Upgraded software and embedded improvements: A puzzle of user heterogeneity

Raviv Murciano-Goroff, Ran Zhuo, and Shane Greenstein

December 2023

Abstract

The rise in cyberattacks over the past two decades spurred interest in policies improving cybersecurity. One focus of that research is how to create incentives for software vendors to release updates fixing vulnerabilities in their software. An important consideration that has received far less attention in this literature is understanding if software users install available software updates promptly and the factors that may increase or decrease their responsiveness. In this paper, we empirically investigate the propensity of firms to install software updates on the servers running their websites. We compiled a dataset tracking the server software used by over 150,000 medium and large firms in the United States to host their websites between 2000 and 2018. Treating the discovery of security vulnerabilities in the server software as quasi-natural experiments, we examine if and when firms update their server software after the vendors of that software disclose vulnerabilities. We uncover widespread usage of software with severe security vulnerabilities, with nearly 76% of the firms analyzed forgoing installing software updates that fixed severe security vulnerabilities found in their software for at least six months after the release of such updates. Using hazard model analysis that accounts for firms having different organizational routines for updating, we document that usage of cloud-based platforms for hosting websites can decrease the time to installing updates, that technical complexity on sites slows updating, and that the disclosure of severe vulnerability fixes in software updates does not jolt firms into installing them. Finally, we discuss how the relative inattentiveness of firms to act on software update releases should be incorporated into the design of cybersecurity policies.

Murciano-Goroff: ravivmg@bu.edu, Boston University; Greenstein: sgreenstein@hbs.edu, Harvard Business School and NBER; Ran Zhuo: ranzhuo@umich.edu, University of Michigan. We are grateful to Kenji Nagahashi and Mark Graham from The Internet Archive for providing the data for this research, and appreciate feedback we received from Sam Ransbotham, Ashish Arora, Ivan Png, Min Jung Kim, and seminar participants at the University of Toronto, University of Illinois, Purdue, Wharton, and Harvard Business School. We also thank Harvard Business School, the Ewing Marion Kauffman Foundation for providing funding for this work. The authors are pleased to acknowledge that the computational work reported in this paper was performed on the Shared Computing Cluster, which Boston University's Research Computing Services administer. The authors take responsibility for all errors.

1. Introduction

Over the past twenty years, cyberattacks on companies have increased. An industry survey in 2021 of Chief Information Security Officers found that the vast majority saw an increase in disruptive cyberattacks within the past year (EY Americas, 2021). Many cyberattacks exploit vulnerabilities in the software running on servers, the computers that host companies' websites. These events garner media attention and spur calls for increased investment and diligence in cybersecurity from policymakers (Barrett, 2018). In response, strategy and information systems scholars have explored and analyzed policies aimed at improving cybersecurity.

Most of the policies considered in this literature emphasize providing incentives for software vendors to build secure software and to release software updates fixing security vulnerabilities, known as patches, in a timely manner. These policies include encouraging vendors to produce higher quality software by making software vendors liable for the cost that users face from installing patches (August and Tunca, 2011),[1] encouraging vendors to more quickly release updates and patches by mandating the public disclosure of security vulnerabilities (Arora, Telang, and Xu, 2008), and decreasing the potential for malicious actors to learn from the vulnerability disclosures and patches by carefully limiting the frequency vendors release patches and the amount of information that software vendors disclose about vulnerabilities (Rescorla, 2004; Mitra and Ransbotham, 2015).

With its focus on incentivizing software vendors to provide software updates, the current literature has largely taken for granted that software users will adopt and install these updates when available.[2] And yet, many cyberattacks are successful partly because companies neglected to install available software updates that addressed security vulnerabilities in the server software they were using (Ranger, 2019).[3] For example, in 2017, the UK's National Health Service fell

[1] August and Tunca (2011) study the provisioning of patches in an environment with profit-maximizing software vendors. In the setting we study, the software is open source and created by a collective of volunteers and a non-profit foundation. Thus, charging the "software vendor" for patch costs is not possible.
[2] Most of the papers cited anticipate that some portion of software users will install available updates. While the fraction of users that install these updates are sometimes a parameter of the models studied, to our knowledge, no other paper has attempted to estimate the heterogeneity in that parameter or the dynamics of how users install updates over time.
[3] Other examples include Bank of America's ATMs being out-of-service and Continental Airline's flight being cancelled due to cyberattacks exploiting unpatched vulnerabilities that were more than six months old (Baroudi Bloor, 2003).

victim to a cyberattack that exploited a vulnerability in their server software for which a software update had been available—but not installed—for over a month (Acronis International, 2017; Palmer, 2017). The cyberattack resulted in the cancellation of thousands of operations, including those of emergency patients. In the same year, a breach of Equifax, which exposed the private information of over 143 million individuals, occurred when hackers exploited a vulnerability in the server software hosting the Equifax website for which a patch had been available for two months (Goodin, 2017). In 2018, the city of Atlanta suffered a hacking incident, halting many of the city's departments and operations. Following the incident, an audit found 1,500 to 2,000 vulnerabilities in the city's system, with some of the vulnerabilities being present in their system since almost a year prior (Harvey, 2018; Goldenberg and Zlatev, 2022).

Furthermore, many of the cybersecurity policies that create incentives for software vendors to release patches are designed according to expectations about the rate at which software users will install those updates. Given that malicious actors are given a running start from the time a vulnerability is announced to when a firm implements the update, mandating the disclosure of software vulnerabilities to incentivize software vendors to release updates more quickly may ultimately be detrimental to cybersecurity if firms are unlikely to install the updates promptly (Arora et al., 2006; Choi et al., 2010).[4] Therefore, to fully analyze and optimize cybersecurity policies, we need an understanding of software users' decisions regarding software updates.

To do so, in this paper we empirically investigate the extent to which companies install available software updates on their servers, the extent to which companies leave their web server software insecure by forgoing available software updates, and why some companies are faster at adopting and installing software updates than others. Leveraging detailed data on the server software used to host the websites of over 150,000 U.S. medium to large companies and organizations between 2000 and 2018, we analyze the frequency and timing of when those organizations installed available software updates to their server software. We focus on

[4] Arora, Nandkumar, and Telang (2006) is a good example of the need for empirical evidence of software users' uptake of updates. In their study considering policies related to the release of patches, they note that the release of software patches may increase the exploitation of vulnerabilities if software users do not install the updates. In the case where users are inattentive to updates, releasing fewer or more limited software updates may be optimal. By providing evidence on software user adoption of patches, we help fill the literature gap identified in their paper.

organizations that use the Apache web server, the most popular web server software deployed in use globally, for hosting their websites.

Web server software offers a valuable lens for understanding software improvements and cybersecurity. Web servers are ubiquitous and critical to the modern web-based commercial Internet. Millions of firms in the United States and hundreds of millions across the globe use web servers to support billions of web pages, including those serving sensitive financial and personal information (Greenstein and Nagle, 2014). Therefore, highlighting the prevalence of vulnerabilities and the instigators of installing updates can improve cybersecurity. Besides the cybersecurity benefits, many server software updates include features and bug fixes that improve the experience of Internet users. Therefore, increasing the speed that companies adopt software updates could also support economic growth and an improved user experience.

The Apache Web Server, the focus of this study, is the most popular web server software globally and is used by millions of businesses to support their websites. Besides its importance because of its widespread use, Apache is ideal for our study because detailed information about its versions, updates, and vulnerabilities is publicly available. During the period studied in this article, 28 severe vulnerabilities and 130 less severe vulnerabilities were discovered in the Apache software. Each vulnerability reported to Apache was scored along multiple dimensions that are visible to us. In addition, 115 software updates for the Apache server software were released during that same time. Each of these updates was released along with a list of the security vulnerabilities being corrected, the bugs fixed, and the new features included in the update. Finally, as Apache is open source, each update was made freely available to anyone online. Therefore, our analysis of the timing of installing these software updates is not confounded by the pricing of the software or the availability of its software updates.

Our unique dataset that tracks the server software was culled from raw data from the Internet Archive's Wayback Machine, which has routinely visited millions of websites every month and recorded the content and metadata about that site, including the name and version number of the server software hosting each website. By tracking the server software being used to host each organization's website over time, we can observe when an organization updates its server software and when it chooses to forgo updating and continue using the aging or vulnerable software.

Our analysis proceeds along three lines. First, we assess how common security vulnerabilities are in the server software of organizations using Apache. We find widespread use of server software with severe security vulnerabilities. Between 2000 and 2018, nearly 60% of the organizations in our dataset had a publicly disclosed security vulnerability on their server. In some months, such as October 2004, nearly every Apache webserver hosting the organization homepages in our sample operated with a publicly disclosed severe security vulnerability. This finding suggests that opportunities for cyberattacks are alarmingly numerous.

Second, we document the cross-sectional characteristics of organizations that are more likely to install an Apache server software update following its release using a sequential logit estimation procedure. We find that 76% of organizations with severe vulnerabilities in our data do not install readily available server software updates even six months after their release. After controlling for a variety of characteristics of the organizations, we find that few observables can account for the variation in organizations' response to released software updates. Instead, unobservable and persistent differences in organizations are the primary drivers of different approaches to updates.

Lastly, we explore the time dimension of our data and document the characteristics of organizations and attributes of the Apache software updates associated with organizations being faster or slower to install those updates over time. Using a survival model with time-varying covariates, we find that organizations with technically complex websites are slower to update when new versions are released. At the same time, those using cloud-hosting services are quicker. We also demonstrate that organizations are more deliberate in installing major version updates and updates filled with new features than minor updates or updates exclusively fixing security vulnerabilities.

Our work fills both a conceptual and an empirical gap in the cybersecurity literature. Previous works about cybersecurity and software updates have primarily focused on software vendors (Arora, Nandkumar, and Telang, 2006; Arora, Telang, and Xu, 2008; August and Tunca, 2011; Mookerjee et al., 2011; Mitra and Ransbotham, 2015). With some exceptions, the models presented in these papers treat software users as homogenous and portray software users' decisions regarding when to install updates as deterministic or a function of update quality. And yet, software vendors and users share responsibility for cybersecurity: vendors provision patches, but software users decide when to install those patches. Therefore, the behavior of software users regarding whether and when to install software updates can play an important role in determining how to optimize the cybersecurity policies considered in the previous literature. Indeed, in their study on the tradeoffs of mandating faster disclosure of software vulnerabilities, Arora, Nandkumar, and Telang (2006) acknowledged that releasing a patch could increase the number of cyberattacks and cited empirically understanding the factors that hasten or slow user-patching as a promising area for future research.

We contribute conceptually by describing and ultimately empirically estimating a simple model of companies' decisions regarding installing server software updates. Similar to the framework presented by Dey et al. (2015), our model builds on the long-noticed phenomenon that there is considerable heterogeneity in the regular updating cycle and firms' approaches to installing software updates (Arbaugh, Fithen, and McHugh, 2000). Some companies routinely update their server software, while others have a more ad-hoc approach to updating. In addition, among companies that wish to eventually adopt software updates, a variety of frictions and costs can cause delays in installing those updates (Baroudi Bloor, 2003; Dissanayake et al., 2022; August and Tunca, 2006; August et al., 2014; Kang, 2022).[5] Therefore, our paper describes a two-stage decision for companies: firms choose a routine with regard to updates—which could be to always update immediately, never update, or update after a waiting period—and those who choose to install updates, also decide how long to wait until they install a particular update. Unlike Dey et al. (2015), our model maps closely to a sequential logit setup, allowing us to empirically estimate which organizational characteristics are associated with different approaches to updating and which attributes of patches are associated with faster diffusion and adoption.

We also contribute by filling an empirical gap in the literature. Our analysis is one of the first to document and analyze the heterogeneity in server software-updating decisions across a wide range of companies and organizations over a long time period. Previous works recognized

[5] Kang (2022) emphasizes user incentives for upgrading enterprise software with many complements and the costs of accounting for such operational complexity. In a for-profit setting, August et al. 2014 investigate optimal trade-offs between the cloud-supported provision of upgrades or on-premise upgrades in the face of heterogeneous user valuation of quality. For-profit firms target their promotions to segments that demand low, medium, or high security, depending on the risks and costs of alternatives. Relatedly, August and Tunca, 2006 analyze incentives to patch in both a for-profit and free setting, assuming that upgrade behavior reflects forward-looking incentives but ignores externalities on others. In the for-profit environment, incentives to fix are too low, requiring vendor subsidies to induce optimal behavior. With freeware, the incentives are too low (high) when the risks and costs are minor (significant).

that firms did not immediately install patches after their release. For example, Arbaugh et al. (2000) found instances of servers being hacked using vulnerabilities patched two years prior. But that analysis was conditioned on successfully exploiting the vulnerability to detect that a vulnerability had not been patched. Our data allows us to see how many companies operate with vulnerabilities, putting them at-risk regardless of whether or not those vulnerabilities are ultimately exploited. Furthermore, to our knowledge, only two prior empirical research studies have examined longitudinal investment in cybersecurity and linked it to outcomes. Li et al. (2021) examined hospital adoption of security software and investment in related activities, while Liu et al. (2020) examined higher education and governance and associated actions. Both papers link these security investments to the propensity to suffer a security incident as well as exogenous organizational features and processes, using cross-sectional variance to infer causal determinants.[6] Our study takes a different approach, leveraging the quasi-random discovery of vulnerabilities and our longitudinal data to gain causal inference. In addition, our study looks at an extended nearly 20-year period, organizations across different industries, in addition to the attributes of the software updates themselves.[7]

The findings from our work also connect with the vast literature measuring the quantity and quality of IT and its impact on the productivity of organizations (Jorgensen, 2005; Jorgensen et al., 2016; Brynjolfsson and Hitt, 2003; Tambe and Hitt, 2012). Researchers have found persistent differences across organizations in the investments in and returns from IT (Foster et al., 2001; Aral et al., 2006).[8] Prior work (Murciano-Goroff et al., 2021) made a direct link with productivity analysis. It showed that high productivity correlates with high-quality web software, which arises at firms that upgrade more frequently and maintain software close to the frontier. In contrast, in the present study, we link upgrading behavior to a new valuable outcome, cybersecurity vulnerabilities, which has yet to be a focus of prior productivity analysis. Similar to patterns identified in previous research, we find persistent differences across firms in IT investment behavior, which motivates analyzing the determinants of firm heterogeneity in

[6] Li (2021) stresses the returns at organizations that invest in on-premises processes, such as anti-virus, intrusion detection, and authentication. Liu (2020) found behavior consistent with a tradeoff between granting autonomy and flexibility in using information systems and enforcing standardized, organization-wide security protocols—the more complex the computing environment, the higher the returns on centralized governance that limits vulnerabilities.
[7] This include the number of new features, bug fixes, severe and non-severe vulnerabilities fixed in each update.
[8] Some of this variance can be explained by the differences in returns related to the size and scale of implementations and strategic investments that enable leadership persistence (McElheran, 2015; Besen and Righi, 2019; Tambe et al., 2020; Zolas et al., 2020).

investment in software quality. Unlike that work, we focus our analysis on user responsiveness to announcements, the availability of patches, and the causal determinants of the variance in upgrade activity.

Finally, our results have important implications for companies as well as policymakers. In the final section of this article, we discuss some of those implications. In particular, given that the majority of firms are predominantly inattentive to vulnerability disclosures, releasing software updates with less information about the vulnerabilities inside may be socially efficient. Furthermore, we argue that more attention should be paid within firms to how technical complexity can inhibit firms from staying secure. Automating updating, such as services provided through cloud providers, could help.

2. Setting

In this section, we describe how software vendors receive reports of vulnerabilities and create software updates to fix those vulnerabilities. We also provide a simple model of how software users respond to the availability of new software updates, including the decisions of whether to install software updates and, if installing them, how quickly to adopt these software updates.

2.1. Reported, Disclosed, and Fixed Vulnerabilities

Our study examines organizations using server software, a particular type of computer program that enables users to host a website. When an individual visits an organization's website, the individual's computer sends a request to that organization's server. The server processes the request using server software that determines which content to send back to the individual. For example, after an individual connects to Amazon.com, the Amazon server software determines which products and prices to display to that individual. Similarly, after an individual connects to their bank's website, the server software determines who the individual is, what information they should or should not have access to from the bank's database, and what transactions that user should be allowed or not allowed to make.

Because server software plays these critical roles in guarding and transmitting sensitive data, server software vendors have developed procedures for finding and fixing security vulnerabilities. Software vendors typically accept reports from users about bugs and potential security vulnerabilities. Teams of security experts vet these submissions, known as *reported vulnerabilities*. Many software vendors also submit these bugs to the National Institute of Standards and Technology (NIST) to be scored based on the potential of that vulnerability to harm users.[9] The bugs that score "high" for their impact on the security of software and for their exploitability we call *severe security* bugs. These bugs harm an organization's system's quality and security for two reasons. First, these bugs are easily exploitable. According to the scoring system, most severe security bugs do not require local access to the system to perform the attack; attackers can perform the attack over the network and often need no or little authentication for accessing and exploiting the vulnerability. Moreover, once exploited, these bugs can result in significant losses. These include and are not limited to a partial or total disclosure of user information, a modification of some or all of the system's files, reduced performance, or a complete system shutdown (Mell et al., 2007).

After evaluating a reported vulnerability, software vendors decide when to disclose that vulnerability to the public. Vendors often initially keep reported vulnerabilities secret from the public, so malicious actors are not tipped off about their existence. When software vendors believe it is prudent to do so, they publicly acknowledge the vulnerability. We refer to these as *disclosed vulnerabilities*. Vendors disclose these vulnerabilities when it is essential to warn their users about security risks to encourage them to monitor their systems more carefully or to take mitigating actions, such as updating their software. Finally, software vendors develop, and release updates that fix the vulnerabilities, and software users decide whether and when to install it.[10] At that point, the vulnerabilities are called *fixed vulnerabilities*.[11] To decrease the probability that malicious actors exploit a bug, software vendors often release software updates and disclose vulnerabilities simultaneously.

[9] The National Institute of Standards and Technology (NIST) maintains the National Vulnerabilities Database (NVD). When a bug is reported, it is entered in the NVD, and a score is computed based on the Common Vulnerability Scoring System (CVSS). https://nvd.nist.gov/vuln-metrics/cvss#
[10] While this is the process that most software vendors hope will occur, some bugs are discovered and handled outside this procedure. In particular, some bugs are discovered when a user notices and discusses problems with the program without knowing the situation, indicating an underlying vulnerability. In those cases, the date the bug is discovered and the date the vendor publicly acknowledges the vulnerability may be the same. In addition, the vendor may acknowledge the bug before a software update is ready to be released. In addition, proprietary server software, such as the Microsoft IIS server software, automatically "pushes" updates to some users.
[11] The statement of this process is available on the website of the Apache Software Foundation Security Team at https://www.apache.org/security/. To the best of our knowledge, the overview of this process has stayed the same since the early days of the Apache Software Foundation.

We focus our analysis on the web server software, Apache, and the organization that supports it, the Apache Foundation, who follows the above outlined process for handling reports of bugs and security vulnerabilities. The Apache software descended from the first web server, built on the new diffusing World Wide Web, released by Tim Berners-Lee in 1991. In 1993, the National Center for Supercomputing Applications (NCSA) at the University of Illinois developed a computer program called the NCSA HTTPd server. The HTTPd server software supported sharing content on the web through the Hypertext Transfer Protocol (HTTP). NCSA made HTTPd available as shareware within academic and research settings, along with the underlying code. HTTPd's adoption spread quickly, partly because the servers did not restrict the usage or modification of the software. Many web administrators took advantage by adding improvements as needed or communicating with the lead programmer, Robert McCool, who coordinated adding and releasing extensions. When, in the spring of 1994, McCool left the University to become one of the first ten employees of the newly founded Netscape, the development of the web server software fragmented with eight different teams working on eight distinct vintages of the software. In 1995, the eight teams decided to coordinate their efforts into one server known as Apache (ostensibly because it was "a patchy web server"). The University of Illinois then transferred the server software development to the Apache organization without licensing or restrictions. Apache subsequently grew in popularity as the commercial internet grew, becoming widely used. Today, the Apache Foundation coordinates the development of the Apache server software, receives reports of vulnerabilities, and orchestrates the disclosure of vulnerabilities and the release of software updates to mitigate those vulnerabilities.

2.2. Server Software Users' Response to Software Updates

In this section, we sketch a simple model of organizations' response to the release of software updates fixing security vulnerabilities in server software.

When a software update becomes available for the server software that an organization uses to host their website, the organization chooses whether or not to install that update.[12] Drawing on prior research, we posit that each organization chooses a baseline regarding security

[12] Organizations employ technical staff to maintain IT infrastructure, including the server software for hosting the organizations' websites. IT decisions are typically made by individual server administrators within organizational contexts. Decisions about installing, updating, and changing server software are an interplay of individual and organizational-level factors. This paper discusses server administrators and organizations interchangeably as decision-makers regarding server software.

and software updating practices (Dey et al., 2015; Dissanayake et al., 2022). For some firms, this baseline may be to keep their software at the technological and security frontier by immediately installing updates and security patches as they become available. These firms we dub the *Frontier Chasers* (FC). Other firms choose a baseline rate of update that largely ignores available software updates. We call these firms the *Do-Nothing* (DN) firms. Once firms opt into either an FC or DN updating routine, their behavior when software updates become available is deterministic.

Firms that opt not to install updates immediately after their release and yet do intend to install the updates eventually, we call *Do-Something* (DS) organizations. These organizations must decide how long to wait following the release of a software update until their organization installs the update.

Which routine an organization chooses—FC, DN, or DS—may be a function of many complex factors, including organizational capital, management practices, and cybersecurity human capital. Furthermore, the organization's geographic location may influence local labor market conditions and the costs of hiring labor to operate a process.

Costs and benefits influence both the baseline updating routine of an organization and the responsiveness of an organization to the release of software updates.[13]

The costs affiliated with installing software updates may derive from the frictions that the organizations encounter when updating. For example, if administrators must multitask across a large set of operational needs, then updating the Apache server may involve time and high opportunity costs (Dissanayake, et al, 2022). If an organization has been operating for longer, administrators may inherit configurations that accumulate myopic decisions from the past. If, however, a server is hosted on a cloud provider, such as Amazon Web Services (AWS), administrators may find it less costly to manage the transition from one software version to another. In addition, the technical complexity of an organization's IT operations is likely to be associated with organizations being less responsive to changes in known vulnerabilities in software. Websites and other IT infrastructures that depend on large numbers of interconnected

[13] This conceptualization parallels the rational inattention framework applied to organizations (Matějka and McKay, 2015). Much like that literature, organizations have a baseline propensity towards an action, updating. In addition, each organization has factors that influence their attentiveness to new information, namely responsiveness to the presence of bugs in their software and the release of new software versions.

or interdependent technologies are more challenging to update.[14] They require more forethought and planning to avoid introducing incompatibilities when transitioning software versions. Because of that friction, more technically complex IT infrastructures are less likely to respond quickly to discover vulnerabilities.[15]

As for the benefits from installing software updates, these may include direct benefits, such as faster loading times for their site or the ability to display and deploy frontier web apps that engage users. For example, deploying servers to support Web 2.0 applications may motivate upgrades. These updates may also decrease the probability of security breaches. The prominence of an organization's homepage is also likely to increase the benefits of being responsive to the discovery of cybersecurity vulnerabilities. Highly trafficked websites are also more likely to be targeted by hackers. Finally, the ability of a website to conduct e-commerce and financial transactions also correlates with the benefits of updating after the announcement of cybersecurity vulnerabilities in software. Cybercriminals are likely to target sites that collect and store financial records. So, server administrators at organizations with such sensitive data are also more likely to watch carefully for updates patching security vulnerabilities.

Figure 1 summarizes our basic model of the sequential decisions of firms following the release of a server software update.

3. Data and Sample Construction

3.1. Key samples and data sources

We combine two data sources to construct our first key sample, a broad panel that tracks Apache server software used by medium to large organizations in the United States between 2000 and 2018. This sample is suitable for a census of usage and vulnerabilities in the user base over time.

[14] If updates have the potential to disrupt organizations' servers, then organizations with lower probabilities of being targeted by cyberattacks may feel inclined to delay installing updates until other organizations have attempted to do so. In this way, organizations may be free-riding on other organizations being first-movers and figuring out how to mitigate any negative impacts from installing updates (Hui-Wen and Png, 1994).

[15] Rather than attempting to install all available updates and patches, organizations may attempt to prioritize vulnerabilities with higher exploitability. Previous research has provided models of what such a prioritization strategy could be (Jacobs et al, 2020). In Appendix A4, we examine if our results appear different when focusing exclusively on highly exploitable vulnerabilities.

To construct this sample, we first collected information on all organizations in the Bureau van Dijk Mint Global database with at least 50 employees located in the United States, and listing a website. For each organization in this database, the data provides information about the estimated number of employees, the industry of the organization, and the geographic region in which the organization operates.[16]

We then found and extracted information on the web server software used by each of the organizations in our Mint Global data in each month between 2000 and 2018 from the Internet Archive (IA) Wayback Machine. The IA is a non-profit organization that has routinely scanned millions of publicly facing websites for the past two decades and taken snapshots of the content on those sites. When connecting to a website, the server software that hosts the site will respond with both the content of the site as well as metadata about the server and software hosting the website. This metadata often contains the vendor of the server software hosting a site and the server software version number (e.g., Apache 1.3.6).[17] The responding server also communicates its IP address, which is a sequence of numbers that indicates where the server hosting a website was located. The IA collected and stored the metadata and IP addresses for every website they scanned, and every time they scanned each site. Using the list of organizations and their web addresses from Mint Global, we found the associated sites in the IA data, extracted the metadata from each time the IA connected with that site, and parsed the metadata for the server software and version numbers. In addition, we also examined the IP addresses of each website, as recorded in the IA data. Using a list of all IP addresses associated with Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform, we flag if the servers hosting an organization's website were located on the cloud at a given time.

We combine these two sources of raw data into a broad, monthly panel dataset of Apache web server users by keeping observations of organization-months that used Apache web server software. In total, this panel contains close to 5 million organization-month observations of

[16] An organization in our dataset is mapped to a website domain. If two organizations have the same website domain, they are treated as part of the same organization.

[17] Users have a choice regarding how much information their server response headers show about their web servers, ranging from showing the full information including the vendor, version, and operating system to showing no information at all. Setting anything less than showing the server vendor and version is not recommended. As the Apache Foundation puts it, "…[Obscuring server header] makes it more difficult to debug interoperational problems. Also note that disabling the Server header does nothing at all to make your server more secure. The idea of 'security through obscurity' is a myth and leads to a false sense of safety." See: https://httpd.apache.org/docs/2.4/mod/core.html#servertokens.

Apache server usage from 150,854 organizations between 2000 and 2018. This panel corresponds to the data item shown in Table 1 Row (i)(a).

To enable our analysis of user decisions, we need to be able to precisely determine the time of update, we therefore construct a restricted panel of Apache users that were frequently captured by the IA from our broad panel. This restricted panel will allow us to determine when an organization installed a new version based on when the server version number changes between adjacent observations. For this panel, we keep the subset of organizations from the broad panel that had their website scanned by the IA at least every three months on average. The IA was not able to scan every website every single month. By focusing on organizations that have their website scanned regularly, we can reasonably precisely observe the time when an organization installs a software update. We include only organizations with at least ten months of observed Apache usage. As Apache makes a new minor release every 4.5 months on average, the restriction ensures that we observe organizations' updating decisions over at least two average software release cycles. We also drop an observation if the specific scan of the website by the IA and the following scan for that website are more than 4.5 months apart. Finally, we do not include the first observed Apache updating cycle and the last observation of Apache usage of an organization to adjust for censoring observations of a website. After these restrictions, our restricted panel of Apache users with frequent captures contains around three million organization-month observations from 70,092 U.S. organizations between 2000 and 2018. This panel corresponds to the data item in Table 1 Row (i)(b).

Given that we can precisely detect the time software update happens in our restricted panel, we can construct several different outcome variables for this dataset. The first one is the variable $updated_{it}$, which is a binary that equals 1 if organization i makes an update in month t. An update happens in month t either when the organization uses a higher major version in the next observed month[18] or when the organization uses a higher minor version in the next observed month.[19] Another outcome variable is a "time-to-event" variable $timeToUpdateMonthEnd_{it}$, which captures the number of months since organization i's previous update of Apache server software.

[18] E.g. the organization uses Apache 2.2 then changes to Apache 2.4.
[19] E.g. the organization uses Apache 2.2.10 then changes to Apache.2.2.11.

Moreover, we can construct variables that capture an organization's historical updating behavior. We construct $infrequentUpdater_i$ and $frequentUpdater_i$, which are binary variables that capture how frequently an organization i updated its Apache server software in its first nine observed months. If it did not update at all, the variable $infrequentUpdater_i$ is equal to 1. If it updated twice or more, the variable $frequentUpdater_i$ is equal to 1.[20]

For each observation in our broad and restricted panels, we match the Apache server software version an organization's website is using with information about that software version from the Apache Software Foundation and the National Vulnerability Database (NVD). The Apache Foundation and the NVD contain detailed information about each version of Apache software and every vulnerability in each version. This includes the degree and severity of security vulnerabilities, the date vulnerabilities were reported to the Apache Foundation, the date those vulnerabilities were disclosed to the public, and the date of the release of new versions that fix each vulnerability. For each organization-month observation in our panel, we check if the server software version contained a *reported*, *disclosed*, or *fixed* severe security vulnerability at that time. We also checked if a newer server software version had been released and if that version was considered a major or a minor update. Finally, we also parse the Apache software "changelogs," which are documents summarizing the changes made in a software version update for the number of severe and non-severe security vulnerabilities fixed and the number of new and improved features added to the software. The information we obtained from the Apache Foundation and the NVD correspond to data items (ii)—(v) in Table 1.

Merging information from the Apache Foundation and the NVD to our panels allows us to construct a range of variables that describe the vulnerability and quality status of an organization's server software in a month. $hasSevereBugs_{it}$ captures whether the version of the software in use had publicly disclosed severe security vulnerabilities. $withinThreeMonthsSinceAddSevereBug_{it}$ is a binary which is equal to 1 if the version in use had additional severe security vulnerabilities disclosed within three months before month t. $newVersionAvailable_{it}$ captures whether newer versions of Apache server software, major or minor, to what organization i was using in month t, were available. $newMinorVersionAvailable_{it}$ captures whether newer minor versions of Apache server

[20] The two variables are undefined for the first nine observations of Apache usage of each organization because those observations are used to construct the variables.

software were available. $severeBugsFixed_{it}$ captures the number of severe security vulnerabilities the *oldest* newer minor version (if one was available) fixed. If no newer minor version was available for organization i in month t, this variable is set to zero. Similarly, $nonsevereBugsFixed_{it}$ captures the number of nonsevere bugs the *oldest* newer minor version fixed, and $featureChanges_{it}$ captures the number of feature changes the *oldest* newer minor version has, if such a version was available.

Building on our restricted panel, we also create a cross-sectional dataset that we call the release window cross section. This dataset tracks the decisions of organizations during six-month windows following the releases of Apache software updates that fixed severe security vulnerabilities. An observation in this dataset is an organization during one of these six-month windows. The Apache version in use at this organization at the beginning of the six-month window must have severe security vulnerabilities that the new release fixed. The dataset contains 161,106 observations for 55,558 organizations during 15 unique six-month release windows.[21] The outcome of interest from this dataset is whether the organization, faced with the release of an update that fixed severe vulnerabilities in its used version, chose to chase the frontier, do something, or do nothing.[22] This cross-sectional dataset corresponds to the data item in Table 1 Row (vi).

3.2. Additional Covariates

We augment our key datasets with additional variables from various other data sources.

To gauge the amount of traffic an organization's website received, we obtained website traffic rankings from Alexa for the top one million websites each year between 2010 and 2018.

We match the organizations in our sample with the Harte Hanks database to capture the scale and complexity of an organization's IT operations. This database contains information such as the number of personal computers owned by the organization, the number of IT staff, the IT budget, and the software budget in 2017. We also manage to obtain, for a fraction of the

[21] We only include releases that fixed severe security vulnerabilities and that were at least six months apart from the next such releases to enable us to observe organizations' decisions with respect to the given releases over a reasonably long duration.
[22] We define the frontier chasers to be organizations who updated within two months following the release, do-something organizations to be those who updated within six months, and do-nothing organizations to be those who did not update within six months.

organizations, whether they have outsourced their IT operations sometime between 2005 and 2009.

For a subset of organization websites in our data, we also have data on the technologies used in these organizations' websites between 2016 and 2018. We collected this data from the HTTP Archive, an organization that takes snapshots of websites and analyzes them for their technical attributes. Specifically, the HTTP Archive runs an open-source tool developed by the company Wappalyzer on each website, which flags the technologies being used. The data capture both technology categories that contribute to the basic architecture of websites, such as web frameworks and databases, and technology categories that support monetization and e-commerce, such as marketing automation and payment processors. This data helps us construct proxies for websites' technical complexity and the organizations' intent to monetize their websites. We conceptualize the number of web technologies as being correlated with the technical complexity of the website and, thus, the technical complexity when attempting to update the site's web server. We think monetization tools indicate that security vulnerabilities would be particularly damaging and that significant benefits exist to keep the website secure. This data is available for 24,263 organizations in our dataset.

To study whether disclosures of data breaches by organizations in the same geography or the same industry prompt organizations to improve and secure their own software, we also obtain data on breach disclosures from Privacy Rights Clearinghouse's Data Breach Chronology, the best publicly available data source for breaches to our knowledge. Data Breach Chronology compiles breach disclosures from various sources, including the media, state attorney generals' offices, and other online trackers for breach disclosures. This data contains 2,366 disclosed data breaches between 2005 and 2018 in all fifty US states and Washington D.C. Each disclosure includes the reporting organization, the date of disclosure, and the state and broad industry category of the organization's operations. For 1,490 (63%) of these, the organizations also reported the number of affected personal records.

Because our information about organizations from Mint Global contains data from only one year, 2018, we also repeat some of our analysis for publicly traded firms. Publicly traded companies make available data on an annual basis about their operations. Therefore, we focus on publicly traded firms for some of our analysis. Our data for those firms come from Compustat and covers the full panel of U.S. public firms every year. This data contains a wide range of organization characteristics such as the total assets, capital expenditure, cash flow, and income, allowing us to examine organization characteristics that might affect updating decisions deeply. The data's time dimension also enables us to study the effects of changes in financials within organizations. A drawback to using this data source is a significant sample size reduction, given that only a tiny fraction of organizations in our sample are public firms.

Table 1 enumerates and summarizes all the data sources used in our analysis. Table 2 shows summary statistics for the restricted panel of web server usage in Table 1 Row (i)(b) merged to the various organization and website characteristics. Appendix Table A1 shows detailed variable definitions. Appendix Table A2 displays the correlation of various measures used in our analysis.

4. Empirical Framework

The first component of our empirical analysis is descriptive, where we document the extent and distribution of security vulnerabilities in the Apache server software being used by organizations over time. To perform this analysis, we use our broad panel data with almost five million observations from 150,854 organizations, described in Row (i)(a) of Table 1. We examine the proportion of organizations using Apache server software with severe security vulnerabilities and illustrate changes over time and heterogeneity across the industry, geography, and firm size. We show these results in Sections 5.1—5.3.

4.1. Sequential logit analysis for cross-sectional determinants of user decisions

Based on the empirical evidence we find from the above exercise, we then investigate the characteristics of organizations associated with better responsiveness to vulnerability disclosure and software improvement. We focus on examining which organizations updated quickly (FC), slowly (DS), versus did not update at all (DN) during the six-month windows following new releases that fixed severe security vulnerabilities.

Using the Release Window cross-sectional dataset, we estimate an empirical model that mirrors the model presented in Figure 1. Specifically, we estimate a sequential logit model with two stages of choice: In the first stage, an organization decides whether to update to the new version that fixes its severe security vulnerabilities immediately within two months (FC) versus to delay (DS or DN). If the organization decides to update immediately, its decision outcome is

FC. If the organization chooses to delay, it enters the second stage, where it decides whether to do something within the extended observation window of six months (DS) versus to do nothing (DN).

The sequential logit regression models each stage of choice separately using a standard logit decision model for the subsample of organizations that are "at risk" of making the decisions. For the first stage of the decision, with FC as the base outcome, the probability of updating immediately and choosing FC is specified as follows:

$$P_{iw}(choosingFC) = \frac{1}{\left(1+exp\left(b_{10}+b_{11}X_{1,iw}+\cdots+b_{1j}X_{j,iw}\right)\right)} \qquad (1)$$

where i denotes the organization and w denotes the release window for the observation. $X_{j,iw}$ includes characteristics of the organization.[23]

If an organization chooses not to respond immediately, it enters the second stage. With DS as the base outcome, the probability of choosing DS conditioning on entering the second stage is the following:

$$P_{iw}(choosingDS|notchoosingFC) = \frac{1}{\left(1+exp\left(b_{20}+b_{21}X_{1,iw}+\cdots+b_{2j}X_{j,iw}\right)\right)} \qquad (2)$$

And the probability of choosing DN conditioning upon entering the second stage is the following:

$$P_{iw}(choosingDN|notchoosingFC) = \frac{exp\left(b_{20}+b_{21}X_{1,iw}+\cdots+b_{2j}X_{j,iw}\right)}{\left(1+exp\left(b_{20}+b_{21}X_{1,iw}+\cdots+b_{2j}X_{j,iw}\right)\right.} \qquad (3)$$

By examining the coefficients on the components of $X_{j,iw}$, we can understand which organizational characteristics are associated with the organization having an FC, DS, or DN approach. We show these results in Section 5.4.

4.2. Survival analysis for within-organization, time-varying determinants of user decisions

The sequential logit model is helpful for shedding light on the determinants of user decisions in response to releases at the cross section. However, the model does not permit one to

[23] If some characteristics of an organization change within the six-month release window w, we use in the regressions the values at the beginning month of the release window.

vary an organization's characteristics within each release window and only uses cross-sectional variations and data. As a result, if there are organizational characteristics or other covariates that vary within organization over time that influence how soon an organization updates, the model does not capture these effects. Our last piece of empirical analysis, therefore, utilizes the time dimension of our restricted panel to shed light on the time-varying organizational characteristics and attributes of software update releases that instigate or impede the rate that organizations install those updates.

Because the outcome of interest in this analysis is a time-to-event variable—the amount of time since the organization's previous update to the organization's next update, we estimate survival models with time-varying covariates to study what predicts updating faster or slower. The standard Cox proportional hazards regression model with time-varying covariates is as follows:

$$h(t) = h_0(t)exp\left(b_1 X_{1t} + b_2 X_{2t} + \cdots + b_j X_{jt}\right) \qquad (4)$$

where $h(t)$ is the hazard function, representing the expected monthly updates given that the version has continued usage for t months. $h_0(t)$ is the baseline hazard and represents the hazard when all the independent variables $X_{1t}, \dots, X_{jt}$ are equal to zero.[24]

While the above model allows us to explain differences in the rates that organizations update their server software using the observed time-varying organization and environmental attributes, there is the possibility of remaining unobserved organizational routines around updating. This raises a methodological concern. Specifically, the model in Equation (4) assumes that the baseline hazard $h_0(t)$ is identical across organizations. However, as we will show later in the Results section, we have discovered considerable heterogeneity across the organization in updating decisions that observable covariates cannot explain in our cross-sectional regressions. Not accounting for the unobserved differences in the organizations' baseline hazard, or

[24] Implementation-wise, the Cox model with time-varying covariates requires two distinct variables to keep track of the time-to-update. For example, suppose a software updating cycle consists of three months and there are covariates that vary at the monthly frequency, then this updating cycle would make three separate monthly observations. The first observation has a starting time-to-update at month 0 and an ending time-to-update at month 1. It is tied to the combination of covariate values in the first month. The second observation has a starting time-to-update at month 1 and an ending time-to-update at month 2. It is tied to the combination of covariate values in the second month. The third observation has a starting time-to-update at month 2 ending time-to-update at month 3. It is tied to the combination of covariate values in the third month. The event equals 1 ("updated") only for the third observation.

propensity in updating would give rise to a reverse causality issue. For example, organizations that lag in updating their Apache server software are more likely to use older versions that accumulate more severe security vulnerabilities over time. If we assumed that all organizations have the same baseline hazard and estimated a Cox model, the estimated coefficient on $hasSevereBugs_{it}$ would be biased downward, meaning that we are more likely to find a negative effect of severe security vulnerabilities on the organizations' decisions to update.

Therefore, to control for unobservable differences in different organizations' baseline hazards, our preferred specification is a stratified Cox model:

$$h_i(t) = h_{0i}(t)exp\left(b_1X_{1t} + b_2X_{2t} + \cdots + b_jX_{jt}\right) \qquad (5)$$

where the i subscript on the hazard functions denotes stratum organization i. Appendix A3 discusses additional justifications for stratifying at the organizational level. Under this model, the effect of a time-varying variable is identified by the changes to that variable within an organization. To identify the heterogeneity in the effect of the time-varying variable across organizations, one can easily do that by interacting the time-varying variable with organization characteristics. Since the stratified Cox model absorbs the unobserved differences in the baseline updating rate across organizations, the interaction terms isolate the observed factors associated with differences in organizations' speed of updating.

In our analysis, we include a variety of covariates in $\boldsymbol{X_{it}}$. One primary time-varying variable of interest, $hasSevereBugs_{it}$ captures whether the version of the software in use had publicly disclosed severe security vulnerabilities. Additional time-varying variables of interest are $newMinorVersionAvailable_{it}$ and $newVersionAvailable_{it}$. Including the two variables serves two purposes. First, we want to investigate how much organizations responded to vulnerabilities versus releases of new versions. Second, if no new versions are available for organizations to update when severe vulnerabilities are disclosed, organizations' inaction is not due to a response failure. If we do not control for the availability of new versions, major or minor, we will underestimate the effect of disclosures of vulnerabilities on organizations' updating decisions.

We think of the discovery, reporting, and disclosures of severe security vulnerabilities and the releases of new versions to be plausibly exogenous timing to individual organizations' IT staff. Though testing exogeneity assumptions formally is generally difficult, our assumption in this setting is intuitive. The vast majority of web developers and IT professionals are not directly involved in developing the Apache server software and its vulnerability disclosure decisions. However, they can potentially influence the process through bug discovery and reporting. IT professionals discover bugs in the software when they interact with it; for example, when they use the software to build applications or develop new features of the software itself—the chances of finding bugs increase when the total interaction with the software increases. However, as many individuals and organizations around the work worldwide on the Apache web server, changes in the activities of any particular organization (other than the Apache Foundation) are tiny compared to the total amount of interaction. To confirm that individual organizations' actions do not have an outsized impact on Apache server software's bug handling and release process, we inspect our data on who was credited with discovering or reporting each vulnerability to the Apache Software Foundation. Other than the staff at the Apache Software Foundation Security Team, we find almost no overlap in the names and organizational affiliations of the reporters.

To understand what factors are associated with organizations being more or less responsive to the disclosure of security vulnerabilities and the availability of new software versions during their IT operation cycles, we include interactions between the above time-varying variables and organization and environmental attributes in some specifications of our empirical analysis.

We show results from our survival analysis in Section 5.5.

5. Results

5.1. Prevalence of Security Vulnerabilities

In our broad panel, we begin by documenting the extent and distribution of security vulnerabilities in the Apache server software being used by organizations between 2001 and 2018. We focus on organizations operating with vulnerabilities under three different scenarios: 1) severe security vulnerabilities that have been reported to Apache, 2) severe security vulnerabilities that are publicly disclosed, and 3) severe security vulnerabilities that Apache had previously released a patch fixing.

Figure 2(a) shows the fraction of organizations using Apache versions with reported severe security vulnerabilities. Organizations with these vulnerabilities would be susceptible to potential attacks. The figure shows that a considerable fraction of organizations' servers operated with these vulnerabilities throughout our sample. Over the course of the almost 20 years in our sample, the fraction of firms operating with these vulnerabilities was smallest in November 2015 at around 30%. The fraction is the highest, at almost 100%, in December 2015 when a critical security vulnerability in a module that assigns content metadata to the content selected for an HTTP response was discovered.

Figure 2(b) shows the fraction of organizations using Apache versions with publicly disclosed severe security vulnerabilities. Malicious parties could easily utilize information about disclosed vulnerabilities to find and target organizations using server software with these vulnerabilities. The fraction of organizations using Apache versions with publicly disclosed severe security vulnerabilities was above 19% in all months we observed. The fraction peaked in October 2004 at 98% when several severe security vulnerabilities affecting Apache 2.0.x became public in the latter half of 2004.

One might wonder if security vulnerabilities are primarily found on the websites of incumbent firms while new entrants avoid operating with vulnerabilities by selecting newer and more secure server software. Figures 2(a) and 2(b) display the percentage of firms with reported and disclosed security vulnerabilities separately for incumbent and new organizations. Both figures show that new organizations were better at adopting more up-to-date, secure software versions by a small margin. Unlike incumbent firms, new entrants are unburdened by technical investments of the past. Therefore, entrants may be able to install updates and new versions of software more easily and maybe more discerning and aware of recent releases of software versions.

Figure 3 shows the fraction of organizations using Apache versions with one or more severe security vulnerabilities that were publicly disclosed and already fixed in newer versions of the Apache server software. Alarmingly, except for the early period until March 2002, more than 19% of organizations in all months of our sample used Apache versions with severe security vulnerabilities already fixed in newer versions. In addition, this plot documents that a significant fraction of organizations operate server software with multiple severe security vulnerabilities, providing malicious actors numerous avenues for attacking these organizations' websites.

Given the prevalence of using vulnerable server software, these figures alarmingly demonstrate that hackers would have no trouble finding servers to exploit.

5.2. Responding to Updates

Closer scrutiny of Figure 3 allows us to discern organizations' responses to releases of new versions that fixed severe security vulnerabilities. Figure 3 plots the release dates of the fixes, represented by the vertical grey lines. This allows us to gauge both organizations' propensity to install updates and the speed of response to those fixes. Organizations' response to these releases is slow and unthorough. A significant fraction of organizations did not respond to releases. For example, in July 2006, the Apache Foundation released Apache 1.3.37, Apache 2.0.59, and Apache 2.2.3, which fixed a vulnerability that allows remote attackers to cause a denial of service (application crash) and possibly execute arbitrary code. Many organizations, however, took months or even years before updating to those releases. The fraction of organizations using problematic versions of Apache peaked at 92% after the release, but the rate declined by only 1.6% per month for the next three years. By mid-2009, over 30% of organizations continued operating vulnerable versions of the Apache server software.

The red oval in Figure 3 shows the six-month window following the July 2006 release. Among the users impacted by the specific vulnerability and the release (70% of the user base), 16% acted as FC organizations and installed the available software update within two months. In addition, 12% of firms acted as DS organizations and updated within six months. The remaining 72% of DN organizations chose not to install the update even six months later.[25]

Repeating the above exercise, we can identify the FC, DS, and DN organizations in different six-month release windows in our data. Each release window corresponds to a release that fixed severe security vulnerabilities, and that was at least six months from the next such release. Overall, we have 161,106 observations of organization-release windows, which constitutes our release window cross-sectional dataset. We find from this dataset that for 12% of the observations, the organizations can be categorized as FC, 12% as DS, and 76% as DN.

[25] Note that we do not include in the above computation the 30% of the user base that either used Apache versions free from any public severe vulnerabilities or used versions not impacted by that specific vulnerability.

5.3. Heterogeneity in Responding to Updates

To what extent is the lack of response to released software updates fixing severe security vulnerabilities similar across organizations or isolated to particular subsets of organizations, such as incumbent firms, firms in specific industries, or firms in particular geographies?

Figure 4 explores the heterogeneity in these patterns across organizations and websites with different characteristics. In Figure 4 Panel (a), we split our sample into new and existing sites. If new users were more discerning when actively selecting a software product, they might select newer versions that fixed well-known severe vulnerabilities and avoid versions with those vulnerabilities. Indeed, in the plot, we show that a lower fraction of new websites installed problematic interpretations of the server software relative to existing sites.

Panel (b) examines the differences based on the organization's size. For example, smaller organizations might be more agile and able to update their server software faster due to a small coordination cost across their IT-related departments. Indeed, the plot shows that larger organizations more frequently maintained vulnerable software even when fixes were available, whereas smaller organizations were more inclined to switch to newer versions with fixes.

Panel (c) breaks down the organizations by the geographic region of the organization's headquarters. If different geographic regions have different levels of attentiveness regarding security issues in software—for example, if Silicon Valley-based firms more frequently hear about security bugs through informal networks—we might expect geographic variation in using vulnerable versus secure software. Furthermore, if the local labor market for security and engineering professions differs across regions, this could influence software updating decisions. In the plot, however, we find a relatively slight variation in software usage with and without severe security vulnerabilities across geography.

Panel (d) displays the variation across industries. We find some substantive differences in the software updating decisions. For example, organizations in the Health and Food & Accommodation industries less frequently maintain server software with severe vulnerabilities than those in the Finance and the broadly defined Information industries when fixes are available.

Figure 4 shows some heterogeneity. However, the main finding from these plots is that compromised Apache servers are prevalent across organizations at different stages of development, large and small, in all states and industries.

5.4. Explaining the Response to Updates

What explains why some firms respond to the release of updates by installing them and others largely ignore these updates? In this section, we investigate the cross-sectional determinants of user decisions regarding whether an organization is an FC, DS, or DN using estimates from the sequential logit regression model described in Equations (1)-(3) fitted to our release window dataset.

Table 3 reports the results of that estimation. In the first specification, we fit the model using our 161,106 observations of user decisions to explanatory variables that are relatively well populated. After matching the various data sources, we end up with 108,385 observations. In the second specification, we add variables that shed light on potentially significant margins of the decision process but are much less well populated. These variables include whether the organization has outsourced IT its operations, metrics for the technical complexity of its website, and whether the organization has monetized its website. This sample has 26,540 observations. Despite the very different sample sizes, the main patterns that emerge from the two regressions are very similar.

Historical updating behavior is a large and significant predictor of user decisions in response to releases of fixes to severe vulnerabilities. We include both the $infrequentUpdater_i$ and the $frequentUpdater_i$ variables in the sequential logit regressions. Therefore, the dropped category for historical updating behavior is those who updated once during the first nine months of observations.

The sequential logit coefficients for our larger sample in the first stage of decision, shown in Column (1), suggest that an organization that has historically been an infrequent updater has a $exp(0.167)$ =1.18 times the odds of the dropped category of delaying and becoming DN or DS holding other variables constant. In contrast, an organization that has historically been a frequent updater has a $exp(-0.239)$ =0.787 times the odds of the dropped category of delaying and becoming DN or DS. If an organization has moved to the second stage, shown in Column (2), the organization that has historically been an infrequent updater has a $exp(0.192)$ =1.21 times the odds of the dropped category of doing nothing. In contrast, a historically frequent updater has an $exp(-0.0987)$ =0.91 times the odds of the dropped category of doing nothing. The results from the smaller sample are qualitatively similar.

We also find that whether an organization operates its server software from the cloud is a large and significant predictor of responding to the releases, particularly in the second stage. As shown in Columns (2) and (4), an organization on the cloud has at most an $exp(-0.505)$ =0.60 times the odds an organization not on the cloud of doing nothing.

Moreover, data breaches that happened right before the release event window in the same broad industry category are correlated with a need for more response. This result may reflect that industries that persistently do a poor job of updating and keeping their software secure suffer from more cyberattacks. Shown in Columns (2) and (4), a 1% increase in the number of personal records affected in data breaches disclosed from the same broad industry category is associated with at least an $exp(0.01 * 0.0207)$ =0.02% increase in the odds of doing nothing. The effect is statistically significant, though economically small.

We find most of the other variables insignificant in predicting user responses. Notably, we find an organization's location, revenue, and whether it is public matters little. General measures of an organization's IT operations, such as the number of PCs and IT staff, also do not seem to matter for the cross-section. Even the technical complexity of the website and that the website was used for monetization did not generate fewer or more responses. Moreover, the constant term at both stages is the most significant predictor of user decisions. This indicates that unobserved characteristics primarily drive organizational decisions.

Our evidence so far points to organization-specific characteristics in the detail of its IT operations to result in persistent differences in responses. Some of those characteristics are observed, such as using cloud services like AWS. The remaining variation is mainly due to unobserved differences.

5.5. Speed of Response to Available Updates

Organizations forgo immediately installing updates must decide when they will install available updates in the future. This decision of how long to wait to install updates may be influenced by the characteristics of the organization as well as attributes of the software updates themselves. Many of those characteristics could be time-varying, whose effects our sequential logit analysis cannot capture. In this section, we investigate the determinants of the speed of updating by exploiting the variations of those time-varying variables in our restricted panel.

We begin by exploring the simple correlations between the time-varying organizational and software characteristics and the speed of installing software updates. Table 4 Column (1) shows the estimated Cox proportional hazard model represented in Equation (4). The estimated coefficient on the availability of a new software version is correlated with a higher hazard rate of updating. The coefficient on $hasSevereBugs_{it}$ is significantly negative, however, implying that running server software with security vulnerabilities correlates with a $1 - exp\ (-0.174)$=16.0% lower monthly updating rate. This correlation, however, suffers from reverse causality. Organizations that less frequently or more slowly update their server software are also likely to accumulate vulnerabilities in that software. In order to isolate the relationship between the presence of security vulnerabilities in software and the decision of organizations regarding when to update, we need to adjust for organizations' baseline propensity to update their server software.[26]

One way to control for organizations' baseline propensity for updating is to include variables that capture that propensity. Column (2) of Table 4 shows the results where we include a range of organization and software usage characteristics as controls, for example, organizations' historical updating patterns captured by the variables $infrequentUpdater_i$ and $frequentUpdater_i$, and the full sets of state and industry dummies. The coefficient on $hasSevereBugs_{it}$ continues to be negative and significant, indicating sizable unexplained heterogeneity in organizations' updating decisions, apparently not explained by organizations' observed characteristics.

A more flexible way to control for organizations' baseline propensity for updating is to stratify the Cox model at the organization level, as in Equation (5). Doing so allows each organization to have a completely different baseline hazard and overcomes the reverse causality issue. The stratification approach is similar to the organization's fixed effects in a linear regression model. We use the within-organization variations of the independent variables to study their impact on the updating decisions.

[26] We visualize the relationship between the rate of updating and the presence of severe security vulnerabilities in server software in Appendix Figure A2.

Column (3) of Table 4 shows the estimates of that model without additional controls. The stratification changes the estimated sign on $hasSevereBugs_{it}$. The coefficient is 0.067, both positive and significant at 1%. This implies that if an organization's Apache server software is disclosed to have severe security vulnerabilities, holding all else fixed, it is associated with an $exp(0.067) - 1 = 6.9\%$ increase in the organization's hazard rate of updating.

That the addition of controls in Columns (1)—(2) does not change the sign or significance of the coefficient on $hasSevereBugs_{it}$ while stratification by organization in Column (3) does is telling. Organizations' software updating behavior is more associated with unobservable organization characteristics than observed ones. Updating decisions largely cannot be explained by obvious factors, such as industry, geography, website traffic, usage of cloud services, or even technologies on site. Instead, organizations likely have operational differences in software updating habits for various idiosyncratic reasons.

The other rows of Table 4 Column (3) show that organizations are more responsive to the availability of new software than the announcement of a severe security vulnerability in the software they use. The coefficient on $newMnorVersionAvailable_{it}$ is substantially positive and significant, at 0.680. This implies that if there is a newer minor version than the version of Apache the organization is currently using, holding all else fixed, including the availability of newer major versions, it is associated with an $exp(0.680) - 1 = 97.4\%$ increase in the organization's probability of updating per month.

Column (4) of Table 4 further breaks down the effects of vulnerability disclosures and new releases on the propensity to update. Column (4) of Table 4 adds coefficients for if the version of Apache software an organization is using has a severe security vulnerability disclosed within the previous three months and if the updates available to an organization contain fixes for severe security bugs, non-severe security bugs, as well as if the update contains new or improved features unrelated to security.

The coefficient on $withinThreeMonthsSinceAddSevereBugs_{it}$ is large, positive, and significant at 0.214, meaning that an organization would increase its probability of updating by $exp\ (0.214) - 1 = 23.9\%$ if a severe vulnerability was disclosed less than three months ago. Moreover, the inclusion of this variable makes the coefficient on $hasSevereBugs_{it}$ close to zero and insignificant, suggesting that organizations' responses to severe vulnerabilities are concentrated to within three months since each disclosure.

The coefficient on $severeBugsFixed_{it}$, which captures the number of severe vulnerabilities the new minor version has fixed, is positive but small and not statistically significant. The coefficient on $nonSevereBugsFixed_{it}$, which captures the number of non-severe vulnerabilities the new minor version has fixed, is negative and significant. Each additional fix of non-severe bugs is associated with $1 - exp\ (-0.012) =$ 1.2% lower hazard rate of updating. The coefficient on $featureChanges_{it}$, which captures the number of feature changes the new minor version has implemented, is similarly negative and significant. Each additional feature change is associated with $1 - exp\ (-0.007)$ = 0.7% less updating. This implies that updates that contain many feature changes are also less likely to be installed immediately.

The results in Table 4 Column (4) demonstrate that organizations are more responsive to easily installed, incremental software updates than larger and more complex ones. New minor software versions are more likely to inspire an organization to adopt that update quickly. In contrast, major updates or updates composed of many non-essential feature changes are less likely to garner fast adoption.

Lastly, we examine if there is heterogeneity across organizations in their responses to vulnerability disclosures and releases of new versions. Specifically, Column (5) of Table 4, we interact the organizational attributes with vulnerability disclosures and the availability of new versions.

The estimated coefficient on the interaction term $newMinorVersionAvailable_{it} \times hasAlexaRank2010_i$ is 0.157 and significant. This implies that organizations have an $exp\ (0.157) - 1 = 17.0\%$ higher probability of updating per month when a new minor version is available if Alexa ranked the organization's website as among the top one million by traffic in 2010. High-traffic websites are frequently the target of cybercriminals, and, therefore, may see considerable benefits to installing updates to their server software.

Column (5) of Table 4 also shows that organizations using the cloud are more responsive to releasing new software versions. The coefficient on the $cloud_{it}$ variable is large, negative, and significant, meaning that if a site is hosted on the cloud, it is associated with a smaller probability of updating per month at the baseline. The interaction term $ewMinorVersionAvailable_{it} \times cloud_{it}$, however, is large, positive, and significant, meaning that when new minor releases come out, being hosted on the cloud is associated with a higher probability of updating. This suggests that organizations may face different incentives in updating on the cloud. For example, monitoring day-to-day may become costly, but new releases may prompt IT professionals to perform checkups and maintenance.

Taken together, these results demonstrate that costs and benefits are associated with the responsiveness of organizations to changes in their server software's up-to-datedness. Cloud providers may be able to lower the relative cost of responding to software releases. At the same time, high traffic to a website might encourage an organization to install new minor versions of server software when released.

Similar to high-traffic websites, public firms may also face increased visibility, making them particularly attractive targets for cyberattacks. In Column (6) of Table 4, we repeat our analysis regarding heterogeneity, focusing exclusively on publicly traded firms in our sample. In addition, because public firms disclose more information about their firm activities, we include additional control variables.

The estimated coefficient on $newMinorVersionAvailable_{it} \times monetizationTechs_i$ is positive and significant. This coefficient suggests that when new minor releases come out, an additional monetization technology embedded on a firm's website is associated with an $exp\ (0.123) - 1 =$ 13.1% increase in the hazard rate of updating. The coefficient on the interaction term $newMinorVersionAvailable_{it} \times techs_i$ suggests that that when new minor releases come out, an additional category of technology embedded on the firm's website is associated with a $1 - exp\ (-0.058) =$ 5.6% decrease in the hazard rate of installing updates.

Despite the much smaller sample size and the compromise on the precision of the estimates, the results in Column (6) highlight that public firms are more responsive to the new release of new software versions when they monetize their websites and thus have higher benefits to maintaining secure websites, and are less responsive when their websites are technically more complex.

For both the broader sample of organizations and the sub-sample of publicly traded firms, releases of software updates, rather than the announcement of security vulnerabilities, are the primary instigator of the pace of adopting updates. Similarly, the technical complexity of an organization's website and the content of the new release slows the pace at which an organization installs important software updates.

5.6. Robustness and Alternative Remediation Methods

Some severe security vulnerabilities associated with Apache only impact a subset of Apache deployments. For example, a bug may create a vulnerability for organizations running the Apache software on a Linux-based server, but not create a vulnerability for organizations running the Apache software on a Windows-based server. If vulnerabilities are bespoke to their context, then organizations may reasonably forgo installing updates that are unrelated to their server setup. During the time frame of our analysis, 5 out of the 28 severe security vulnerabilities in Apache only impacted deployments on specific operating systems. In Appendix Tables A3 and A4, we test if these vulnerabilities qualitatively change our results by dropping them from our regressions. The sequential logit regressions in Table A3 shows that the results are qualitatively and quantitatively similar when we drop event windows around the releases of fixes to operating system-specific vulnerabilities. When it comes to the time-to-update in the survival analysis, if organizations are more responsive to vulnerabilities specific to their operating systems, we will observe the coefficient on $hasSevereBugs_{it}$ to be biased downward when we drop operating system-specific vulnerabilities in the construction of the $hasSevereBugs_{it}$ variable. Appendix Table A4 shows that the coefficient on $hasSevereBugs_{it}$ turns negative, consistent with this intuition. The coefficients of other variables are similar to those in Table 4.

In addition, organizations may choose to ignore or delay installing updates related to severe security vulnerabilities that do not seem easily exploitable. In particular, if the potential costs of installing updates are high, organizations may wait to install updates that have a low probability of being exploited. Furthermore, they may use that delay time to learn from their counterparts about any technical issues or pitfalls from installing the updates. We test if security vulnerabilities with low potential exploitability have a sizeable impact on our results. In Appendix Tables A5 and A6, we re-estimate our main specifications, but drop 9 out of the 28 severe security vulnerabilities that are not considered to be highly exploitable. Again, overall, the coefficients in Appendix Tables A5 and A6 demonstrate the same basic results as our previous results. One difference, however, is that the coefficient on the organization using a cloud service provider is no longer negative and significant in the sequential logit regressions. This could be because cloud hosts are more attentive to both highly exploitable and low exploitable vulnerabilities, whereas firms that are not on a cloud provider focus more attention on highly exploitable vulnerabilities.

Finally, organizations may be slow to patch their web server software because they rely on other forms of remediations. For example, organizations may have intrusion detection systems (IDSs) installed that provide alerts when suspicious activity is found on the organizations' machines. If the IDS provider updates their database of vulnerabilities in a timely manner, organizations may at least be able to detect the exploitation of a vulnerability even if they have yet to install the software patch closing that vulnerability (Ransbotham, et al, 2012). Unfortunately, our data does not allow us to observe which organizations contract with an IDS and which do not.

6. Conclusion and Discussion

This study examined the largest assembled dataset tracking security vulnerabilities in open-source server software used by over 150,000 organizations in the United States between 2000 and 2018. Our goal was to understand a previously unexplored research question: how fast do software users respond to the availability of secure versions, and what determines the variance in the installation distribution? Previous research on cybersecurity has primarily focused on the role of the software vendors in providing updates for vulnerabilities and took for granted that users would automatically implement them, an approach that ignores the significant heterogeneity in whether and when software users respond to such updates.

Across our data, we found widespread usage of insecure server software by organizations. In nearly every month between 2000 and 2018, no less than 19% of the Apache servers in use contained a severe security vulnerability. This alarming finding means that the opportunities for malicious actors to exploit vulnerabilities are extremely widespread online.

We also found that organizational routines regarding updating server software are hard to predict using the observable characteristics of organizations in our data. Organizations can be broadly classified as *Frontier Chasers* (FC), *Do-Nothing* (DN) firms, and *Do-Something* (DS) firms based on how quickly they update to new versions of the Apache server software after its release. FCs are those that keep their software at the technological and security frontier by immediately installing updates and security patches as they become available. DNs choose a baseline rate of update that largely ignores available software updates. DSs are those that opt not to install updates immediately after their release yet do intend to install the updates eventually.

Our analysis revealed that an organization's industry, geography, and characteristics of the organization's website do not strongly predict the organization's routine regarding software updates. Instead, persistent unobservable aspects of organizations explain much of the variation in the updating routines. These unobservables may include organizational culture, complementary technologies, external vendor relationships, or other factors for which observational data cannot be easily obtained.

Lastly, we examined the organizational characteristics and attributes of Apache updates that predict how quickly an organization will install an update. Econometrically, analyzing organizational characteristics poses a challenge, as there is a reverse causality concern that makes vulnerabilities and slow responses to available software updates appear correlated: the longer that an organization uses a version of software, the more likely that version of software is to contain a severe vulnerability. Implementing a hazard model approach with stratification, thereby allowing for organization-specific proclivities to upgrade, we undercover that the factors most predictive of an organization installing a software quickly are those that cut down the cost of the update, such as the usage of cloud-hosting services such as AWS, and those that raise the benefits of a secure website, such as the presence of monetization technologies on the site. The factors that are more predictive of slower rates of installing updates are organizations whose websites are more technically complex and may need to be more careful about the potential of updates to break components of their site. As for the attributes of the Apache updates that may affect organizations' responses to available updates, we documented that software updates that are smaller and contain fewer major security updates are more likely to be adopted by organizations versus updates that contain new features or major security patches.

The findings in this paper highlight ways that organizations can increase cybersecurity. First, leaders of organizations can institute organizational routines that are attentive to cybersecurity and software management. Our data highlighted that most organizations do not routinely install software updates within the six months following their release. Indeed, we found relatively few firms that demonstrated the habit of installing software updates when they became available. And yet, cyberattacks are likely to pick up on hints from these update releases and target sites during the months following the release (Mitra and Ransbotham, 2015). Therefore, managers should consider if a routine that includes updating at regular intervals is beneficial for their organization. Second, organizations can take actions that reduce the cost of installing updates. One way that an organization can decrease the frictions involved in updates is by hosting their website on a cloud-based platform that assists with installing software updates. Another way that organizations can reduce friction is by decreasing the technological complexity of their website. Many websites use a complex web of technologies in the code of their website. The complexity, however, makes it challenging for a firm to install updates, as the update may break the components of the site. Managers who wish to mitigate such friction can consider pruning the technologies utilized for serving the site.

Our findings also have implications for policymakers. First, scholars and policymakers have debated whether or not to mandate the disclosure of software vulnerabilities (Arora, Telang, and Xu, 2008). Given how slow and unthorough organizations are installing updates, mandating disclosure about software vulnerabilities may be socially inefficient. Disclosures of software vulnerabilities could point malicious actors towards exploitable bugs without instigating users' faster adoption of software updates. Second, software vendors have been torn between two approaches: disclosing information about vulnerabilities in hopes that the description of the problem might instigate users to install patches or releasing software updates without details on vulnerabilities (Mitra and Ransbotham, 2015). Our findings suggest that the latter approach, in which little information about vulnerabilities is disclosed, warrants more consideration. Our analysis of the time until a firm installed a software update revealed that fixes for severe bugs in an update did not significantly speed up the installation of that update. Instead, much of the variation in time to installation derives from persistent organizational routines and unobservable organizational heterogeneity. This suggests that disclosing more information about vulnerabilities will give malicious actors more information to craft their attacks but may not jolt firms into hardening their defenses. Third, scholars have debated if software vendors should be liable for the cost of installing patches (August and Tunca, 2011). The lengthy delay between the release of software updates and the installation that we discovered in our data reveals that the costs associated with installing patches must be quite high for many firms. While it may be challenging to enact this policy for open-source software, our results suggest that subsidizing patching costs may be beneficial to increasing cybersecurity.

Our work does have limitations. First, we are only examining Apache web servers. While open-source server software operates on the majority of servers today (Greenstein and Nagle, 2014), the process of installing updates on proprietary server software, especially from software vendors that automatically send software updates to users, may be different. We were able to acquire data for our analysis because Apache server software is open source, the Apache Foundation is transparent about vulnerability reports, and the Apache Foundation documents and releases information on the contents of every software update. Proprietary software vendors are less transparent about their products and users. However, future research on the usage of proprietary software would enable a more complete picture of software user behavior more generally. Second, we have limited ability to match our data on server software decisions at organizations with other management practices at those organizations. The observed routines regarding software updates may reflect broader managerial routines or IT investments. While we have attempted to match our data with information in the World Management Survey, the overlap in samples provide limited statistical power for analysis, and the select sample of matching firms constrains the external validity of any findings. We hope that future researchers will find ways to expand on the work that we have begun by attempting to understand how managerial practices more broadly influence IT and cybersecurity investments.

References

Accenture. 2019. "Cost of Cybercrime Study." Ninth Annual. https://www.accenture.com/us-en/insights/security/cost-cybercrime-study.

Acronis International. 2017. "The NHS Cyber Attack: How and Why It Happened, and Who Did It." Case Study. Acronis International. https://www.acronis.com/en-us/articles/nhs-cyber-attack/.

Aral, Sinan, Erik Brynjolfsson, and D.J. Wu (2006), "Which came First, IT or Productivity? The Virtuous Cycle of Investment and use in Enterprise Systems," Twenty Seventh International Conference on Information Systems, 27, 22.

Arbaugh, W. A., W. L. Fithen and J. McHugh, "Windows of vulnerability: a case study analysis," in Computer, vol. 33, no. 12, pp. 52-59, Dec. 2000, doi: 10.1109/2.889093.

Arora, Ashish, Rahul Telang, and Hao Xu. 2008. “Optimal Policy for Software Vulnerability Disclosure.” Management Science 54 (4): 16.

Arora, Ashish, Ramayya Krishnan, Rahul Telang, Yubao Yang, 2010, "An Empirical Analysis of Software Vendor's Patch Release Behavior: Impact of Vulnerability Disclosure" Information Systems Research, 21, 1, 115-132.

August, Terrence, and Marius Florin Niculescu, Hyoduk Shin, 2014, "Cloud Implications of Software Network Structure and Security Risks," Information Systems Research, 25, 3, 489-510.

August, Terrence, and Tunay Tunca, 2006, "Network Software Security and User Incentives," Management Science, 52, 11, 1703-1720.

Barrett, M. (2018), Framework for Improving Critical Infrastructure Cybersecurity Version 1.1, NIST Cybersecurity Framework, [online], https://doi.org/10.6028/NIST.CSWP.04162018, https://www.nist.gov/cyberframework (Accessed July 17, 2023)

Bessen, Jim, and C. Righi (2019). "Shocking Technology: What Happens When Firms Make Large IT Investments?" SSRN Electronic Journal.

Brynjolfsson, Erik, and Lorin Hitt, 2003, “Computing Productivity: Firm-Level Evidence.” Review of Economics and Statistics, 85, 4, 793-808.

Cavusoglu, Hasan, Huseyin Cavusoglu, and Jun Zhang. 2008. “Security Patch Management: Share the Burden or Share the Damage?” Management Science 54 (4): 657–70.

Choi, J.P., Fershtman, C. and Gandal, N. (2010), “Network Security: Vulnerabilities And Disclosure Policy,” *The Journal of Industrial Economics*, 58: 868-894. https://doi.org/10.1111/j.1467-6451.2010.00435.x

Cohen, M. D., & Bacdayan, P. (1994). Oganizational routines are stored as procedural memory: Evidence from a laboratory. *Organization Science, 5*(4), 554–568. https://doi.org/10.1287/orsc.5.4.554

Comino, Steven, Fabio Manenti, and Franco Mariuzzo, 2018, "Updates Management in Mobile Applications: iTunes versus Google Play." Journal of Economics and Management Strategy, 28, 3, 392-419.

Cyert, R. M., & March, J. G. (1963). *A behavioral theory of the firm.* Prentice Hall/Pearson Education.

Dey, Debabrata, Atanu Lahiri, and Guoying Zhang, 2015, "Optimal Policies for Security Patch Management," Informs Journal on Computing, 27, 3, 462-477

Dissanayake, Nesara, Asangi Jayatilaka, Mansooreh Zahedi, M. Ali Babar, 2022, "Software Security Patch Management -- A Systematic Literature of Challenges, Approaches, Tools, and Practices," Information and Software Technology, 144,106771.

Dissanayake, Nesara, Mansooreh Zahedi, Asangi Jayatilaka, and Muhammad Ali Babar. 2022. Why, How and Where of Delays in Software Security Patch Management: An Empirical Investigation in the Healthcare Sector. Proc. ACM Hum.-Comput. Interact. 6, CSCW2, Article 362 (November 2022), 29 pages. https://doi.org/10.1145/3555087

Efron, Bradley. "Logistic Regression, Survival Analysis, and the Kaplan-Meier Curve." *Journal of the American Statistical Association* 83, no. 402 (1988): 414–25. https://doi.org/10.2307/2288857.

EY Americas, 2021. " Cybersecurity: How do you rise above the waves of a perfect storm?," Report, July 22, 2021, https://www.ey.com/en_us/cybersecurity/cybersecurity-how-do-you-rise-above-the-waves-of-a-perfect-storm.

Fine, JP. 2002. "Comparing Nonnested Cox Models." *Biometrika* 89 (3): 635–48.

Foster, L., John Haltiwanger, and C.J. Krizan. (2001), "Aggregate Productivity Growth: Lessons from the Microeconomic Evidence." In (eds) Hulten, Charles, Edwin Dean, and Michael Harper, New Developments in Productivity Analysis. NBER; Cambridge, MA.

Fowler, Bree, (2023), "Data Breaches Hit Lots More People in 2022," *CNET*, https://www.cnet.com/tech/services-and-software/data-breaches-hit-lots-more-people-in-2022/

Goodin, Dan, (2017), "Failure to patch two-month-old bug led to massive Equifax breach," *Arstechnica*, September 13, 2017, https://arstechnica.com/information-technology/2017/09/massive-equifax-breach-caused-by-failure-to-patch-two-month-old-bug/.

Goldenberg, Amit, and Julian Zlatev. (2022) "Atlanta Ransomware Attack (A)." Harvard Business School Case 923-009.

Grambsch, Patricia M, and Terry M Therneau. 1994. "Proportional Hazards Tests and Diagnostics Based on Weighted Residuals." *Biometrika* 81 (3): 515–26.

Greenstein, Shane, and Frank Nagle. "Digital Dark Matter and the Economic Contribution of Apache." Research Policy 43, no. 4 (May 2014): 623–631.

Harvey, Sarah. 2018. "Ransomware Alert: Lessons Learned from the City of Atlanta," KirkpatrickPrice Blog, April 3, 2018, https://kirkpatrickprice.com/blog/ransomware-alert-lessons-learned-city-atlanta/

Hui-Wen, Koo and I.P.L. Png, "Private security: Deterrent or diversion?" *International Review of Law and Economics*, vol. 14 (1), 1994, https://doi.org/10.1016/0144-8188(94)90038-8.

IBM. 2020. “Compromised Employee Accounts Led to Most Expensive Data Breaches Over Past Year.” Cambridge, MA: IBM. https://newsroom.ibm.com/2020-07-29-IBM-Report-Compromised-Employee-Accounts-Led-to-Most-Expensive-Data-Breaches-Over-Past-Year.

Jacobs, Jay, Sasha Romanosky, Idris Adjerid, and Wade Baker, 2020, “Improving vulnerability remediation through better exploit prediction,” *Journal of Cybersecurity*, 6(1), https://doi.org/10.1093/cybsec/tyaa015

Jorgenson, Dale. 2005, Productivity, Vol. 3 Information Technology, and the American Growth Resurgence. MIT Press.

Jorgenson, Dale W., Mun S. Ho, and Jon D. Samuels. 2016. “The Impact of Information Technology on Postwar U.S. Economic Growth” Telecommunications Policy. 40(5), 398-411.

Kang, Hye Young, 2022, "Too Much can be as bad as too little: Product Update Strategy for Online Digital Platform Complementors." Industrial and Corporate Change, 31, 6, 1494-1516.

Kleinbaum, David G, and Mitchel Klein. 1996. *Survival Analysis a Self-Learning Text*. Springer.

Leyden, Ben, 2022, "There's an App (Update) for That." Working Paper. Cornell. April.

Li, He, Sungjin Yoo, and William Kettinger, 2021, "The Role of IT Strategies and Security Investments in Reducing Organizational Security Breaches," Journal of Management Information Systems, 38, 222-245.

Liu, Che-Wei, Peng Huang, and Henry Lucas, 2017, "It Centralization, Security Outsourcing, and Cybersecurity Breaches: Evidence from US Higher Education," ICIS Proceedings. http://aisel.aisnet.org/icis2017/Security/Presentations/1.

McElheran, Kristina, 2015, "Do Market Leaders Lead in Business Process Innovation? The Cases(s) of E-Business Adoption," Management Science, 61, 6, 1197-1216.

Mell, Peter, Karen Scarfone, and Sasha Romanosky. 2007. “A Complete Guide to the Common Vulnerability Scoring System Version 2.0,” FIRST, [online], https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=51198, http://www.first.org/cvss/cvss-guide.pdf (Accessed May 30, 2023).

Mitra, Sabyasachi, and Sam Ransbotham. 2015. “Information Disclosure and the Diffusion of Information Security Attacks.” Information Systems Research 26 (3): 565–84. https://doi.org/10.1287/isre.2015.0587.

Mookerjee, Vijay, Radha Mookerjee, Alain Bensoussan, and Wei T. Yue. 2011. “When Hackers Talk: Managing Information Security Under Variable Attack Rates and Knowledge Dissemination.” Information Systems Research 22 (3): 606–23. https://doi.org/10.1287/isre.1100.0341.

Murciano-Goroff, Raviv, Ran Zhuo, Shane Greenstein, 2021. “Unsung Software and Veiled Value Creation: Illustrations from Server Software.” Research Policy. 50, Pp. 1-31.

Palmer, Danny. 2017. “WannaCry Ransomware: Hospitals Were Warned to Patch System to Protect against Cyber-Attack - but Didn’t.” ZDNet, October 27, 2017. https://www.zdnet.com/article/wannacry-ransomware-hospitals-were-warned-to-patch-system-to-protect-against-cyber-attack-but-didnt/.

Ranger, Steve. 2019. “Cybersecurity: One in Three Breaches Are Caused by Unpatched Vulnerabilities.” ZDNet, June 4, 2019. https://www.zdnet.com/article/cybersecurity-one-in-three-breaches-are-caused-by-unpatched-vulnerabilities/.

Ransbotham, Sam, Sabyaschi Mitra and Jon Ramsey, 2012. “Are Markets for Vulnerabilities Effective?,” *MIS Quarterly* 36(1) (March 2012).

Souppaya, Murugiah and Karen Scarfone, “Guide to Enterprise Patch Management Planning: Preventive Maintenance for Technology,” NIST, SP 800-40 Rev. 4 (April 2022).

Steinberg, Scott. 2019. “Cyberattacks Now Cost Companies $200,000 on Average, Putting Many out of Business.” CNBC. October 13, 2019. https://www.cnbc.com/2019/10/13/cyberattacks-cost-small-companies-200k-putting-many-out-of-business.html.

Tambe, Prassanne, and Lorin, Hitt, 2012. “The Productivity of Information Technology Investments: New Evidence from IT Labor Data.” Information Systems Research. 3. 599-617.

Tambe, Prassane, Lorin Hitt, Daniel Rock, and Erik Brynjolfsson. (2020), "Digital Capital and Superstar Firms," SSRN.

Zolas, N, Z. Kroff, E. Brynjolfsson, K. McElheran, D.N. Beede, C. Buffington, N. Goldschlag, L. Foster, and E. Dinlersoz (2020), "Advanced Technologies Adoption and use by U. S. Firms: Evidence from the Annual Business Survey." NBER Working paper 28290.

Tables and Figures

Figure 1. A simple model of user decisions

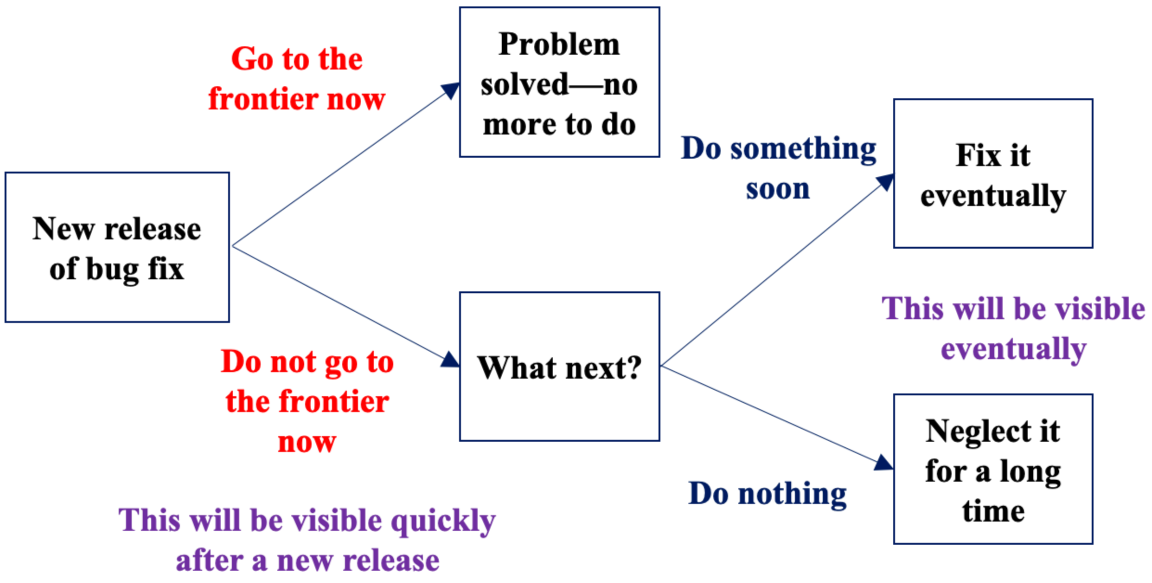
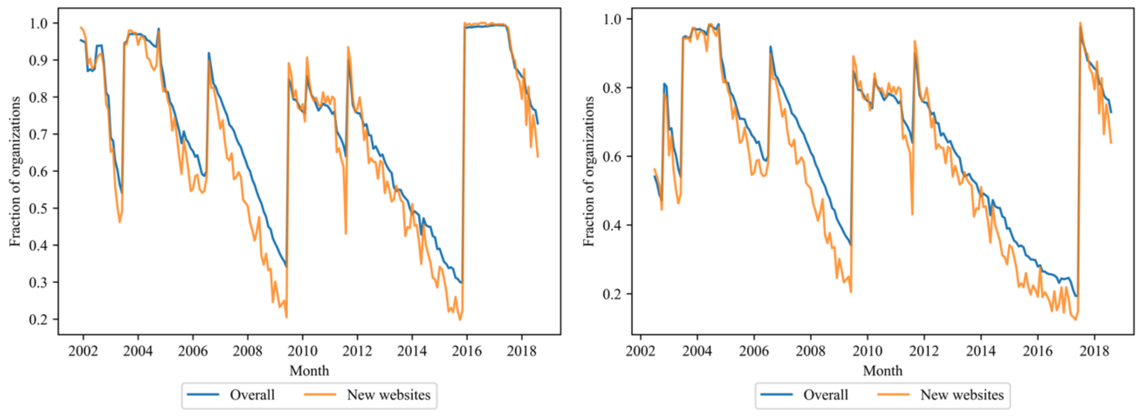

Go to the frontier now
Problem solved—no more to do
New release of bug fix
Do something soon
Fix it eventually
This will be visible eventually
What next?
Do not go to the frontier now
Neglect it for a long time
Do nothing
This will be visible quickly after a new release

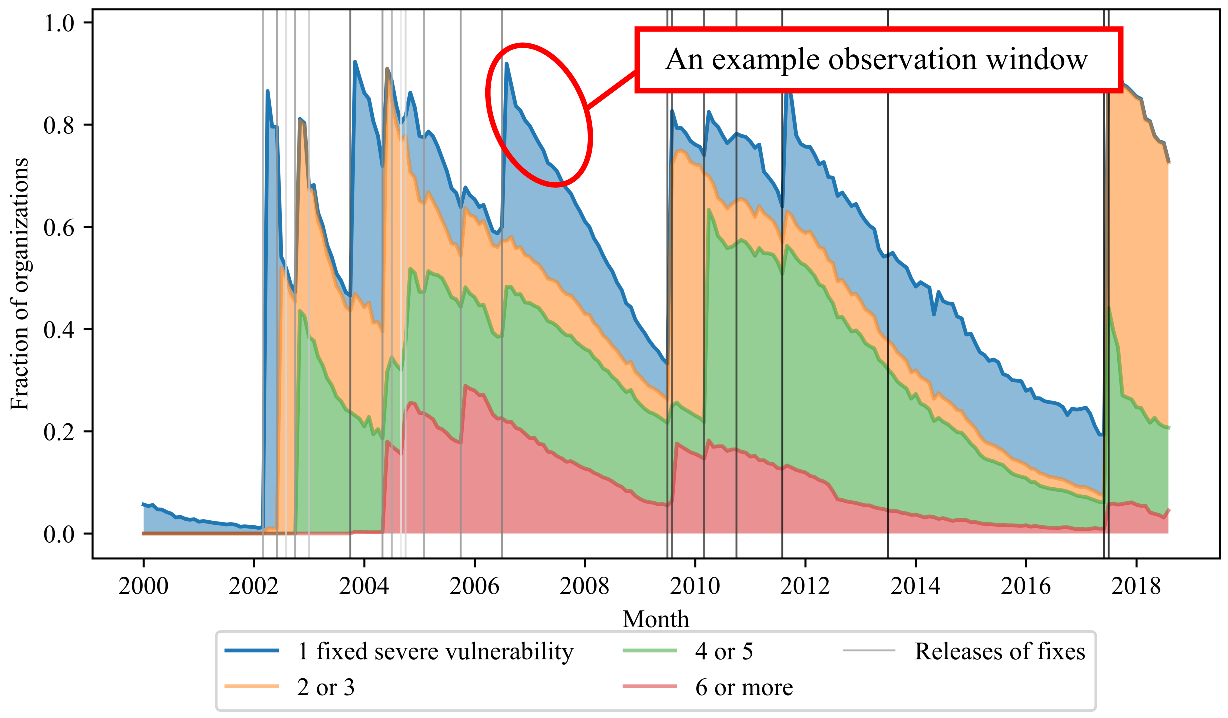
Figure 2 Organizations Operating with Severe Security Vulnerabilities



(a) Reported (b) Disclosed

Notes: A new website is the first capture of a website by the Internet Archive between Jan 1, 2000, and August 31, 2018. Only vulnerabilities rated to have “high” severity by the National Vulnerability Database (NVD) are included in computing the fractions. Figure 2(a) plots the fraction of organizations in our broad panel between December 2001 and August 2018. The determination of when a severe security vulnerability was discovered is by the time the vulnerability was reported to the Apache Software Foundation. The first severe vulnerability report in our data is from November 2001. Figure 2(b) plots the fraction of organizations in our broad panel between July 2002 and August 2018. The determination of when a severe security vulnerability became public is by the Apache Software Foundation. The first severe vulnerability that became public in our data was in June 2002.

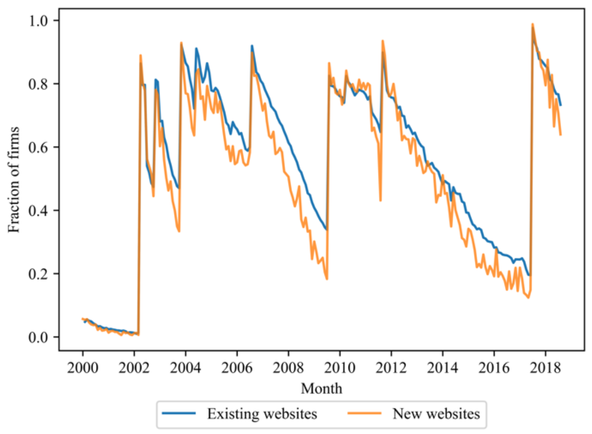
Figure 3 Organizations Operating with Fixed Severe Security Vulnerabilities



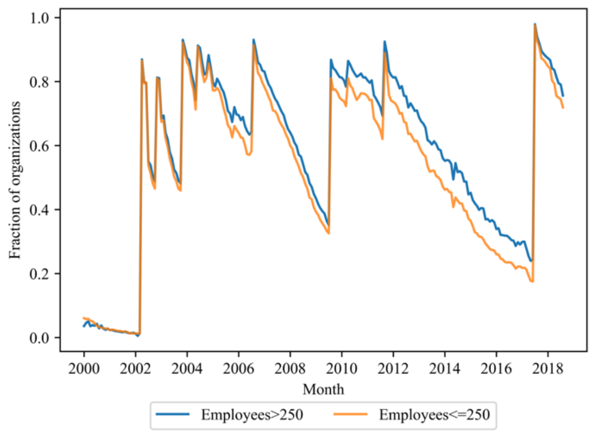
Notes: Figure 3 plots the fraction of organizations in our broad panel between Jan 2000 and August 2018. The determination of when a severe security vulnerability was fixed is by the Apache Software Foundation. The first severe vulnerability fix release date in our data is September 1998.

Figure 4 Heterogeneity in Operating with Fixed Severe Security Vulnerabilities

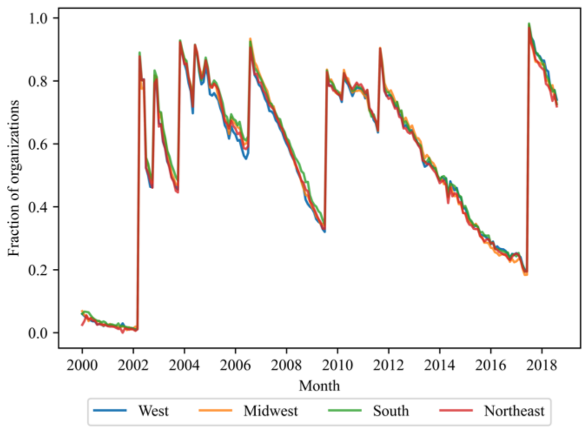
(a) Existing websites vs. new websites



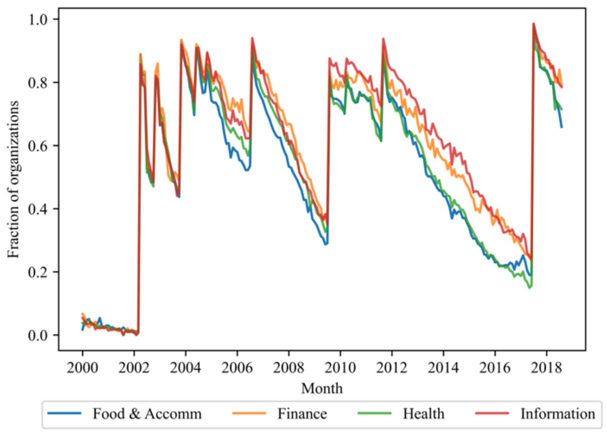
(b) Large vs. small organizations



(c) Geography



(d) Industry



Notes: Panels (b), (c), and (d) include observations from organizations that have their information in the Mint Global database. The number of such organizations is 145,515 (96% of the broad panel). The classification of industries is by two-digit NAICS code. The rest of the notes of Figure 3 apply.

Table 1 Summary of Samples and Data Sources

	Data	Description	Frequency	Source
Key samples and data sources				
(i)(a)	Broad panel of organizations' Apache web server usage	The version of the Apache web server of 150,854 homepages of US organizations	Monthly between Jan 1, 2000, and August 31, 2018; 4,863,383 observations in total; on average, each organization has 32 observations.[27]	The Internet Archive
(i)(b)	Restricted panel of organizations' Apache web server usage(frequent captures, uncensored)	The version of the Apache web server of 70,092 homepages of US organizations that were frequently captured by the Internet Archive;[28] each observation is not from a left-censored updating cycle and is not the	Monthly between Jan 1, 2000, and August 31, 2018; 3,104,455 observations in total; on average, each organization has 44 observations.[30]	The Internet Archive

[27] If an organization's website only became active after Jan 1, 2000, or went offline before Dec 31, 2018, the months before the website went online and the months after the website went offline do not have observations. For some organizations, the Internet Archive did not capture the active websites every month. Therefore, those months do not have observations. In addition, some organizations have used other web servers (such as Microsoft's IIS or Nginx) during the period. Those observations are not included.

[28] We only include organizations that were, on average, captured by the Internet Archive every three months or more frequently. Moreover, we only have organizations with at least ten monthly observations of Apache web server usage. Doing so allows us to observe organizations' upgrading decisions over a duration equal to at least two average release cycles of the Apache web server software, as the average release cycle of an Apache web server is 4.5 months. We then drop an observation if the time gap between the observation and the following observation of the organization is more than 4.5 months apart.

[30] Footnote 22 applies.

		last observation of Apache usage of the organization in the raw data[29]		
(ii)	Apache web server security vulnerabilities	Security vulnerability reported to the Apache Software Foundation; variables include the version(s) affected, date of reporting, the date the vulnerability was made public, date of release and version of the fix, and (when available) who was credited with the discovery/reporting of the vulnerability	158 unique vulnerabilities; 5,192 vulnerability-version pairs; for vulnerabilities reported before or on August 31, 2018	The Apache Foundation
(iii)	Severity of security vulnerabilities	Severity rating of "high," "medium," or "low" for each security vulnerability; breakdown scores for the impact and exploitability of each vulnerability are also available	28 vulnerabilities were rated "high," 123 were rated "medium," and 7 were rated "low."	National Vulnerability Database (NVD)
(iv)	Apache web server version release dates	The release date of each minor version (e.g., 2.4.1) of the Apache web server software	115 releases of minor versions between 2000 and 2018; those minor versions belong to six different major versions (e.g., 2.4)	Authors' compilation
(v)	Apache version change logs	The number of changes implemented in each release of a minor version; combined with (ii), we can identify the number of changes corresponding to fixing severe vulnerabilities, fixing non-severe bugs, and improving the features		The Apache Foundation
(vi)	Release window cross section of organizations'	Organizations' responses to releases of new Apache versions that fixed severe security vulnerabilities in their used versions;	Cross-sectional; each observation corresponds to one organization during one six-month release	Authors' construction

[29] Our survival models account for time-dependent covariates that vary monthly. Left-censoring due to the first data capture is problematic for all observations associated with the left-censored updating cycles in determining the time-to-update. Right-censoring due to the last data capture is only difficult for the final observation of the organization's Apache usage in determining whether updating has happened during the final observed month. We, therefore, drop the problematic observations.

	responses to bug fixes	responses are measured in the six-month window following each release, and fall into three categories: frontier chaser (installing the release within two months), do-something (installing within six months), and do-nothing (not installing within six months)	window; 161,106 observations in total from 55,558 organizations and 15 six-month release windows; we only include releases that fixed severe security vulnerabilities and that were at least six months apart from the next such releases	based on data items (1)(b)-(v)
Additional covariates				
(vii)	Alexa web traffic ranking	Ranking of top one million websites by traffic	Yearly between 2010 and 2018	Alexa
(viii)	Organizations' IT operations	Characteristics of US organizations' IT operations; variables include the number of personal computers, the number of IT staff, IT budget, and software budget	A cross-sectional snapshot of the database in 2017	Harte Hanks
(ix)	Organizations' IT sourcing	Whether the organization has outsourced its IT operations	The database contains the IT sourcing variable yearly between 2005 and 2009 for a small group of organizations; we consider an organization to have outsourced its IT operations if it has outsourced in one or more years during 2005—2009	Harte Hanks
(x)	Websites' cloud usage	Whether an organization's web server has used an IP address associated with AWS, Azure, or Google Cloud	For AWS, Azure, and Google usage, we use snapshots of all IP addresses associated with those services taken on March 25, 2020, August 13, 2023, and August 14, 2023 respectively.	AWS, Microsoft, and Google
(xi)	Websites' technology use	Technologies used in building 24,263 organizations' homepages; 45 technology categories, including analytics, e-	The data was captured between 2016 and 2018; we consider the website to have used a particular technology if	HTTP Archive

		commerce, and web frameworks; 531 technologies, including jQuery, Google Tag Manager, and WordPress	we observe the usage of the technology anytime during 2016—2018	
(xii)	Disclosures of data breaches	2,366 data breaches disclosed between 2005 and 2018; variables include the disclosing organization, the date of disclosure, the state and broad industry category the organization operates in, and the number of records affected		Privacy Rights Clearinghouse's Data Breach Chronology
(xiii)	Organization characteristics	214,199 organizations in the US with at least 50 employees; organizations are identified by their homepages; variables include location, NAICS code, number of employees, and revenue	A cross-sectional snapshot of the database on August 28, 2018	Mint Global by Bureau van Dijk
(xiv)	Organization characteristics (public firms)	US public firms' characteristics; variables include total assets, capital expenditure, depreciation and amortization, employees, cash flow, and net income	The yearly panel between 2000 and 2018	Compustat

Table 2 Summary statistics

Variable	Obs.	Mean	SD	Min	Max
$updated_{it}$	3,104,455	0.062	0.240	0	1
$timeToUpdateMonthEnd_{it}$	3,104,455	17.926	18.543	1	202
$infrequentUpdater_{i}$	2,701,344	0.338	0.473	0	1
$frequentUpdater_{i}$	2,701,344	0.287	0.453	0	1
$hasSevereBugs_{it}$	3,104,455	0.632	0.482	0	1
$withinThreeMonthsSinceAddSevereBug_{it}$	3,104,455	0.189	0.391	0	1
$newVersionAvailable_{it}$	3,104,455	0.952	0.215	0	1
$newMinorVersionAvailable_{it}$	3,104,455	0.837	0.370	0	1
$severeBugsFixed_{it}$	3,049,753	0.348	0.707	0	4
$nonsevereBugsFixed_{it}$	3,049,753	1.409	1.579	0	9
$featureChanges_{it}$	3,049,753	20.331	21.336	0	91
$hasAlexaRank2010_{i}$	3,104,455	0.219	0.414	0	1
$logPCs_{i}$	2,522,559	4.206	1.471	0	11.458
$ITStaffCategory_{i}$	2,522,337	1.457	1.417	0	9
$logITBudget_{i}$	2,522,559	12.816	2.539	0	23.575
$logSoftwareBudget_{i}$	2,522,559	11.103	2.421	0	22.222
$outsourced_{i}$	711,441	0.171	0.377	0	1
$cloud_{it}$	3,104,455	0.022	0.147	0	1
$techCategories_{i}$	665,360	10.047	3.849	1	22
$techs_{i}$	665,360	13.520	6.104	1	38
$monetizationTechs_{i}$	665,360	2.842	2.240	0	15
$countBreachState_{it}$	3,104,455	0.490	1.343	0	18
$logNoAffectedBreachState_{it}$	3,104,455	1.630	3.916	0	21.822

$countBreachInd_{it}$	3,104,455	2.171	3.878	0	31
$logNoAffectedBreachInd_{it}$	3,104,455	4.178	5.885	0	21.822
$state_{i}$	3,019,376				
$naics_{i}$	2,958,680				
$logEmployment_{i}$	3,026,268	5.339	1.196	3.932	14.715
$logRevenue_{i}$	3,026,268	9.213	2.802	0	20.031
$isPublic_{iy}$	3,104,455	0.027	0.161	0	1
$logCapxCompustat_{iy}$	82,935	8.635	3.416	-5.690	17.015
$logEmploymentCompustat_{iy}$	82,935	6.482	2.859	0	14.604
$logTotalAssetsCompustat_{iy}$	82,935	19.469	3.886	0	28.424
$logDepreciationCompustat_{iy}$	82,935	15.332	4.556	0	24.029
$logIncomeCompustat_{iy}$	82,935	5.346	16.082	-	24.455
$logCashflowCompustat_{iy}$	80,701	0.008	0.215	-2.534	1.825

Notes: Each observation represents one month in an updating cycle of an organization that used Apache server software. The number of organizations in this data is 70,092. The event is making an update of the version of Apache server software an organization used. The number of events is 191,106. The number of updating cycles is 308,689. 308,689 – 191,106= 117,583 updating cycles are right-censored.

Table 3 Sequential logit regressions of cross-sectional determinants of user decisions

	Sequential logit regression		Sequential logit regression	
	(1)	(2)	(3)	(4)
	DN/DS vs FC	DN vs DS	DN/DS vs FC	DN vs DS
$infreqentUpdater_i$	0.167***	0.192***	0.169**	0.0490
	(0.0402)	(0.0420)	(0.0795)	(0.0694)
$frequentUpdater_i$	-0.239***	-0.0987***	-0.195**	-0.226***
	(0.0445)	(0.0353)	(0.0908)	(0.0719)
$logEmployment_i$	0.0379**	0.0122	0.0254	0.00790
	(0.0163)	(0.0179)	(0.0326)	(0.0282)
$logRevenue_i$	-0.00138	0.000733	-0.00897	0.00232
	(0.00877)	(0.00785)	(0.0132)	(0.0132)
$isPublic_{iy}$	-0.0646	0.0616	-0.0373	-0.00897
	(0.0786)	(0.0801)	(0.121)	(0.116)
$logPCs_i$	0.0445	0.0171	0.0166	0.0458
	(0.0298)	(0.0316)	(0.0573)	(0.0654)
$ITStaffCategory_i$	-0.0120	-0.00805	0.00644	-0.0165
	(0.0253)	(0.0287)	(0.0372)	(0.0403)
$logITBudget_i$	-0.0377	-0.0595	0.0398	-0.00401
	(0.0450)	(0.0400)	(0.0910)	(0.0839)
$logSoftwareBudget_i$	0.0399	0.0679	-0.0524	-0.0105
	(0.0446)	(0.0468)	(0.0823)	(0.0856)
$hasAlexaRank2010_i$	0.153***	-0.0155	0.0260	0.0168
	(0.0509)	(0.0480)	(0.0748)	(0.0795)
$cloud_{it}$	-0.138	-0.577***	-0.128	-0.505***
	(0.119)	(0.103)	(0.198)	(0.174)
$countBreachState_{it}$	-0.0492	0.0208	-0.100	0.0332
	(0.0482)	(0.0495)	(0.0665)	(0.0621)
$logNoAffectedBreachState_{it}$	0.0488	0.0249	0.0667**	0.0263
	(0.0312)	(0.0239)	(0.0319)	(0.0283)
$countBreachInd_{it}$	-0.00496	0.0560***	0.00569	0.0232
	(0.0181)	(0.0214)	(0.0320)	(0.0378)
$logNoAffectedBreachInd_{it}$	-0.0131	0.0207***	0.0202	0.0396***
	(0.0114)	(0.00691)	(0.0129)	(0.0127)
$outsourced_i = 1$ &			0.118	0.145
$outsourcedMissing_i = 0$			(0.166)	(0.169)
$outsourced_i = 0$ &			0.0658	-0.00625
$outsourcedMissing_i = 0$			(0.0658)	(0.0678)

$techCategories_i$			0.0144	0.0245
			(0.0206)	(0.0186)
$techs_i$			-0.00826	-0.0152
			(0.0131)	(0.0146)
$monetizationTechs_i$			-0.00348	-0.0118
			(0.0265)	(0.0279)
Constant	1.681***	1.281***	2.103***	0.809**
	(0.126)	(0.125)	(0.322)	(0.326)
State dummies	Y	Y	Y	Y
Industry dummies	Y	Y	Y	Y
Observations	108,385	108,385	26,540	26,540

Notes: *** p<0.01, ** p<0.05, * p<0.1. Standard errors are clustered by state and industry. Columns (1) and (2) belong to the same regression. Columns (3) and (4) belong to the same regression. The value at the beginning of the observation window is used for each variable with time subscript t for month or y for year. For the variable $outsourced_i$, the dropped category is $outsourcedMissing_i = 1$, where the variable is missing.

Table 4 Cox proportional hazards analysis of updating decisions

	Cox proportional hazards regressions					
						Public firms
			Stratified	Stratified	Stratified	Stratified
	(1)	(2)	(3)	(4)	(5)	(6)
$hasSevereBugs_{it}$	-0.174***	-0.263***	0.067***	0.003	0.160***	0.381
	(0.005)	(0.015)	(0.007)	(0.009)	(0.051)	(0.246)
$withinThreeMonthsSinceAddSevereBug_{it}$				0.214***		
				(0.009)		
$hasSevereBugs_{it} \times hasAlexaRank2010_i$					-0.054	0.099
					(0.037)	(0.179)
$hasSevereBugs_{it} \times cloud_{it}$					0.022	0.457
					(0.086)	(0.359)
$hasSevereBugs_{it} \times techCategories_i$					-0.006	-0.022
					(0.005)	(0.024)
$hasSevereBugs_{it} \times monetizationTechs_i$					-0.009	-0.062
					(0.008)	(0.043)
$newMinorVersionAvailable_{it}$	0.266***	0.322***	0.680***	0.796***	0.604***	0.519
	(0.008)	(0.025)	(0.009)	(0.011)	(0.069)	(0.323)
$severeBugsFixed_{it}$				0.008		
				(0.005)		
$nonsevereBugsFixed_{it}$				-0.012***		
				(0.002)		
$featureChanges_{it}$				-0.007***		
				(0.0002)		
$newMinorVersionAvailable_{it} \times hasAlexaRank2010_i$					0.157***	0.726***
					(0.049)	(0.230)
$newMinorVersionAvailable_{it} \times cloud_{it}$					0.292**	0.221
					(0.136)	(0.453)
$newMinorVersionAvailable_{it} \times techCategories_i$					-0.008	-0.058*
					(0.006)	(0.034)
$newMinorVersionAvailable_{it} \times monetizationTechs_i$					0.022*	0.123**
					(0.012)	(0.064)

$newVersionAvailable_{it}$	0.076***	-0.063	0.370***	0.422***	0.273***	0.187
	(0.013)	(0.042)	(0.014)	(0.015)	(0.039)	(0.198)
$infrequentUpdater_i$		-0.120***				
		(0.015)				
$frequentUpdater_i$		0.185***				
		(0.015)				
$logEmployment_i$		-0.008				
		(0.006)				
$logRevenue_i$		0.003				
		(0.003)				
$isPublic_i$		-0.058				
		(0.028)				
$hasAlexaRank2010_i$		0.053**				
		(0.014)				
$cloud_{it}$		0.009			-0.460***	-0.351
		(0.034)			(0.132)	(0.459)
$techCategories_i$		-0.013**				
		(0.004)				
$techs_i$		0.004				
		(0.003)				
$monetizationTechs_i$		0.012*				
		(0.004)				
$logPCs_i$		-0.017				
		(0.010)				
$ITStaffCategory_i$		0.015				
		(0.008)				
$logITBudget_i$		-0.011				
		(0.019)				
$logSoftwareBudget_i$		0.010				
		(0.019)				
$countBreachState_{it}$		-0.038***				
		(0.008)				
$logNoAffectedBreachState_{it}$		-0.011***				
		(0.002)				
$countBreachInd_{it}$		-0.027***				
		(0.003)				
$logNoAffectedBreachInd_{it}$		-0.013***				
		(0.001)				
$logCapxCompustat_{it}$						-0.041
						(0.029)

$logEmploymentCompustat_{it}$						0.037
						(0.027)
$logTotalAssetsCompustat_{it}$						-0.252**
						(0.078)
$logDepreciationCompustat_{it}$						-0.026
						(0.040)
$logIncomeCompustat_{it}$						-0.000
						(0.003)
$logCashflowCompustat_{it}$						0.159
						(0.313)
State dummies	N	Y	N	N	N	N
Industry dummies	N	Y	N	N	N	N
Observations	3,104,455	548,959	3,104,455	3,049,753	665,360	44,955

Notes: $^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$. Each observation represents one month in an updating cycle of an organization that used Apache server software. The event is the decision to update the version of Apache server software. Standard errors are clustered at the organization level.

Appendix A1. Variable Definitions

Table A1 Variable definitions

Variable	Definition
$updated_{it}$	Binary if organization i makes an update to the Apache server software in month t; an update happens in month t either when the organization uses a higher major version in the next observed month (e.g., the organization uses Apache 2.2 then changes to Apache 2.4) or when the organization uses a higher minor version in the next observed month (e.g., the organization uses Apache 2.2.10 then changes to 2.2.11)
$timeToUpdateMonthEnd_{it}$	The number of months since organization i's previous update of Apache server software
$infrequentUpdater_i$	Binary if organization i did not update server software even once in the first nine observations of Apache usage (Apache on average releases a new minor version once every 4.5 months for a major version); the variable is undefined for the first nine observations of Apache usage of each organization because those observations are used to construct the variable.
$frequentUpdater_i$	Binary if of the organization i updated server software two or more times in the first nine observations of Apache usage; the variable is undefined for the first nine observations of Apache usage of each organization because those observations are used to construct the variable.
$hasSevereBugs_{it}$	Binary if the minor version of Apache software (e.g., Apache 2.2.10) organization i used in the month t that publicly disclosed severe security vulnerabilities.
$withinThreeMonthsSinceAddSevereBug_{it}$	Binary if the minor version of Apache software organization i used in month t had additional severe security vulnerabilities disclosed within three months before month t
$newVersionAvailable_{it}$	Binary if a newer Apache version than organization i's adopted version in month t was available in that

	month; for example, the variable is equal to 1 if organization i used in month t version 2.2.x released in month $t' < t$, and 2.4.y was released after t' and was available at t
$newMinorVersionAvailable_{it}$	Binary if a newer Apache minor version than organization i's adopted version in month t was available in that month; for example, the variable is equal to 1 if organization i used in month t version 2.2.10 released in month $t' < t$, and 2.2.11 was released after t' and was available at t
$severeBugsFixed_{it}$	The number of severe security vulnerabilities the *oldest* new minor version has fixed; for example, if organization i used in month t version 2.2.10 released in month $t' < t$, and 2.2.11, 2.2.12, and 2.2.13 were released after t' and were available at t, this variable captures the number of severe security vulnerabilities fixed in 2.2.11; if there are not new minor version available in month t, this variable is set to zero
$nonsevereBugsFixed_{it}$	The number of non-severe bugs the *oldest* new minor version has fixed; for example, if organization i used in month t version 2.2.10 released in month $t' < t$, and 2.2.11, 2.2.12, and 2.2.13 were released after t' and were available at t, this variable captures the number of nonsevere bugs fixed in 2.2.11; if there are not new minor version available in month t, this variable is set to zero
$featureChanges_{it}$	The number of feature changes the *oldest* new minor version has implemented; for example, if organization i used in month t version 2.2.10 released in month $t' < t$, and 2.2.11, 2.2.12, and 2.2.13 were released after t' and were available at t, this variable captures the number of feature changes in 2.2.11; if there are not new minor version available in month t,, this variable is set to zero.
$hasAlexaRank2010_i$	Binary, =1 if the organization i's website was ranked by Alexa in the top 1 million websites in 2010
$logPCs_i$	Logged value of (number of personal computers+1) at organization i in 2017 according to Harte Hanks; if organization i has multiple establishments, we only use the number of personal computers at the

	establishment with the highest number of IT staff and IT budget.
$ITStaffCategory_i$	The number of IT staff at organization i in 2017 according to Harte Hanks; if organization i has multiple establishments, we only use the value at the establishment with the highest number of IT staff and IT budget; this variable is categorical, it $= 0$ if IT staff is equal to 0, $= 1$ if IT staff is between 1 to 4, $= 2$ if IT staff is between 5 to 9, $= 3$ if IT staff is between 10 to 24, $= 4$ if IT staff is between 25 to 49, $= 5$ if IT staff is between 50 to 99, $= 6$ if IT staff is between 100 to 249, $= 7$ if IT staff is between 250 to 499, $= 8$ if IT staff is between 500 to 999, $= 9$ if IT staff is 1000 and above
$logITBudget_i$	Logged value of (IT budget$+1$) at organization i in 2017 according to Harte Hanks; if organization i has multiple establishments, we only use the IT budget at the establishment with the highest number of IT staff and IT budget.
$logSoftwareBudget_i$	Logged value of (software budget$+1$) at organization i in 2017, according to Harte Hanks; if organization i has multiple establishments, we only use the software budget at the establishment with the highest number of IT staff and IT budget.
$outsourced_i$	Binary if organization i has outsourced its IT at least once between 2005 and 2009, from Harte Hanks.
$cloud_{it}$	Binary if organization i's website in month t used an IP address that belongs to AWS on March 25, 2020, Azure on August 13, 2023, or Google Cloud on August 14, 2023; data on historical IP addresses of cloud providers are not readily available.
$techCategories_i$	The number of technology categories embedded in organization i's website between 2016 and 2018; examples of technology categories include analytics, e-commerce, and web frameworks.
$techs_i$	The number of technologies embedded in organization i's the website between 2016 and 2018; examples include jQuery, Google Tag Manager, and WordPress.

$monetizationTechs_i$	The number of monetization technologies embedded in organization i's. website between 2016 and 2018; technologies in "analytics," "tag managers," "advertising networks," "marketing automation," "e-commerce," and "payment processors" categories are included; measures monetization intent.
$countBreachState_{it}$	The number of data breaches that have been disclosed in the month before month t in the state where organization i's headquarters are located.
$logNoAffectedBreachState_{it}$	Logged value of (the sum of the number of records affected$+1$) for data breaches that have been disclosed in the month before month t in the state where organization i's headquarters are located.
$countBreachInd_{it}$	The number of data breaches that have been disclosed in the month before month t in the broad industry category that other organization i operates in; broad industry categories in the Data Breach Chronology data are Healthcare and Medical Providers (Hospitals, Medical Insurance Services), Businesses (Retail/Merchant including Grocery Stores, Online Retailers, Restaurants), Businesses (Financial Services, Banking, Insurance Services), Businesses (Manufacturing, Technology, Communications, Other), Educational Institutions (Schools, Colleges, Universities)
$logNoAffectedBreachInd_{it}$	Logged value of (the sum of the number of records affected plus1) for data breaches that have been disclosed in the month before month t in the broad industry category that other organizations operate in; broad industry categories in the Data Breach Chronology data are Healthcare and Medical Providers (Hospitals, Medical Insurance Services), Businesses (Retail/Merchant including Grocery Stores, Online Retailers, Restaurants), Businesses (Financial Services, Banking, Insurance Services), Businesses (Manufacturing, Technology, Communications, Other), Educational Institutions (Schools, Colleges, Universities)
$state_i$	State, where the organization's headquarter is located
$naics_i$	The organization i's a two-digit primary NAICS code.

$logEmployment_i$	Logged value of (employment+1) of organization i; from the cross-sectional Mint Global sample of organizations, a single value is associated with an organization.
$logRevenue_i$	Logged value of (revenue+1) of organization i; if revenue is negative, this variable is equal to $-1 \times$ log $(abs(revenue) + 1)$; from Mint Global.
$isPublic_{iy}$	Binary if organization i is a public firm in year y.
$logCapxCompustat_{iy}$	Logged value of (capital expenditure+1) of organization i in year y; from Compustat, this variable is only available for public firms.
$logEmploymentCompustat_{iy}$	Logged value of (employment+1) of organization i in year y; from Compustat, this variable is only available for public firms.
$logTotalAssetsCompustat_{iy}$	Logged value of (total assets+1) of organization i in year y; from Compustat, this variable is only available for public firms.
$logDepreciationCompustat_{iy}$	Logged value of (depreciation and amortization+1) of organization i in year y; from Compustat, this variable is only available for public firms.
$logIncomeCompustat_{iy}$	Logged value of (income before extraordinary items+1) of organization i in year y; if income is negative, this variable is equal to $-1 \times$ log $(abs(income) + 1)$; from Compustat, this variable is only available for public firms.
$logCashflowCompustat_{iy}$	Logged value of (cashflow+1) of organization i in year y; if cashflow is negative, this variable is equal to $-1 \times$ log $(abs(cashflow) + 1)$; from Compustat, this variable is only available for public firms.

Table A2. Correlation Matrix of Measures

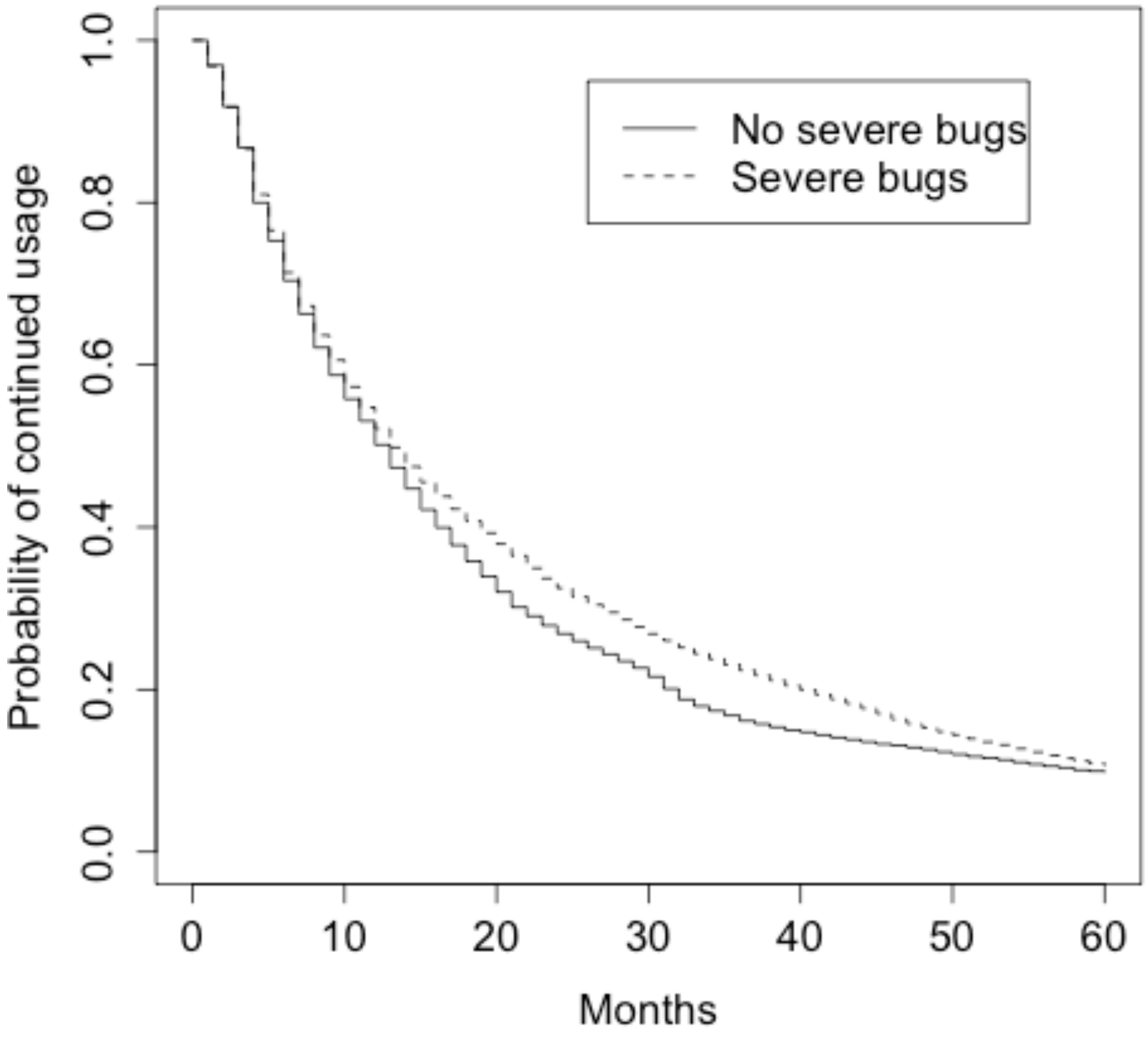
Variables	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)
(1) $hasSevereBugs_{it}$	1.00																	
(2) $withinThreeMonthsSinceAddSevereBug_{it}$	0.30	1.00																
(3) $newVersionAvailable_{it}$	0.24	0.06	1.00															
(4) $newMinorVersionAvailable_{it}$	0.43	0.10	0.53	1.00														
(5) $hasAlexaRank2010_{i}$	0.04	0.00	0.02	0.03	1.00													
(6) $logPCs_{i}$	0.05	0.00	0.02	0.04	0.14	1.00												
(7) $logITBudget_{i}$	0.03	0.01	0.01	0.03	0.08	0.73	1.00											
(8) $outsourced_{i}$	-0.02	0.00	0.00	-0.01	-0.05	-0.12	-0.10	1.00										
(9) $cloud_{it}$	-0.12	-0.04	-0.00	-0.01	-0.00	-0.01	0.01	0.00	1.00									
(10) $techCategories_{i}$	-0.02	-0.02	0.01	0.02	0.17	0.01	-0.02	0.01	0.06	1.00								
(11) $monetizationTechs_{i}$	0.01	0.01	0.00	0.00	0.24	0.05	0.05	-0.00	0.04	0.45	1.00							
(12) $countBreachState_{it}$	-0.06	-0.08	0.03	0.04	0.01	0.00	0.01	-0.00	0.14	0.02	-0.02	1.00						
(13) $logNoAffectedBreachState_{it}$	-0.03	-0.08	0.03	0.03	0.01	-0.00	0.01	-0.00	0.11	0.02	-0.02	0.68	1.00					
(14) $countBreachInd_{it}$	-0.08	-0.14	0.05	0.06	-0.04	-0.03	-0.01	0.00	0.16	0.05	-0.02	0.29	0.22	1.00				
(15) $logNoAffectedBreachInd_{it}$	-0.02	-0.15	0.06	0.07	-0.03	0.02	0.01	-0.01	0.10	0.04	-0.03	0.19	0.21	0.65	1.00			
(16) $logEmployment_{i}$	0.04	0.01	0.01	0.02	0.12	0.48	0.51	-0.09	0.00	-0.06	-0.01	-0.00	-0.00	-0.02	-0.00	1.00		
(17) $logRevenue_{i}$	0.02	0.00	0.00	0.01	0.11	0.27	0.35	-0.05	0.02	0.03	0.03	0.02	0.03	-0.00	-0.01	0.58	1.00	
(18) $isPublic_{iy}$	0.01	0.01	0.01	0.01	0.01	0.09	0.28	0.00	0.05	0.00	0.03	0.03	0.02	0.03	0.01	0.32	0.29	1.00

Appendix A2. Kaplan-Meier Curve for Continued Usage of Software Versions

Figure A1 visualizes the relationship between the time-to-update and the presence of severe security vulnerabilities with Kaplan-Meier survival curves. The horizontal axis shows the number of months an organization has been using a particular server software version. The vertical axis shows the probability that the organization is still using that software version at each time interval. We split the observations into those impacted by publicly disclosed severe security vulnerabilities (dashed line) and those not (solid line). For example, this figure shows that 27% of observations affected by publicly disclosed severe security vulnerabilities continued using the same version after 30 months, whereas 22% of observations not impacted by publicly disclosed severe security vulnerabilities continued using the same version.

The association displayed in Figure A1, however, suffers from reverse causality. The plot suggests that operating software with a severe security vulnerability is associated with updating in a less timely manner. Organizations that update software less frequently are more likely to accumulate bugs over time. This would make the decision to update and the presence of security vulnerabilities appear negatively correlated. Therefore, in our analysis in Table 4 column (3), we control for organizations' baseline propensity to update their software using stratefication.

Figure A1. Kaplan-Meier survival curve for the continued usage of software versions with and without severe security vulnerabilities



Appendix A3. Additional Justification for Stratification

In this appendix, we discuss further justifications for stratifying by organization.

A standard way to test whether stratifying by a given independent variable is needed in a Cox proportional hazard regression is to perform a test of the proportional hazards (PH) assumption. The PH assumption of the Cox model states that one individual's hazard function is proportional to another's. If a predictor does not satisfy the PH assumption, one can control for that predictor by stratifying that predictor (Kleinbaum and Klein, 1996). Testing the PH assumption based on weighted residuals (Grambsch and Therneau, 1994) is widely used and implemented in statistical packages for Cox proportional hazard analysis. For our analysis, we use the built-in cox.zph function in the survival package in R.

Ideally, to test if we need to stratify on the level of organizations, we should include the full set of organization dummies in a Cox model fit and determine whether the PH assumption is satisfied for each organization dummy. Given that our sample has 70,092 organizations, running Cox regressions with many variables is computationally challenging. We, therefore, test one organization dummy at a time. We run a Cox regression with a single organization dummy at a time and test whether the PH assumption is satisfied for that dummy. Suppose the p-value of that test is smaller than the 0.1 thresholds. In that case, we can reject at a 10% significance the hypothesis that the hazard function for that organization is proportional to the hazard function for other organizations.

We perform model fitting and testing of the PH assumption for a random 5% sample of our 70,092 organizations. Overall, we find that 1.7% of organization dummies have p-values smaller than 0.01, 7.8% have p-values smaller than 0.05, and 15.0% have p-values smaller than 0.1. We plot the distribution of p-values in Figure A3. It is visually evident that a substantial proportion of organization dummies do not satisfy the PH assumption. To control for organization effects in Cox regressions, we stratify by the organization.

We note that there are many approaches for model selection based on the likelihood of nested and non-nested Cox models, such as the likelihood ratio test, the Akaike information criterion, and so on (Fine, 2002). These approaches are inappropriate for deciding whether a stratified Cox model should be preferred over an unstratified one. The reason is that a stratified Cox model maximizes a different partial likelihood function than an unstratified Cox model. For example, for a standard unstratified Cox model without time-varying covariates, the partial likelihood function the estimation procedure maximizes is

$$L_p(\boldsymbol{b}) = \prod_i \left\{ \frac{\exp(b_1 X_{i1} + \cdots + b_j X_{ij})}{\sum_{k \in R(t_i)} \exp(b_1 X_{k1} + \cdots + b_j X_{kj})} \right\}^{d_i},$$

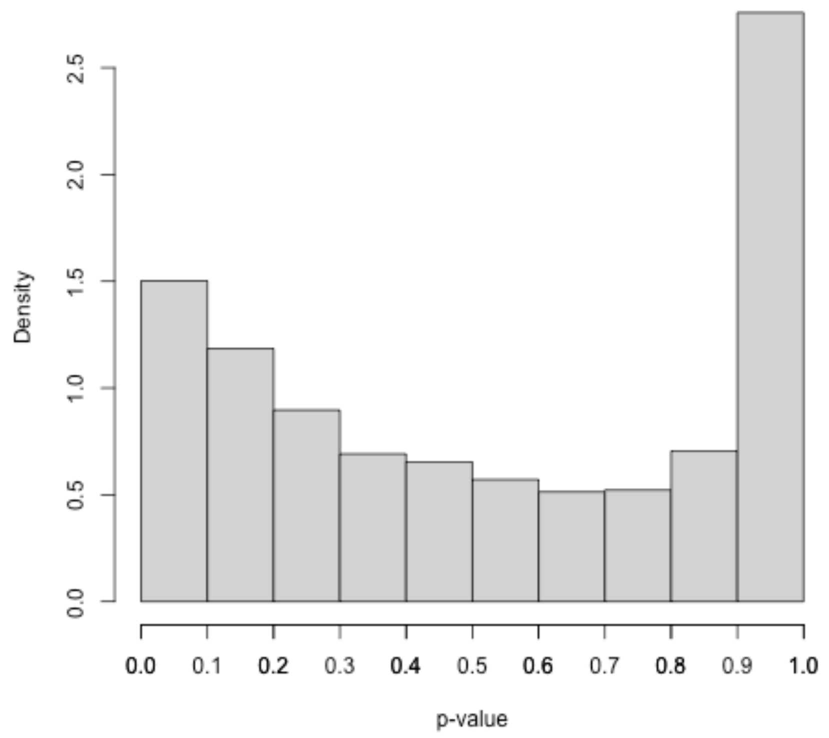
where d_i is the event, t_i is the survival time of individual i, and $R(t_i) = \{k: t_k > t_i\}$ is the risk set at time t_i.

For a Cox model stratified at level l and without time-varying covariates, the partial likelihood function the estimation procedure maximizes is

$$L_p(\boldsymbol{b}) = \prod_l \prod_{i \in A_l} \left\{ \frac{\exp(b_1 X_{i1} + \cdots + b_j X_{ij})}{\sum_{k \in R_l(t_i)} \exp(b_1 X_{k1} + \cdots + b_j X_{kj})} \right\}^{d_i},$$

where A_l is the set of individuals in stratum l, and $R_l(t_i)$ is the risk set for individuals in stratum l. When the number of strata equals 1, this likelihood function reduces to the likelihood function for the unstratified model. When the number of strata is greater than 1, since the denominators in this likelihood function are summed over a subset of the sample of individuals and are smaller, this likelihood is mechanically larger than the likelihood for the unstratified model at any given b and X.

Figure A2 Distribution of p-values of tests of proportional hazards (PH) assumption



Notes: We fit Cox regressions with one organization dummy at a time for a random 5% sample of our sample of 70,092 firms. We then test whether the organization dummy satisfies the PH assumption. 3,505 model fits correspond to 3,505 unique organizations in that 5% sample. The figure plots the distribution of the p-values from the PH assumption tests.

Appendix A4. Robustness to Excluding OS Specific Vulnerabilities

In the following tables, we drop Apache vulnerabilities that exclusively impact a specific operating system. For the sequential logit regressions, we exclude event windows around the releases of fixes to operating system-specific vulnerabilities. For the survival analysis, we exclude operating system-specific vulnerabilities in the construction of the $hasSevereBugs_{it}$ variable.

Table A3 Sequential logit regressions of cross-sectional determinants of user decisions

	Sequential logit regression		Sequential logit regression	
	(1)	(2)	(3)	(4)
	DN/DS vs FC	DN vs DS	DN/DS vs FC	DN vs DS
$infreqentUpdater_i$	0.152***	0.188***	0.135*	0.0392
	(0.0411)	(0.0461)	(0.0780)	(0.0757)
$frequentUpdater_i$	-0.236***	-0.105***	-0.210**	-0.244***
	(0.0449)	(0.0393)	(0.0999)	(0.0802)
$logEmployment_i$	0.0346**	0.00928	0.0115	0.00201
	(0.0171)	(0.0185)	(0.0333)	(0.0306)
$logRevenue_i$	-0.000832	0.00164	-0.00471	0.00114
	(0.00940)	(0.00737)	(0.0131)	(0.0136)
$isPublic_{iy}$	-0.0720	0.0962	-0.0704	0.0389
	(0.0848)	(0.0942)	(0.119)	(0.129)
$logPCs_i$	0.0438	0.0269	0.0176	0.0656
	(0.0307)	(0.0299)	(0.0575)	(0.0669)
$ITStaffCategory_i$	-0.0115	-0.0165	0.00920	-0.0224
	(0.0243)	(0.0293)	(0.0364)	(0.0421)
$logITBudget_i$	-0.0330	-0.0667*	0.0205	-0.0428
	(0.0446)	(0.0406)	(0.0967)	(0.0844)
$logSoftwareBudget_i$	0.0359	0.0747	-0.0333	0.0215
	(0.0445)	(0.0454)	(0.0866)	(0.0868)
$hasAlexaRank2010_i$	0.118**	-0.0262	-0.00695	0.0124
	(0.0566)	(0.0488)	(0.0881)	(0.0825)
$cloud_{it}$	-0.213*	-0.579***	-0.183	-0.545***
	(0.124)	(0.0958)	(0.208)	(0.171)
$countBreachState_{it}$	0.0134	0.0340	-0.0410	0.0324
	(0.0471)	(0.0492)	(0.0714)	(0.0657)
$logNoAffectedBreachState_{it}$	0.0365	0.0226	0.0567*	0.0279
	(0.0271)	(0.0242)	(0.0323)	(0.0294)
$countBreachInd_{it}$	0.00192	0.0528**	-0.000916	0.0205

	(0.0222)	(0.0211)		(0.0343)	(0.0381)
$logNoAffectedBreachInd_{it}$	-0.0131	0.0203***		0.0244*	0.0397***
	(0.0128)	(0.00725)		(0.0142)	(0.0130)
$outsourced_i = 1$ & $outsourcedMissing_i = 0$				0.175	0.172
				(0.180)	(0.160)
$outsourced_i = 0$ & $outsourcedMissing_i = 0$				0.0486	0.0251
				(0.0731)	(0.0679)
$techCategories_i$				0.0136	0.0227
				(0.0223)	(0.0191)
$techs_i$				-0.00647	-0.0141
				(0.0145)	(0.0150)
$monetizationTechs_i$				-0.00537	-0.0154
				(0.0278)	(0.0267)
Constant	1.661***	1.287***		2.214***	0.832**
	(0.132)	(0.145)		(0.350)	(0.341)
State dummies	Y	Y		Y	Y
Industry dummies	Y	Y		Y	Y
Observations	99,447	99,447		23,702	23,702

Notes: Same as those for Table 3.

Table A4 Cox proportional hazards analysis of updating decisions

	Cox proportional hazards regressions					
						Public firms
			Stratified	Stratified	Stratified	Stratified
	(1)	(2)	(3)	(4)	(5)	(6)
$hasSevereBugs_{it}$	-0.323***	-0.381***	-0.092***	-0.153***	-0.005	0.301
	(0.005)	(0.014)	(0.007)	(0.009)	(0.050)	(0.240)
$withinThreeMonthsSinceAddSevereBug_{it}$				0.144***		
				(0.009)		
$hasSevereBugs_{it} \times hasAlexaRank2010_i$					-0.061	0.038
					(0.037)	(0.172)
$hasSevereBugs_{it} \times cloud_{it}$					0.186*	0.659*
					(0.086)	(0.358)
$hasSevereBugs_{it} \times techCategories_i$					-0.004	-0.032
					(0.005)	(0.023)

$hasSevereBugs_{it} \times monetizationTechs_i$					-0.009	-0.043
					(0.008)	(0.042)
$newMinorVersionAvailable_{it}$	0.319***	0.361***	0.732***	0.845***	0.662***	0.564*
	(0.008)	(0.024)	(0.009)	(0.010)	(0.068)	(0.317)
$severeBugsFixed_{it}$				0.051***		
				(0.005)		
$nonsevereBugsFixed_{it}$				-0.019***		
				(0.002)		
$featureChanges_{it}$				-0.006***		
				(0.000)		
$newMinorVersionAvailable_{it} \times hasAlexaRank2010_i$					0.154***	0.742***
					(0.049)	(0.225)
$newMinorVersionAvailable_{it} \times cloud_{it}$					0.234	0.156
					(0.136)	(0.453)
$newMinorVersionAvailable_{it} \times techCategories_i$					-0.008	-0.055
					(0.006)	(0.034)
$newMinorVersionAvailable_{it} \times monetizationTechs_i$					0.021*	0.114*
					(0.011)	(0.063)
$newVersionAvailable_{it}$	0.073***	-0.065	0.367***	0.412***	0.270***	0.179
	(0.013)	(0.042)	(0.014)	(0.015)	(0.039)	(0.198)
$infrequentUpdater_i$		-0.115***				
		(0.015)				
$frequentUpdater_i$		0.188***				
		(0.015)				
$logEmployment_i$		-0.007				
		(0.006)				
$logRevenue_i$		0.003				
		(0.003)				
$isPublic_i$		-0.057				
		(0.028)				
$hasAlexaRank2010_i$		0.055***				
		(0.014)				
$cloud_{it}$		-0.010			-0.504***	-0.417
		(0.034)			(0.132)	(0.459)
$techCategories_i$		-0.013**				
		(0.004)				
$techs_i$		0.004				
		(0.003)				
$monetizationTechs_i$		0.013**				
		(0.004)				

$logPCs_i$		-0.016				
		(0.010)				
$ITStaffCategory_i$		0.014				
		(0.007)				
$logITBudget_i$		-0.010				
		(0.019)				
$logSoftwareBudget_i$		0.009				
		(0.019)				
$countBreachState_{it}$		-0.040***				
		(0.008)				
$logNoAffectedBreachState_{it}$		-0.010***				
		(0.002)				
$countBreachInd_{it}$		-0.029***				
		(0.003)				
$logNoAffectedBreachInd_{it}$		-0.013***				
		(0.001)				
$logCapxCompustat_{it}$						-0.038
						(0.029)
$logEmploymentCompustat_{it}$						0.033
						(0.027)
$logTotalAssetsCompustat_{it}$						-0.247**
						(0.078)
$logDepreciationCompustat_{it}$						-0.027
						(0.039)
$logIncomeCompustat_{it}$						0.000
						(0.003)
$logCashflowCompustat_{it}$						0.154
						(0.312)
State dummies	N	Y	N	N	N	N
Industry dummies	N	Y	N	N	N	N
Observations	3,104,455	548,959	3,104,455	3,049,753	665,360	44,955

Notes: Same as those for Table 4

Appendix A4. Robustness to Excluding Low Exploitability Vulnerabilities

In the following table, we drop Apache vulnerabilities that are scored by NIST as not having an exploitability score of 10 out of 10. For the sequential logit regressions, we exclude event windows around the releases of fixes to not highly exploitable vulnerabilities. For the survival analysis, we exclude not higly exploitable vulnerabilities in the construction of the $hasSevereBugs_{it}$ variable.

Table A5 Sequential logit regressions of cross-sectional determinants of user decisions

	Sequential logit regression		Sequential logit regression	
	(1)	(2)	(3)	(4)
	DN/DS vs FC	DN vs DS	DN/DS vs FC	DN vs DS
$infreqentUpdater_i$	0.179***	0.165***	0.213**	0.0519
	(0.0526)	(0.0522)	(0.0937)	(0.0883)
$frequentUpdater_i$	-0.232***	-0.108**	-0.110	-0.165**
	(0.0560)	(0.0436)	(0.0969)	(0.0822)
$logEmployment_i$	0.0437**	0.0125	0.0461	0.0363
	(0.0201)	(0.0267)	(0.0394)	(0.0358)
$logRevenue_i$	-0.00574	-0.00397	-0.0140	-0.0136
	(0.0111)	(0.00960)	(0.0176)	(0.0126)
$isPublic_{iy}$	-0.0340	0.0222	0.0349	-0.0187
	(0.0950)	(0.0915)	(0.150)	(0.133)
$logPCs_i$	0.0306	-0.00861	-0.0377	0.0514
	(0.0292)	(0.0412)	(0.0581)	(0.0658)
$ITStaffCategory_i$	-0.0103	0.0168	0.0163	-0.0211
	(0.0275)	(0.0298)	(0.0407)	(0.0407)
$logITBudget_i$	-0.0590	-0.0566	0.108	0.0141
	(0.0481)	(0.0443)	(0.105)	(0.0987)
$logSoftwareBudget_i$	0.0622	0.0650	-0.118	-0.0295
	(0.0494)	(0.0438)	(0.0998)	(0.102)
$hasAlexaRank2010_i$	0.230***	-0.00582	0.0468	0.0404
	(0.0733)	(0.0555)	(0.0956)	(0.0963)
$cloud_{it}$	0.613*	-0.124	0.514	0.129
	(0.364)	(0.273)	(0.539)	(0.398)
$countBreachState_{it}$	-0.170***	0.0129	-0.221*	0.0689
	(0.0553)	(0.0654)	(0.115)	(0.101)
$logNoAffectedBreachState_{it}$	0.0850**	0.0378	0.101**	0.0263
	(0.0391)	(0.0248)	(0.0453)	(0.0349)
$countBreachInd_{it}$	-0.0844**	0.0460	0.0452	0.0732
	(0.0415)	(0.0344)	(0.0828)	(0.0524)
$logNoAffectedBreachInd_{it}$	0.0120	0.0259***	0.0173	0.0326**
	(0.0130)	(0.00961)	(0.0202)	(0.0145)
$outsourced_i = 1$ &			0.0703	0.162
$outsourcedMissing_i = 0$			(0.233)	(0.223)
$outsourced_i = 0$ &			0.125	-0.0272

$outsourcedMissing_i = 0$				(0.0842)	(0.0773)
$techCategories_i$				0.00923	0.0213
				(0.0221)	(0.0210)
$techs_i$				-0.00647	-0.0134
				(0.0148)	(0.0169)
$monetizationTechs_i$				0.0154	-0.00888
				(0.0247)	(0.0297)
Constant	1.433***	1.540***		1.853***	0.856**
	(0.137)	(0.137)		(0.338)	(0.387)
State dummies	Y	Y		Y	Y
Industry dummies	Y	Y		Y	Y
Observations	70,464	70,464	18,198	18,198	70,464

Notes: Same as those for Table 3.

Table A6 Cox proportional hazards analysis of updating decisions

	Cox proportional hazards regressions					
						Public firms
			Stratified	Stratified	Stratified	Stratified
	(1)	(2)	(3)	(4)	(5)	(6)
$hasSevereBugs_{it}$	-0.163***	-0.258***	0.080***	0.021**	0.207***	0.264
	(0.005)	(0.014)	(0.007)	(0.009)	(0.049)	(0.239)
$withinThreeMonthsSinceAddSevereBug_{it}$				0.226***		
				(0.009)		
$hasSevereBugs_{it} \times hasAlexaRank2010_i$					-0.069*	0.191
					(0.036)	(0.174)
$hasSevereBugs_{it} \times cloud_{it}$					-0.088	-0.507
					(0.105)	(0.503)
$hasSevereBugs_{it} \times techCategories_i$					-0.005	-0.009
					(0.005)	(0.023)
$hasSevereBugs_{it} \times monetizationTechs_i$					-0.022**	-0.082*
					(0.008)	(0.041)
$newMinorVersionAvailable_{it}$	0.252***	0.305***	0.680***	0.800***	0.598***	0.581*
	(0.008)	(0.024)	(0.009)	(0.010)	(0.068)	(0.318)
$severeBugsFixed_{it}$				0.007		
				(0.005)		
				-0.012***		
$nonsevereBugsFixed_{it}$				(0.002)		

				-0.007***		
				(0.000)		
$featureChanges_{it}$				0.007		
				(0.005)		
$newMinorVersionAvailable_{it} \times hasAlexaRank2010_i$					0.158***	0.693***
					(0.049)	(0.227)
$newMinorVersionAvailable_{it} \times cloud_{it}$					0.315**	0.364
					(0.134)	(0.447)
$newMinorVersionAvailable_{it} \times techCategories_i$					-0.008	-0.064*
					(0.006)	(0.034)
$newMinorVersionAvailable_{it} \times monetizationTechs_i$					0.025**	0.130**
					(0.011)	(0.063)
$newVersionAvailable_{it}$	0.078***	-0.061	0.370***	0.422***	0.273***	0.185
	(0.013)	(0.042)	(0.014)	(0.015)	(0.039)	(0.198)
$infrequentUpdater_i$		-0.120***				
		(0.015)				
$frequentUpdater_i$		0.184***				
		(0.015)				
$logEmployment_i$		-0.008				
		(0.006)				
$logRevenue_i$		0.003				
		(0.003)				
$isPublic_i$		-0.057				
		(0.028)				
$hasAlexaRank2010_i$		0.055***				
		(0.014)				
$cloud_{it}$		-0.008			-0.456***	-0.239
		(0.034)			(0.131)	(0.458)
$techCategories_i$		-0.013**				
		(0.004)				
$techs_i$		0.004				
		(0.003)				
$monetizationTechs_i$		0.012**				
		(0.004)				
$logPCs_i$		-0.017				
		(0.010)				
$ITStaffCategory_i$		0.015				
		(0.008)				
$logITBudget_i$		-0.010				
		(0.019)				

$logSoftwareBudget_i$		0.009				
		(0.019)				
$countBreachState_{it}$		-0.038***				
		(0.008)				
$logNoAffectedBreachState_{it}$		-0.012***				
		(0.002)				
$countBreachInd_{it}$		-0.027***				
		(0.003)				
$logNoAffectedBreachInd_{it}$		-0.013***				
		(0.001)				
$logCapxCompustat_{it}$						0.039
						(0.029)
$logEmploymentCompustat_{it}$						0.037
						(0.027)
$logTotalAssetsCompustat_{it}$						-0.248**
						(0.078)
$logDepreciationCompustat_{it}$						-0.028
						(0.040)
$logIncomeCompustat_{it}$						0.000
						(0.003)
$logCashflowCompustat_{it}$						0.151
						(0.315)
State dummies	N	Y	N	N	N	N
Industry dummies	N	Y	N	N	N	N
Observations	3,104,455	548,959	3,104,455	3,049,753	665,360	44,955

Notes: Same as those for Table 4.