terms, affects the effectiveness of the contest. Our results suggest that in the context of tournaments such as a vulnerability research programs (VRPs), the possibility that agents with different characteristics feel attracted by different types of contract leads the VRP owner to a trade-off between having a larger number of participation but attracting less performant participants and attracting higher quality participants but generating fewer participations. This ex-ante self selection process of the "workers" is all the more important for contests like VRPs since this can reduce the cost induced to manage the non-relevant participations. Furthermore, bug bounty platforms provide valuable data and study cases to analyze the ecosystem around software vulnerability

discovery and fixing. For now, few empirical studies exist on bug bounty platforms, one reason being the difficulty to exploit the data because researchers have usually only partial access to it. An interesting question that could be investigated is whether the VRPs are used simply because it is less costly to employ external individual resources, or because companies truly benefit from the crowd's innovativeness.

Finally, in Chapter 3, we examined how the public disclosure of a critical vulnerability impacts not only the software vendor's behavior but also the behavior of other third parties. We showed that third parties' overall contribution in software security is considerable and that their contribution is significantly affected by externalities like the disclosure of a critical vulnerability. Our findings suggest that one should not ignore the incentives and the potential contribution of third parties when studying software security. This suggests to account for the interactions between all the security actors when analyzing the mechanisms concerning software security rather than to analyze separately the behavior of a software editor or its reaction to attacks. One possible extension of our empirical study would be to create proxies for corporate users' switching costs, which would allow to study whether vulnerability disclosure affects users in different magnitudes according to different kinds of switching costs.

Bibliography

D. Acemoglu, A. Malekian, and A. Ozdaglar. Network security and contagion. *Journal of Economic Theory*, 166:536–585, 2016.

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

R. Anderson. Why cryptosystems fail. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 215–227. ACM, 1993.

R. Anderson. Why information security is hard-an economic perspective. In *Proceedings of the 17th Annual Computer Security Applications Conference*, page 358. IEEE Computer Society, 2001.

R. Anderson. Closing the phishing hole–fraud, risk and nonbanks. In *Federal Reserve Bank of Kansas City–Payment System Research Conferences*, pages 41–56, 2007.

R. Anderson, R. Böhme, R. Clayton, and T. Moore. Security economics and the internal market. *Study commissioned by ENISA*, 2008.

N. Archak and A. Sundararajan. Optimal design of crowdsourcing contests. *ICIS 2009 proceedings*, page 200, 2009.

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, 2006a.

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, 2006b.

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.

K. Arrow. Economic welfare and the allocation of resources for invention. In *The Rate and Direction of Inventive Activity: Economic and Social Factors*, pages 609–626. Princeton University Press, 1962.

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.

T. August, M. F. Niculescu, and H. Shin. Cloud implications on software network structure and security risks. *Information Systems Research*, 25(3):489–510, 2014.

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.

V. Bier, S. Oliveros, and L. Samuelson. Choosing what to protect: Strategic defensive allocation against an unknown attacker. *Journal of Public Economic Theory*, 9(4): 563–587, 2007.

R. Böhme. A comparison of market approaches to software vulnerability disclosure. In *International Conference on Emerging Trends in Information and Communication Security*, pages 298–311. Springer, 2006.

R. Bohme and T. Moore. The iterated weakest link. *IEEE Security & Privacy*, 8(1): 53–55, 2010.

K. Boudreau and K. Lakhani. How to manage outside innovation. *MIT Sloan management review*, 50(4):69, 2009.

K. J. Boudreau, N. Lacetera, and K. R. Lakhani. Incentives and problem uncertainty in innovation contests: An empirical analysis. *Management science*, 57(5):843–863, 2011.

E. Brousseau and T. Pénard. The economics of digital business models: A framework for analyzing the economics of platforms. *Review of network Economics*, 6(2), 2007.

C. B. Cadsby, F. Song, and F. Tapon. Sorting and incentive effects of pay for performance: An experimental investigation. *Academy of management journal*, 50(2): 387–405, 2007.

L. J. Camp and C. Wolfram. Pricing security. In *Proceedings of the CERT Information Survivability Workshop*, pages 31–39, 2000.

P. Casas-Arce and F. A. Martínez-Jerez. Relative performance compensation, contests, and dynamic incentives. *Management Science*, 55(8):1306–1320, 2009.

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.

H. Chesbrough, W. Vanhaverbeke, and J. West. *Open innovation: Researching a new paradigm*. Oxford University Press on Demand, 2006.

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

C. W. Chow. The effects of job standard tightness and compensation scheme on performance: An exploration of linkages. *The Accounting Review*, 58(4):667, 1983.

D. E. Chubin. State of the field the conceptualization of scientific specialties. *The sociological quarterly*, 17(4):448–476, 1976.

J. S. Demski and G. A. Feltham. Economic incentives in budgetary control systems. *Accounting Review*, pages 336–359, 1978.

D. Dey, A. Lahiri, and G. Zhang. Hacker behavior, network effects, and the security software market. *Journal of Management Information Systems*, 29(2):77–108, 2012.

D. Dey, A. Lahiri, and G. Zhang. Quality competition and market segmentation in the security software market. *Mis Quarterly*, 38(2), 2014.

T. Dohmen and A. Falk. Performance pay and multidimensional sorting: Productivity, preferences, and gender. *American Economic Review*, 101(2):556–90, 2011.

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.

B. Edwards, S. Hofmeyr, and S. Forrest. Hype and heavy tails: A closer look at data breaches. *Journal of Cybersecurity*, 2(1):3–14, 2016.

T. Eriksson and M. C. Villeval. Performance-pay, sorting and social motivation. *Journal of Economic Behavior & Organization*, 68(2):412–421, 2008.

D. D. Fehrenbacher, S. E. Kaplan, and B. Pedell. The relation between individual characteristics and compensation contract selection. *Management Accounting Research*, 34:1–18, 2017.

M. Finifter, D. Akhawe, and D. Wagner. An empirical study of vulnerability rewards programs. In *Presented as part of the 22nd {USENIX} Security Symposium ({USENIX} Security 13)*, pages 273–288, 2013.

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

R. L. Fullerton and R. P. McAfee. Auctionin entry into tournaments. *Journal of Political Economy*, 107(3):573–605, 1999.

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

S. M. Garcia and A. Tor. The n-effect: More competitors, less competition. *Psychological Science*, 20(7):871–877, 2009.

A. Gawer and M. A. Cusumano. Industry platforms and ecosystem innovation. *Journal of product innovation management*, 31(3):417–433, 2014.

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 and M. P. Loeb. The economics of information security investment. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):438–457, 2002.

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.

J. A. Jacobs. *In defense of disciplines: Interdisciplinarity and specialization in the research university*. University of chicago Press, 2014.

M. C. Jensen. Paying people to lie: The truth about the budgeting process. *European Financial Management*, 9(3):379–406, 2003.

L. B. Jeppesen and K. R. Lakhani. Marginality and problem-solving effectiveness in broadcast search. *Organization science*, 21(5):1016–1033, 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.

B. C. Kim, P.-Y. Chen, and T. Mukhopadhyay. The effect of liability and patch release on software security: The monopoly case. *Production and Operations Management*, 20(4):603–617, 2011.

J. H. Lala and F. B. Schneider. It monoculture security risks and defenses. *IEEE Security & Privacy*, 7(1):12–13, 2009.

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

E. P. Lazear. Performance pay and productivity. *American Economic Review*, 90(5): 1346–1361, 2000a.

E. P. Lazear. The power of incentives. *American Economic Review*, 90(2):410–414, 2000b.

T. X. Liu, J. Yang, L. A. Adamic, and Y. Chen. Crowdsourcing with all-pay auctions: A field experiment on taskcn. *Management Science*, 60(8):2020–2037, 2014.

T. Maillart, M. Zhao, J. Grossklags, and J. Chuang. Given enough eyeballs, all bugs are shallow? revisiting eric raymond with bug bounty programs. *Journal of Cybersecurity*, 3(2):81–90, 2017.

D. A. Matsa. Competition and product quality in the supermarket industry. *The Quarterly Journal of Economics*, 126(3):1539–1591, 2011.

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.

S. Mitra and S. Ransbotham. Information disclosure and the diffusion of information security attacks. *Information Systems Research*, 26(3):565–584, 2015.

B. Moldovanu and A. Sela. The optimal allocation of prizes in contests. *American Economic Review*, 91(3):542–558, 2001.

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

A. Ozment. Bug auctions: Vulnerability markets reconsidered. In *Third Workshop on the Economics of Information Security*, pages 19–26, 2004.

J. Pénin, C. Hussler, and T. Burger-Helmchen. New shapes and new stakes: a portrait of open innovation as a promising phenomenon. *Journal of Innovation Economics Management*, (1):11–29, 2011.

C. Raasch, V. Lee, S. Spaeth, and C. Herstatt. The rise and fall of interdisciplinary research: The case of open source innovation. *Research policy*, 42(5):1138–1151, 2013.

S. Ransbotham, S. Mitra, and J. Ramsey. Are markets for vulnerabilities effective? *Mis Quarterly*, pages 43–64, 2012.

T. Rayna and L. Striukova. Involving consumers: the role of digital technologies in promoting 'prosumption'and user innovation. *Journal of the Knowledge Economy*, pages 1–20, 2016.

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

J. Salop and S. Salop. Self-selection and turnover in the labor market. *The Quarterly Journal of Economics*, pages 619–627, 1976.

S. E. Schechter. Quantitatively differentiating system security. In *The First Workshop on Economics and Information Security*, pages 16–17. Citeseer, 2002.

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

J. S. Silva and S. Tenreyro. On the existence of the maximum likelihood estimates in poisson regression. *Economics Letters*, 107(2):310–312, 2010.

G. Solon, S. J. Haider, and J. M. Wooldridge. What are we weighting for? *Journal of Human resources*, 50(2):301–316, 2015.

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.

O. Temizkan, R. L. Kumar, S. Park, and C. Subramaniam. Patch release behaviors of software vendors in response to vulnerabilities: An empirical analysis. *Journal of management information systems*, 28(4):305–338, 2012.

C. Terwiesch and Y. Xu. Innovation contests, open innovation, and multiagent problem solving. *Management science*, 54(9):1529–1543, 2008.

H. Varian. System reliability and free riding. pages 1–15, 2004.

E. Von Hippel and G. Von Krogh. Open source software and the "private-collective" innovation model: Issues for organization science. *Organization science*, 14(2): 209–223, 2003.

W. S. Waller and C. W. Chow. The self-selection and effort effects of standard-based employment contracts: A framework and some empirical evidence. *Accounting Review*, pages 458–476, 1985.

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

J. M. Wooldridge. Control function methods in applied econometrics. *Journal of Human Resources*, 50(2):420–445, 2015.

M. Zhao, J. Grossklags, and P. Liu. An empirical study of web vulnerability discovery ecosystems. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1105–1117. ACM, 2015.

M. Zhao, A. Laszka, and J. Grossklags. Devising effective policies for bug-bounty platforms and security vulnerability discovery. *Journal of Information Policy*, 7: 372–418, 2017.

Titre: Economie de la cybersécurité et le marché des vulnérabilités

Mots clés: économie de la cybersécurité, sécurité de l'information, économie de l'innovation, économie industrielle

Résumé: L'environnement cybernétique est devenu un maillon essentiel au fonctionnement de notre société et de nos activités socio-économiques. Cette transformation va de pair avec un changement d'échelle et de portée des menaces de sécurité numérique, qui deviennent d'autant plus nombreuses et plus sophistiquées.

Dans un environnement mondialisé où les systèmes sont connectés à de millions d'autres systèmes, les parties prenantes sont engagées dans de multiples interactions stratégiques. Qui doit-on responsabiliser et de quelle manière, afin d'inciter à une gestion efficace de la sécurité ? De quelle façon les différentes motivations économiques de chacun influencent-elles les décisions en matière de sécurité et par conséquent ont-elles des impacts sur la vulnérabilité d'un système ? Les récentes et rapides évolutions en matière de cybersécurité apportent de nouveaux défis et force est de constater que le développement des solutions techniques ne suffit pas à comprendre et maîtriser les risques.

Dans cette thèse, nous mobilisons les outils de l'économie industrielle pour appréhender des éléments qui ont transformé l'environnement de la cybersécurité : L'adoption de modèle de revenu basé sur la gratuité des logiciels, l'utilisation de mécanisme de crowdsourcing dans la découverte des failles de sécurité, l'implication croissante des acteurs externes à l'entreprise tels que les chercheurs individuels, les concurrents, les firmes de sécurité, ou les organismes publiques impliqués dans l'amélioration de la sécurité des logiciels. Nous nous attachons en particulier à comprendre les motivations des acteurs majeurs de la sécurité, allant de l'éditeur de logiciels aux tierces parties telles que les white-hat hackers et les firmes de sécurité.

Cette thèse est constituée de trois chapitres distincts, présentant chacun une contribution empirique. Le premier chapitre s'intéresse à la relation entre la réactivité des éditeurs de logiciel à corriger les failles de sécurité et l'intensité de la concurrence sur le marché. Nous étudions le cas d'un marché au coeur de la sécurité d'Internet, celui des navigateurs web, où l'utilisateur jouit d'une gratuité et les éditeurs dérivent leur revenu d'un autre marché connexe – celui des moteurs de recherche – et par conséquent sont en concurrence par la qualité du navigateur. A travers l'analyse économétrique de données de la correction des failles de sécurité sur les navigateurs web sur une dizaine d'années, nous montrons que la concurrence sur le marché n'incite pas nécessairement les éditeurs à renforcer la sécurité. Le deuxième chapitre se focalise sur le crowdsourcing des hackers pour découvrir des failles de sécurité, mécanisme représentatif du marché des vulnérabilités qui capitalise sur la contribution des tierces-parties. A travers l'analyse empirique de 156 programmes de chasse aux bug gérés sur la plateforme HackerOne, nous montrons comment la perception de l'incertitude à être rémunérés des hackers, défini par le niveau de détail des termes contractuels, affecte leur choix de participation et par conséquent l'efficacité du programme. Enfin, dans un troisième chapitre, nous examinons comment la divulgation publique d'une vulnérabilité critique sur un système affecte le comportement des acteurs à fournir un effort pour améliorer sa sécurité. A travers l'analyse de panels de données sur 3 cas de divulgation de faille de sécurité particulièrement critique, nous montrons combien les acteurs autres que l'éditeur de logiciel - les chercheurs individuels, les firmes de sécurité, les organismes publiques, etc. - , contribuent de manière significative à améliorer la sécurité du logiciel et sont davantage impactés par des externalités telles que la divulgation publique de failles critiques.

Title: Essays on the economics of information security

Keywords: information security economics, market for vulnerabilities, software security, open innovation, industrial organization

Abstract:
This thesis aims at contributing empirically to the research field of information security economics, by referring to traditional tools and knowledge in economics especially in Industrial Organization. It focuses on new and evolving elements in the cybersecurity environment such as the use of free software revenue models in digital markets (Chapter 1), the introduction of crowdsourcing mechanisms to improve software security (Chapter 2), or the increasing involvement of third parties in software security (Chapter 3). I am particularly interested in understanding the incentives of major actors that contribute to software security, such as software vendors, white-hat hackers, security firms, and other third parties.
The thesis is organized in three chapters, each addressing a separate research question. In a first chapter, I examine the impact of competition intensity on software vendors' security investment behavior. I study the case of a software at the center of Internet security, namely the web browser, in which the vendors derive their revenue from advertising and compete in quality. I find out that market concentration is not necessarily harmful to security provision: indeed, higher market concentration positively impacts vendors' responsiveness in patching vulnerabilities, although this effect is reduced when a vendor is too dominant. In a second chapter, I focus on the crowdsourcing mechanism of white-hat hackers, which is representative of the market for software vulnerabilities that capitalizes on third parties' contribution. I study how hackers' perception of the uncertainty to be rewarded, determined by the level of information a contest provides about the contractual terms, affects their participation and thus the efficiency of the contest. I show that the self-selection process of participants leads to a trade-off between more numerous, but less performant participants, and higher quality but fewer participants. In a third chapter, I examine how the disclosure of a critical vulnerability affects the contribution of software vendors and third parties in discovering new vulnerabilities. I find that third parties' overall contribution in improving software security is considerable and that their contribution is significantly affected by externalities such as the disclosure of a critical vulnerability.

Institut Polytechnique de Paris
91120 Palaiseau, France