Navigating Software Vulnerabilities: Eighteen Years of Evidence from Medium and Large U.S. Organizations

Raviv Murciano-Goroff, Ran Zhuo, and Shane Greenstein

May 2024

Abstract

How prevalent are severe software vulnerabilities, how fast do software users respond to the availability of secure versions, and what determines the variance in the installation distribution? Using the largest dataset ever assembled on user updates, tracking server software updates by over 150,000 medium and large U.S. organizations between 2000 and 2018, this study finds widespread usage of server software with known vulnerabilities, with 57% of organizations using software with severe security vulnerabilities even when secure versions were available. The study estimates several different reduced-form models to examine which organization characteristics correlate with higher vulnerability prevalence and which update characteristics causally explain higher responsiveness to the releases of secure versions. The disclosure of severe vulnerability fixes in software updates does not jolt all organizations into installing them. Factors related to the cost of updating, such as whether the software is hosted on a cloud-based platform and whether the update is an incremental change or a major overhaul, play an important role. Observables cannot easily explain much variation. These findings underscore the urgent need to incorporate organizations' relative (in)attentiveness to act on software update releases into the design of cybersecurity policies.

Murciano-Goroff: ravivmg@bu.edu, Boston University; Ran Zhuo: ranzhuo@umich.edu, University of Michigan; Greenstein: sgreenstein@hbs.edu, Harvard Business School and NBER. We are grateful to Kenji Nagahashi and Mark Graham from The Internet Archive for providing the data for this research and appreciate the 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 and the Ewing Marion Kauffman Foundation for funding 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 administers. The authors take responsibility for all errors.

1. Introduction

Over the past twenty years, disruptive cyberattacks on companies have increased. (EY Americas 2021). Many cyberattacks exploit vulnerabilities in the software running on companies' servers, even when the software vendors previously acknowledged the vulnerabilities and provided updates to fix them (Ranger 2019). For example, in 2017, the UK's National Health Service fell victim to a cyberattack that exploited a vulnerability in their server software. A software update had been available to fix that vulnerability for over a month, but it had not been installed (Acronis International 2017; Palmer 2017). The cyberattack resulted in the cancellation of thousands of operations, including those of emergency patients. In the same year, attackers exploited a vulnerability in the server software hosting the Equifax website, exposing the information of over 143 million individuals (Goodin 2017). Again, a patch had been available for two months. In the wake of a ransomware attack on Atlanta in 2018 that halted many of the city's operations, an audit uncovered 1,500 to 2,000 vulnerabilities in the city's system. Some had been present for almost a year. (Harvey 2018; Goldenberg and Zlatev 2022).

These events garnered media attention and spurred calls for increased investment and diligence in cybersecurity (National Institute of Standards and Technology 2018). The policy proposals considered in the wake of these incidents took different forms. Some policies made software vendors liable for the cost that users face from installing patches. (August and Tunca 2011).[1] Some policies, such as mandated disclosure, encouraged vendors to release updates and patches more quickly (Arora, Telang, and Xu 2008; Arora et al. 2010). Others suggested restricting the frequency of software update releases and the amount of information software vendors disclose about vulnerabilities, decreasing the potential for malicious actors to learn from the vulnerability disclosures (Rescorla 2005; Mitra and Ransbotham 2015).

Whether or not these policies are beneficial depends on three related factors: untested assumptions about the *prevalence* of security vulnerabilities in installed and actively used server software, the

[1] August and Tunca (2011) study the provisioning of patches in an environment with profit-maximizing software vendors. Charging the "software vendor" for patch costs is impossible in our study setting. The software is open-source and created by a group of volunteers and a non-profit foundation.

determinants of organization decisions regarding whether to install software updates to fix vulnerabilities, and the *responsiveness* of organizations to attributes of the updates released, such as whether they respond faster to a high-impact vulnerability. These factors are consequential. *Prevalence* matters because mandating disclosure of vulnerabilities may decrease (increase) malicious attacks if users largely follow (ignore) mandates to install updates. The *determinants* of vulnerabilities matter as well. If they positively (negatively) correlate with the sensitivity of data and the related value of installing security software updates, then a policy of taxing (subsidizing) software usage could dissuade (encourage) organizations with poor (good) updating practices from using vulnerable software. Finally, suppose organizations adopt routines that respond to (ignore) information about the severity of risks. In that case, disclosing such information may speed up (have no effect on) user patching and reduce (increase) cybersecurity risks.

There is little empirical evidence on any of these three factors. That gap reflects the challenges of collecting and analyzing data about company software updating decisions over time and across different circumstances. Previous research and industry reports relied on data and surveys from small cross-sections of firms (AimPoint Group 2020; Positive Technologies 2020), but such data does not fully inform the discussion. For example, annual surveys of IT installations (e.g., Harte Hanks) lack information on the precise timing of update installations. At the same time, retrospective reports describe firms that have experienced hacking incidents but do not paint a picture of all firms. Indeed, to our knowledge, no data set provides information on software updates' prevalence, determinants, and responsiveness across a broad spectrum of organizations and over an extended period. Our goal is to address this gap.

This study analyzes detailed panel data that tracks vulnerabilities in web server software. Vulnerabilities in web server software offer a valuable lens for several reasons. 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). In addition to highlighting the prevalence of vulnerabilities, many server software updates include features

and bug fixes that improve the experience of Internet users. Increasing the speed companies respond to software updates could support productivity improvements and enhance the user experience.

The analysis focuses on organizations using the Apache HTTP Server (hereafter called "Apache"), the most popular web server software in the first few decades of the commercial Internet. Apache is also ideal for study because it publishes detailed information about its versions, updates, and vulnerabilities. From 2000 to 2018, 28 severe vulnerabilities and 130 less severe vulnerabilities were discovered in Apache. Each vulnerability reported to Apache developers was scored along multiple dimensions. During the same period, 115 software updates for Apache were released, along with a list of the security vulnerabilities being corrected, the bugs fixed, and the new features included. Apache is open source, and each update was made freely available to anyone online without restrictions on who could use it or how it could be used. That also has advantages for empirical analysis of the timing of installing these software updates, which is not confounded by the changes in the pricing of the software or the policies regarding the availability of updates.

The measurement of updates comes from raw data recorded on the Internet Archive's Wayback Machine, which has routinely visited millions of websites periodically 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 used to host each organization's website over time, it is possible to observe when an organization updates its server software or when an organization chooses to forgo updating and instead use the aging or vulnerable software. Our data include the websites of over 150,000 U.S. medium to large companies and organizations using Apache between 2000 and 2018. This is the most extensive data set assembled about user installations of software updates. Section 4 explains how the dataset is compiled after Sections 2 and 3 review relevant theory and background.

The analysis proceeds along three related lines. First, Section 5 assesses the prevalence of security vulnerabilities in the server software of organizations using Apache and finds widespread use of server software with severe security vulnerabilities. Between 2000 and 2018, 57% of organizations used software with severe security vulnerabilities even when secure versions were available. Almost every

Apache HTTP server hosting the organizations' homepages has operated with a publicly disclosed severe security vulnerability for some months. In other months, less than 25% contained one or more severe vulnerabilities. While the precise configuration of companies' server systems may prevent some of these vulnerabilities from being exploited by malicious actors, this finding suggests that a surprisingly large number of organizations operate their websites using outdated and potentially insecure software.

Second, Section 6 analyzes the determinants of organization decisions regarding whether to install software updates. Linear probability and Poisson regression analysis of the number of vulnerabilities show that factors related to the cost of updating, such as whether a company hosts their server on a cloud provider, can explain more of the variation than factors associated with the value of cybersecurity, such as being in an industry likely to handle sensitive personal data. In contrast, firms that stand to lose the most from a cyberattack, such as high-traffic websites and organizations whose websites have monetization technologies, are surprisingly more likely to have vulnerabilities in their software.

After the econometric specification controls for various organizational characteristics, few observables can account for the lion's share of variation in organizations' vulnerability prevalence. For example, an organization's industry, geography, and revenue do not strongly predict whether the organization accumulates vulnerabilities. Instead, persistent unobservable differences between organizations drive varying amounts of vulnerabilities accumulated.

Section 7 explores the r*esponsiveness* of organizations to attributes of the updates released, such as whether they respond faster to an especially severe vulnerability. It documents the characteristics of organizations and attributes of the Apache software updates associated with the faster or slower installation of those updates in the presence of newly discovered severe vulnerabilities. The best-fitting hazard model is a stratified hazard specification, which accounts for persistent unobservable differences between organizations. The estimates from this model show that organizations are faster at installing minor updates or updates that fix multiple severe security vulnerabilities. They are slower to install multifaceted and complex updates containing minor bug fixes and feature improvements.

The final section of this article, Section 8, discusses the policy and managerial implications of the findings. Since most organizations are predominantly inattentive to vulnerability disclosures, a policy mandating immediate, full disclosure may not be optimal from a societal standpoint. Policymakers should consider the delicate balance between alerting potential attackers and informing users of risks. A viable policy could involve releasing updates with key details emphasized, such as the exploitability of a vulnerability, while withholding less critical information, like the specific technical details of the fix. Although not a complete solution—since malicious actors might still investigate and analyze the source code of updates—a carefully designed limited disclosure policy could, to some extent, slow attackers (Mitra and Ransbotham, 2015) and benefit users who are slow to respond, compared to full disclosure. Our estimates also suggest that solutions should be unbundled. Complex security updates that incorporate many different minor bug fixes and feature improvements could hinder timely updating. It may be socially more efficient to separate security fixes from other types of fixes and enhancements—simple is better. Furthermore, the estimates imply that organizations should pay more attention to how technical complexity can inhibit their ability to stay secure. Reducing the cost of updating, such as by utilizing cloud hosting services, could facilitate more rapid updating.

This examination fills a critical empirical gap in cybersecurity literature. Much of the prior literature 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 exceptions, primarily based on interviews and surveys, most papers portray software users' decisions of when to install updates as deterministic or a function of update quality (Arora, Telang, and Xu 2008). No empirical evidence verifies these behavioral assumptions.

Previous work has acknowledged that not all firms immediately install patches after their release. However, these findings are based on limited descriptions of user-updating behavior from the selected firms. For example, Arbaugh, Fithen, and McHugh (2000) examined data on the vulnerabilities exploited by hacked firms. They lacked data on the fraction of firms operating with known vulnerabilities that did not get hacked. A body of security literature has examined the prevalence of vulnerabilities and attacks

within the web technology stack and user heterogeneity (Vasek, Wadleigh, and Moore 2016; Tajalizadehkhoob et al. 2017; West and Moore 2022; Jenkins et al. 2024). This paper contributes by linking vulnerability prevalence to a broad range of organizational, website, and software update characteristics and extending the analysis across many organizations over a substantial period.

This study also contributes by explaining the factors influencing the rate at which organizations install software updates. Like Dey, Lahiri, and Zhang (2015), this study's approach builds on the long-noticed phenomenon of considerable heterogeneity in the regular updating cycle and organizations' 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. Among companies that wish to eventually adopt software updates, a variety of frictions and costs can also cause delays in installing those updates (Dissanayake, Jayatilaka, et al. 2022; Dissanayake, Zahedi, et al. 2022; August and Tunca 2006; August, Niculescu, and Shin 2014; Kang 2022; Tiefenau et al. 2020).[2] This approach to empirical analysis enables inference as to why some organizations keep their server software close to the technological frontier and install updates promptly. In contrast, others seem to plod along, accumulating vulnerabilities.

In addition, this study answers a long-standing call for understanding the factors influencing organizations' responsiveness to the release of software updates. In their research on the tradeoffs of mandating faster disclosure of software vulnerabilities, Arora, Nandkumar, and Telang (2006) acknowledge the possibility that releasing a patch could increase the number of cyberattacks and called for empirically understanding the factors that hasten or slow user-patching as a promising area for future research. This study leverages the quasi-random discovery of vulnerabilities to analyze longitudinal data

[2] 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-premises 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 if 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).

to gain causal inferences on the factors impacting update decisions. This gives us insights into the attributes of software updates that influence server updating.

Only two prior empirical research studies have examined longitudinal investment in cybersecurity and linked it to outcomes. Li, Yoo, and Kettinger (2021) examined hospital adoption of security software and investment in related activities while Liu, Huang, and Lucas (2017) examined higher education and governance and associated actions. Both papers link these security investments to the propensity to suffer a security incident and exogenous organizational features and processes, using cross-sectional variance to infer causal determinants.[3] Unlike these, the data used in this analysis allows us to infer the relatively precise time each organization installed each software update released between 2000 and 2018. This level of detail enables us to estimate the impact of factors that increase (decrease) organizations' updating rate.

2. Theory of Server Software Users' Response to Software Updates

The prevalence of users operating software with known vulnerabilities, the determinants of decisions regarding whether to install software updates and the responsiveness of users to the release of updates that fix vulnerabilities are consequential factors for evaluating cybersecurity policies. This section provides a sketch of why these three factors impact the effectiveness of policies regarding cyber security proposed in the literature and motivates this study's empirical investigations of them.[4]

Prevalence. The prevalence of known vulnerabilities is a crucial empirical primitive to document and analyze. Arora, Caulkins, and Telang (2006) recommend that policymakers mandate that software vendors disclose known vulnerabilities within a relatively short time to motivate those vendors to produce

[3] Li, Yoo, and Kettinger (2021) stresses the returns at organizations that invest in on-premises processes, such as anti-virus, intrusion detection, and authentication. Liu, Huang, and Lucas (2017) 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.

[4] While the models vary somewhat, most have a similar setup. Typically, these models represent firms' decisions regarding installing a security update as a static problem. Firms have idiosyncratic values when using a particular piece of software. When a patch is released, firms that install the patch pay a fixed cost of patching but gain protection from the associated security vulnerability. In contrast, firms that do not patch face the expected cost of a hack of their systems. For vulnerabilities with negative externalities, such as when a hacked system may be used in a DDOS attack, the probability of the attack may be proportional to the fraction of other firms who also do not patch. In addition, the expected damage from an attack may be proportional to the firm's value from its system.

and release updates quickly. The benefits of disclosure must be weighed against the detrimental effects. The downside is obvious: mandated disclosure provides malicious actors information that could be useful for hacking systems that have yet to install the updates to fix associated vulnerabilities. However, if most organizations install updates when vulnerabilities are disclosed, then mandated disclosure will increase the provisioning of updates and improve the security of firm software. If most organizations forgo installing available updates, policymakers must be cautious with such a policy. As Choi, Fershtman, and Gandal (2010) note, mandated disclosure is beneficial only in contexts where malicious actors are slower to discover vulnerabilities than software vendors are to issue patches and users are to install them.

Determinants. Why does a population of software users display different prevalences of vulnerabilities within their installed software? Understanding the determinants of firm decisions regarding installing software updates, as well as the correlations between organization attributes and the prevalence of vulnerabilities, are essential for evaluating cybersecurity policies.

August and Tunca (2006) suggest several mechanisms to improve user incentives to install updates and patch vulnerabilities, including patching rebates and taxing software users to dissuade low-valuation firms who are unreliable patchers from abandoning the software rather than using it without installing updates. The optimality of those mechanisms depends on both the cost of patching and the value risked by not patching. For example, for freeware, August and Tunca (2006) conclude that a usage tax is the most effective policy except when both patching costs and value at risk are low, in which case a patching rebate prevails. Similarly, Cavusoglu, Cavusoglu, and Zhang (2008) study coordination mechanisms through which software vendors and users jointly manage patch release and adoption. They suggest that the effectiveness of the proposed mechanisms, namely cost sharing and liabilities, also depends on the cost of patching and the value at risk. If a large firm stands to lose significantly when a vulnerability is exploited and updates more frequently, then the socially optimal cost-sharing scheme between the vendor and that firm would have the vendor bearing less of the cost.

Responsiveness. The preceding concern directs attention to explaining why some organizations tolerate more vulnerabilities than others. A related issue focuses on understanding why organizations

respond more promptly to specific software updates than others and what factors influence this variability. Responsiveness is a critical element of many models used to evaluate cybersecurity policy.

Mitra and Ransbotham (2015) assess the optimal amount of information software vendors should disclose in updates. This decision involves a trade-off. On the one hand, as shown in Ransbotham, Mitra, and Ramsey (2012), providing information about vulnerabilities gives malicious actors information that could be used to attack systems. On the other hand, providing information about vulnerabilities may also instigate more organizations to install updates and patches if the details about the vulnerability frighten the organizations regarding the risks of not installing the associated updates. Moreover, disclosing information about new functionalities and features in software updates, in addition to vulnerability fixes, could either signal significant costs related to installation or the additional value of updating. Whether providing detailed information about vulnerabilities or new functionalities will be beneficial depends on whether detailed information and what kind of information induces firms to install updates.

3. Setting

This study focuses on the Apache HTTP Server software and the organization that supports it, the Apache Software Foundation (ASF). Four reasons motivate this focus. First, as the second most popular open-source project after Linux, Apache represents a significant component of the digital economy. Second, the ASF's processes for reporting, disclosing, and rectifying security vulnerabilities reflect the security practices typical of open-source software. Third, potentially severe and economically significant consequences could result from poorly secured Apache server software, so the setting has policy importance. Lastly, the setting enables the collection of highly detailed data on usage, vulnerability status, and updating behavior for a large group of U.S. organizations, enabling empirical analysis.

Server software like Apache is a computer program that enables users to host a website. When an individual visits an organization's website, the individual's web browser 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 and accesses their online banking accounts, the server software transmits the individual's login information and other sensitive personal financial data between the individual's web browser and the bank's backend software.

Apache's emergence as a popular server software makes it ideal for open-source and cybersecurity studies. Apache descended from the first server software. In 1993, the National Center for Supercomputing Applications (NCSA) at the University of Illinois developed a computer program called the NCSA HTTPd server, which supported sharing content on the newly diffusing World Wide Web. 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. In 1995, different teams of developers decided to coordinate their efforts into one server known as Apache (because it was "a patchy web server"). The University of Illinois then transferred the development to the ASF without licensing or restrictions. Apache became popular as the commercial internet grew. Murciano-Goroff, Zhuo, and Greenstein (2021) found that Apache was the most popular server software and powered 40% of the websites of medium to large U.S. organizations between 2000 and 2018.

The ASF coordinates the development of the Apache server software, receives reports of vulnerabilities, orchestrates the disclosure of vulnerabilities, and releases software updates to mitigate those vulnerabilities. Their vulnerability handling process has been typical of open-source software.[5] In the most standard scenario, the ASF accepts reports from users about potential vulnerabilities. A team of security experts vet these submissions, known as *reported vulnerabilities*. After evaluating a reported vulnerability, the ASF initially keeps the reported vulnerability secret from the public so malicious actors are kept from being tipped off about its existence. At the same time, teams of developers develop a fix. When the ASF believes it is prudent to do so, it publicly

[5] The statement for this process can be found at https://www.apache.org/security/. To the best of our knowledge, this process has not changed substantially since the founding of the Apache Software Foundation.

discloses the vulnerability. We refer to these as *disclosed vulnerabilities*. The ASF presumes Apache users are vigilant in monitoring their systems for vulnerabilities. When a fix for a vulnerability is developed, the ASF releases the fix as part of a new version of Apache. Users are responsible for deciding when to update their software, mitigating the risk of exploitation by malicious actors.

While this is the process that the ASF hopes will occur, some vulnerabilities are handled outside this procedure. Some vulnerabilities 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 vulnerability is reported and the date the ASF publicly discloses the vulnerability may be the same, and the ASF may disclose the vulnerability before a software update with the fix is ready to be released.

Many Apache vulnerabilities have severe consequences. Using data feeds provided by the ASF, the National Institute of Standards and Technology (NIST) scores vulnerabilities based on their potential to harm users.[6] We call vulnerabilities "severe" when the vulnerabilities score "high" for severity in the scoring system. These severe vulnerabilities are particularly harmful for two reasons. First, these vulnerabilities are easily exploitable. According to the scoring system, most severe security vulnerabilities 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 to access and exploit the vulnerability. Moreover, once exploited, these vulnerabilities can result in significant losses. These include and are not limited to partial or total disclosure of user information, modifying some or all the files in a system, reduced performance, or a complete system shutdown (Mell, Scarfone, and Romanosky 2007). As of August 1, 2018, among the 158 Apache vulnerabilities reported, 28 vulnerabilities scored "high" in severity.

Beyond Apache's widespread use, its process for managing vulnerabilities, and its cybersecurity policy significance, the context is enriched by the availability of data on Apache usage and update practices among a broad array of U.S. organizations. The ASF has made public extensive information on

[6] The ASF submits Apache vulnerabilities to the Common Vulnerabilities and Exposures (CVE), an international, community-based data registry. Using CVE's data feed, the NIST maintains the National Vulnerabilities Database (NVD). When a vulnerability is reported to CVE, 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#

each Apache version's vulnerabilities, fixes, and feature enhancements to enable precise measurement of users' vulnerability and fix status. These data aspects will be elaborated on in the following section.

4. Data

This section explains the essential and ancillary data sources. A summary of data sources is presented in Table 1. We defer the construction of our various samples to their respective sections later.

4.1. Key data source – server software usage panel of U.S. organizations

The critical data source is a broad panel data set that tracks server software used by medium to large organizations in the U.S. between 2000 and 2018. It records the usage of Apache and other server software, including Microsoft's IIS and Nginx, and tracks the installation of updates over time.

This panel data results from an extensive data collection process. It begins with information on all organizations in the Bureau van Dijk Mint Global database that have at least 50 employees in the U.S. and listed websites. Each organization's estimated number of employees, revenue, industry, and headquarters location are available. We treat organizations with the same website domain as one organization.

The Universal Resource Locator (URL) for each organization between 2000 and 2018 is matched with information from the Internet Archive (IA) Wayback Machine. The IA, a non-profit organization, has routinely scanned millions of publicly facing websites for the past two decades and taken snapshots of the content on those sites. When an individual connects to a website, the server software that hosts the site responds with the site's content and metadata about the server software. This metadata often contains the name of the server software and the server software version number (e.g., Apache 1.3.6).[7] The responding server also communicates its IP address, a sequence of numbers indicating where the server is located. The IA collects and stores this metadata, the IP addresses, and the date of each scan. We compiled the

[7] Users have a choice regarding how much information their server response headers show about their server software, ranging from no information to complete information, including the name, version, and operating system. Setting anything less than showing the server's name and version is not recommended. As the ASF puts it, "… [Obscuring server header] makes it more difficult to debug inter-operational problems. Also, note that disabling the Server header does nothing 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.

server software name, version number, IP address, and the date recorded in the metadata for each IA scan of the organizations in the sample of U.S. organizations.

The IA's scanning frequency is irregular. Sometimes, a site is scanned multiple times within a month; at other times, it may be scanned only once over several months. We retain only the first scan of the month for any organization with multiple scans. For sites scanned less frequently than monthly, this irregularity poses challenges to precisely measuring when updates occur. Although this does not affect much of the empirical analyses, the empirical study of how quickly organizations respond to available updates will be limited to a subset of organizations where the time to update can be precisely measured.

The empirical analysis will focus on Apache due to the abundance of publicly available information regarding its vulnerabilities, fixes, and feature enhancements. Equivalent data is unavailable for Microsoft IIS. We supplement with information about other organizational features and website characteristics.

4.2. Apache version characteristics, including vulnerability and fix status

It is necessary to understand how the versions of Apache are numbered. Apache versions are identified by three numbers separated by dots, for example, Apache 1.3.37. The first two numbers, such as Apache 1.3, represent the *major* version. Each major version introduces significant improvements in performance and functionality. Apache has had several major stable releases, including Apache 1.3 in 1998, Apache 2.0 in 2002, Apache 2.2 in 2005, and Apache 2.4 in 2012. The ASF simultaneously makes minor updates to various major versions, providing vulnerability fixes and incremental improvements. The third number in the version denotes the *minor* version within the major version. For instance, Apache 2.4.1 was released in February 2012, followed by 2.4.2 in April and 2.4.3 in August of the same year.

For each minor version, it is possible to gather information about its vulnerabilities from the ASF and NIST, including each vulnerability's severity, the date it was reported, the date it was disclosed to the public, and the release dates of new versions that fix the vulnerability. This enables determining whether each observation of an Apache minor version in the server usage panel had a *reported, disclosed, or fixed* severe security vulnerability at any given time and whether an updated major or minor version addressing the vulnerability was available then. In addition to vulnerabilities, we parsed Apache's changelogs, which

are documents summarizing changes made in each software version update, to obtain the number of new and improved features added to each new minor version compared to the previous minor version.

Based on the Apache version number, our vulnerability measure is not equivalent to the actual attack surface. This discrepancy arises due to the possibility of users taking alternative measures for vulnerability mitigation. For example, users may develop their own patches that address the vulnerability without changing the version number. Appendix A.4 explains the practice of backporting, which could secure the software for the short term without altering the version number. While we believe that backporting can alleviate some security concerns in the short run for a subset of users, we do not think this practice would significantly alter the core findings of our paper, as explained in Appendix A.4.

4.3. Organization Characteristics

The Bureau van Dijk Mint Global database provides a cross-sectional snapshot of the estimated number of employees, revenue, industry, and location for all organizations in the sample. The publicly traded firms have additional data about their annual operations. The data for these firms come from Compustat and cover the full panel of U.S. public firms annually. This data contains a wide range of organizational characteristics, such as total assets, capital expenditure, cash flow, and income, allowing us to examine organizational factors that might affect updating decisions. The data's temporal dimension also enables us to study the effects of financial changes within organizations. However, using this data source reduces the sample size, given that a small fraction of organizations in the sample are public firms.

Additional organizational characteristics are relevant to cybersecurity. The scale and complexity of an organization's IT operations are measured by the number of personal computers owned, the number of IT staff, the IT budget, and the software budget from Harte Hanks for 2017. Data from Harte Hanks also measures whether a subset of organizations outsourced their IT operations between 2005 and 2009.[8]

Data from Privacy Rights Clearinghouse's Data Breach Chronology is used to examine if data breach disclosures by organizations in the same geographic or industry sectors prompt others to secure

[8] Harte Hanks expanded their data collection gradually over time, and data in early years had limited mapping with our panel. The firm also only compiled data on IT outsourcing for a select group of organizations during 2005-2009.

their software. This data is the most comprehensive public source for breach information. In the U.S., data breach notification laws are enacted across all states, with most states adopting these laws in 2005 and 2006. This data source aggregates disclosures from media, state attorneys general offices, and breach disclosure trackers. It includes 2,366 breaches from 2005 to 2018, detailing the organization, disclosure date, state, and industry. For 63% of these breaches, the number of affected records was also reported.

4.4. Website Characteristics

To gauge an organization's website traffic, we obtained Alexa's traffic rankings for the top one million websites annually available to us from 2010 to 2018.

The IP address for a website enables the determination of whether an organization's website server software is likely cloud-based.[9] We obtained the IP addresses associated with major cloud providers and checked whether each organization's IP address belongs to a cloud provider. We did this for the major cloud providers Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform.

For a subset of the organizations' websites in the dataset, we have data on the web technologies used on those sites gathered from the HTTP Archive, which analyzed websites' technical attributes using an open-source tool developed by Wappalyzer starting in 2016. The data captures technology categories fundamental to website architecture, such as web frameworks and databases, as well as those supporting monetization and e-commerce, including marketing automation and payment processors. This information helps us construct proxies for the websites' technical complexity and the organizations' intent to monetize their websites.

5. Prevalence

5.1. Sample Construction

We apply two restrictions to the server usage panel dataset. First, the data set retains only the observations where Apache was used as the server software, excluding data related to other server

[9] We used snapshots of IP addresses associated with AWS, Microsoft Azure, and Google Cloud Platform taken on March 25, 2020, August 13, 2023, and August 14, 2023, respectively. Historical IP ranges of these cloud services are not available to the best of our knowledge.

software like IIS and Nginx. Second, the analysis applies only to observations where the complete Apache version number has been captured in the format of three numbers separated by dots. Although the ASF recommends that Apache respond to requests by revealing its full software version number, sometimes organizations either do not display the Apache version number or only show the major version, likely to conceal the exact software version they are using. The complete version number is crucial for mapping each version to its associated vulnerabilities. Applying these restrictions yields 4.9 million organization-month observations from 150,836 organizations. Creating the sample for the prevalence analysis requires combining this refined server usage panel with data on Apache version characteristics, such as the number of reported, disclosed, and already fixed vulnerabilities over time. Summary statistics of this sample are provided in Table 2.

5.2. Empirical strategy

To analyze vulnerability prevalence across organizations, we aggregate the sample at the monthly level and utilize line plots to document the extent and distribution of security vulnerabilities in the Apache server software used from 2000 to 2018. These plots show the proportions of organizations using Apache with reported, disclosed, and already fixed severe security vulnerabilities each month.

Analyzing the proportion of organizations with reported vulnerabilities is informative because these vulnerabilities are known to at least a small group of security experts, and the proportion represents the stock of organizations with vulnerable server software. Assessing the proportion with publicly disclosed vulnerabilities is critical; malicious parties could exploit this information to target organizations using server software with these vulnerabilities, thus highlighting the pool of organizations at risk of attack. Lastly, examining the proportion of organizations with vulnerabilities that are already fixed in newer versions sheds light on the stock of organizations that are slow to apply software updates, providing empirical evidence contributing to the debate on the benefits and costs of mandated vulnerability disclosures.

5.3. Results

Figure 1(a) depicts the proportion of organizations using Apache versions with reported severe security vulnerabilities over time. This plot is generated by aggregating across organizations the binary indicator variable, $severeBugReported_{it}$, which signifies whether severe security vulnerabilities have been reported for an organization-month. The figure demonstrates that a significant fraction of organizations' servers operated with these vulnerabilities. Over almost 20 years, the proportion of firms operating with these vulnerabilities averaged 68%, reaching nearly 100% between December 2015 and June 2017. This peak corresponds with the discovery of a critical security vulnerability (identifier: CVE-2017-7679) in a module responsible for assigning content metadata to HTTP responses. This finding indicates a high prevalence of organizations using Apache versions with known vulnerabilities.

Figures 1(b) and 1(c) decompose Figure 1(a) into the proportions of organizations with reported but undisclosed vulnerabilities and those with reported and disclosed vulnerabilities, respectively. This analysis helps assess whether the high prevalence of known vulnerabilities was due to slow disclosures by the ASF or organizations continuing to use vulnerable software versions even after disclosures.

As shown in Figure 1(b), no organizations had reported undisclosed vulnerabilities for most months, suggesting that disclosures occurred swiftly after reports within the same month. In some instances, disclosures did not occur in the same month as the reports but were still relatively prompt. This corresponds to the spikes, or the rapid increase and decrease in the proportion of organizations with undisclosed vulnerabilities observed between 2002 and 2010. One exception is from December 2015 to June 2017, when a significant proportion of organizations were affected by the reported but undisclosed vulnerability CVE-2017-7679. The ASF received this report in November 2015 but did not disclose and fix it until June 2017.

The line plot in Figure 1(c) is generated by aggregating the binary indicator variable, $severeBugDisclosed_{it}$, which signifies whether severe security vulnerabilities have been disclosed for the version in use for an organization-month. In contrast to Figure 1(b), Figure 1(c) aligns closely with

Figure 1(a), indicating that most organizations operating vulnerable Apache software did so not because of delayed disclosures by the ASF but because they continued to use the vulnerable software after disclosures were made. The figure shows that a significant fraction of organizations' servers operated with these vulnerabilities throughout the sample period. Over nearly 20 years, the proportion of firms operating with these vulnerabilities averaged 60%, peaking at 98% in October 2014. This peak corresponds to discovering a critical security vulnerability (identifier: CVE-2004-0885) in a module responsible for strong cryptography. The finding suggests a high prevalence of organizations using Apache with known and disclosed vulnerabilities. While the exploitability of these vulnerabilities depends on the precise configuration of each organization's operating system, and organizations could have applied temporary mitigation methods (e.g., self-developed patches, backports), the high prevalence of vulnerabilities suggests that firms are operating outdated and potentially insecure server software.

Figures 1(d) and 1(e) further decompose Figure 1(c) into the proportions of organizations with disclosed but unfixed vulnerabilities and those with disclosed and already fixed vulnerabilities, respectively. This analysis helps to assess whether the high prevalence of disclosed vulnerabilities was due to the ASF not offering fixes or because organizations continued to use vulnerable software even after newer versions with fixes were available. The former would suggest security mismanagement on the part of the ASF, which would call for policies for better governance of software developers. The latter would suggest security mismanagement by the organizations themselves, which would call for policies that promote more effective user updating behaviors.

Although a number of organizations dealt with disclosed vulnerabilities without fixes between 2003 and 2005, as shown in Figure 1(d), almost no organization faced disclosed but unfixed vulnerabilities for most months from 2006 to the end of the sample period. This indicates that the ASF has become more effective in managing disclosures and releases of new versions with fixes as it has matured. In contrast to Figure 1(d), Figure 1(e) is closely aligned with Figure 1(c). This shows that most organizations operating vulnerable Apache software did so not because the ASF delayed releases of new versions with fixes but because they continued to use the vulnerable software after fixes were made available.

Figure 2 further dissects Figure 1(e), displaying the fraction of organizations using Apache versions with one or more disclosed and fixed severe security vulnerabilities. The blue line plot in Figure 2 is identical to the line plot in Figure 1(e), generated by aggregating the binary indicator variable $severeBugFixed_{it}$, This signifies whether the organization's Apache version that month had severe security vulnerabilities already fixed in newer versions. Alarmingly, 57% of organizations in the sample used Apache versions with disclosed severe security vulnerabilities already fixed in newer versions. Yet, these organizations did not adopt the fixed versions. The proportion of these organizations almost never fell below 20% throughout the sample period.

Additionally, Figure 2 shows that many organizations operated Apache versions with multiple severe security vulnerabilities already fixed in newer versions. The number of such vulnerabilities is captured by the count variable, $numSevereBugFixed_{it}$. On average, organizations utilized Apache versions with two severe vulnerabilities that were already fixed in newer versions; a notable proportion operated with six or more. This highlights organizations' inaction. Apache fixed the software, but the companies did not make the local updates necessary to complete the fix.

Figure 2 also plots the release dates of fixed versions, represented by vertical dotted lines. For example, in July 2006, the ASF released Apache 1.3.37, Apache 2.0.59, and Apache 2.2.3. These releases fixed a severe vulnerability (identifier: CVE-2006-3747) that allows remote attackers to cause a denial-of-service attack. The red oval in Figure 2 highlights the update window following the July 2006 release. A closer examination of that update window reveals that many organizations took months or even years to update to those releases. The fraction of organizations using problematic versions of Apache peaked at 92% following the release, but the rate declined by only 1.7% per month over the next three years. By mid-2009, over 30% of organizations were still operating vulnerable versions of the Apache software.

5.4. Policy implications

The empirical results on the prevalence of Apache vulnerabilities offer abundant policy implications. The analysis reveals that the ASF has generally been prompt in disclosing and addressing vulnerabilities

upon their discovery. Instances where severe vulnerabilities remained undisclosed for more than two months post-reporting are scarce in the dataset. Additionally, while there was a period between 2003 and 2005 during which the ASF took longer to fix disclosed vulnerabilities, there has been a marked improvement in its security practices since 2006, with such instances becoming exceedingly rare. Their actions align with Arora, Caulkins, and Telang's (2006) recommendation to disclose vulnerabilities quickly to accelerate the development of patches and updates.

Conversely, the results highlight a significant risk linked to quick and/or compulsory disclosure stemming from users' delayed actions in updating to available fixed versions. Although swift disclosure may motivate software developers to produce fixes more quickly, it also risks leaving many users who are slow to respond susceptible to exploitation by malicious entities.

6. Determinants of Fixing Vulnerabilities

6.1. Sample Construction

To study the determinants of organizational differences in vulnerability prevalence, we combined the analysis sample for prevalence with various organizational and website characteristics that serve as explanatory variables. Table 2 shows summary statistics for the merged sample used in this section for analyzing the determinants of vulnerabilities in firms' server software.

Key outcome variables are $severeBugFixed_{it}$ and $numSevereBugFixed_{it}$, which capture the extent and number of severe vulnerabilities in the Apache versions organizations use. $severeBugFixed_{it}$ is the binary indicator variable that indicates whether an organization, in the given month, used an Apache version that had severe security vulnerabilities that were already fixed in newer versions. $numSevereBugFixed_{it}$ is a count variable, tallying the number of such severe security vulnerabilities in the Apache version used by the organization in the given month. These variables are the best at capturing differences in organizations' actions or inactions compared to the number of reported or disclosed vulnerabilities, which are also influenced by the actions of the ASF.

A range of explanatory variables captures the cost of patching, as motivated by August and Tunca (2006). The binary indicator variable $newUser_{it}$ signifies whether the observation represents the first instance of the organization's website using Apache, as captured by the IA. Unlike incumbent users of server software, new users of Apache are not constrained by previous technical investments and should, therefore, face lower costs when adopting the latest, vulnerability-free software versions. The variables PCs_i, $ITBudget_i$, and $SoftwareBudget_i$ reflect the number of computers, the IT budget, and the software-specific budget an organization had in 2017, retrieved from the Harte Hanks database. These variables proxy for the overall scale and complexity of IT operations within an organization. Specifically for the technical complexity of the website that the server software hosts, we utilize HTTP Archive's website technology data to construct variables $techCategories_i$ and $techs_i$ to denote the number of technology categories (e.g., JavaScript Frameworks, Marketing Automation) and the individual technologies (e.g., jQuery, Google Analytics) embedded within the website over the time we observe this data. The variables $outsourced_i$ and $cloud_{it}$ indicate whether the organization has outsourced its IT operations and whether the server software was likely hosted on a cloud provider, AWS, Azure, or Google Cloud.[10] Outsourcing and cloud hosting reduce an organization's day-to-day costs of monitoring and securing software, as many of the maintenance tasks are delegated to third parties.

A few variables reflect both the cost of updating and the value at risk. One such variable is $highTraffic_i$, a binary indicator variable that captures whether a website experienced high traffic based on Alexa rankings, defined as being in the top 100,000 most visited websites at any point during the time we observe this data. Another is $monetization_i$, a binary indicator variable showing whether a website has embedded monetization technologies for e-commerce, marketing automation, and payment

[10] We observe outsourcing status for only a few organizations between 2005 and 2009. We define $outsourcing_i$=1 if the organization has outsourced during 2005—2009 so the variable only has the organization subscript i. We also define a binary indicator variable $outsourcingMissing_i$ to indicate whether outsourcing information is missing for an organization. This variable will help us to keep observations with missing outsourcing information in regressions, allowing us to preserve a relatively large sample size and statistical power.

processing.[11] A high-traffic website or an e-commerce site is valuable, but updating is also more expensive due to the potential for interrupted services.

Additional variables reflect the value at risk from not updating, which was also motivated by August and Tunca (2006). These include binary indicator variables $finance_i$, $healthcare_i$, and $govt_i$ for organizations in the finance or healthcare sectors or public administration. These sectors are likely to process highly sensitive personal data, representing an exceptionally high value at risk. These variables also include $countBreachState_{it}$ and $countBreachIndustry_{it}$, which capture the number of data breaches in the same state or industry as the organization under observation in a given month. The variables $noAffectedBreachState_{it}$ and $noAffectedBreachIndustry_{it}$ capture the total number of personal records reported to have been compromised in those breaches. While data breaches do not directly increase the value at risk for organizations in the same state or industry, they could heighten the awareness of the value at risk. The estimation also includes standard organizational characteristics such as employment, revenue, and whether the organization was publicly listed. For publicly listed firms, additional variables are included, such as the firm's total assets and income each year.

6.2. Empirical Strategy

A linear probability model can estimate how different organizational and website characteristics affect whether the organization used an Apache version with severe security vulnerabilities already fixed. The endogenous variable is $severeBugFixed_{it}$. The regression specification is given by:

$$severeBugFixed_{it} = \beta_0 + \beta_1 X_{1,it} + \cdots + \beta_k X_{k,it} + \delta_t + \epsilon_{it}, \qquad (1)$$

where $X_{1,it}, \ldots, X_{k,it}$ represents the explanatory variables, including organizational and website characteristics, and δ_t denotes month-fixed effects. The inclusion of month-fixed effects is motivated by the patterns observed in Figure 2, which show that the months in which fixes were released resulted in significant shifts in the prevalence of fixed severe vulnerabilities among organizations.

[11] This variable is defined as 1 if the website has embedded any of the technology categories "analytics," "tag managers," "advertising networks," "marketing automation," "e-commerce," and "payment processors" during 2016—2018.

A Poisson regression model can estimate how different organizational and website characteristics influence the number of fixed severe security vulnerabilities in the Apache version used by organizations. The endogenous variable is $numSevereBugFixed_{it}$. Under this model, $numSevereBugFixed_{it}$ is presumed to follow a Poisson distribution, with the parameter $\lambda_{it} = E(numSevereBugFixed_{it})$ being a log-linear function of the explanatory variables $X_{1,it}, \ldots, X_{k,it}$:

$$\log\left(E(numSevereBugFixed_{it})\right) = \beta_0 + \beta_1 X_{1,it} + \cdots + \beta_k X_{k,it} + \delta_t + \epsilon_{it}. \quad (2)$$

Like the linear probability model, $X_{1,it}, \ldots, X_{k,it}$ includes organizational and website characteristics and δ_t denotes month-fixed effects.[12]

6.3. Results

Table 4 presents regression results for the linear probability model and the Poisson model across different combinations of explanatory variables. Columns (1) and (4) show the regression results for the linear probability and Poisson models, respectively when only the most well-populated explanatory variables are included. This approach enables the preservation of a large sample of organization months. Columns (2) and (5) introduce less well-populated explanatory variables. Lastly, Columns (3) and (6) show the results from adding organization-fixed effects to both models. Although the regressions utilize two distinct outcome variables, and sample sizes for each specification differ significantly, the results are remarkably consistent across the various specifications. Appendix Table A4 includes variables from Compustat about public firms, which effectively restricts the sample to these firms. The results from public firms are very similar to those from the full sample.

Variables capturing the cost of updating are good predictors of both the presence of fixed severe vulnerabilities and the number of fixed severe vulnerabilities. Being a new user is associated with a decrease of 3.3 to 4.7 percentage points in the probability of having any fixed severe vulnerabilities, and

[12] While the outcome variable $numSevereBugFixed_{it}$ shows evidence of overdispersion (it has a mean of 2.012 and a standard deviation of 2.312 as shown in Table 3), a fixed effects Poisson model is robust to overdispersion and is preferred over a fixed effects negative binomial model due to its robustness to both over- and underdispersion, heterogeneity in the variance-mean relationship across observations, violations of conditional independence, and serial correlation, which the fixed effects negative binomial model is not robust to (Wooldridge 1999).

with at least a $1 - \exp(-0.124) = 11.7\%$ reduction in the number of fixed severe vulnerabilities. Cloud-hosting is linked to a 10.9 to 19.9 percentage point decrease in the probability of having fixed severe vulnerabilities and at least a $1 - \exp(-0.414) = 33.9\%$ decrease in the number of fixed severe vulnerabilities. Both variables are significant at the 1% level across all specifications. While other variables that capture the cost of updating do not provide estimates as significant or consistent across different specifications, the direction of the estimates generally supports the hypothesis that organizations with higher costs of updating tend to have a higher prevalence of vulnerabilities. Organizations with more PCs are more likely to have severe vulnerabilities already fixed in newer versions, and a greater number of them, though the effect is not economically significant. Similarly, organizations that have not outsourced their IT operations are more likely to have such vulnerabilities and more of them, compared to those that have outsourced or about which we lack information. Evidence regarding website technical complexity is inconclusive, as estimates for $techCategories_i$ and $\log(techs_i + 1)$ are close to zero for the full sample and do not have a consistent direction for the public firms.

For variables that capture both the high cost of updating and the high value at risk, the estimates suggest that organizations are more concerned about the cost of updating than the value at risk. The estimates for $highTraffic_i$ are positive and highly significant across all specifications. These estimates indicate that websites with high traffic are 2.6 to 5.0 percentage points more likely to use Apache versions with severe vulnerabilities that are already fixed in newer versions and $\exp(0.036) - 1 = 3.7\%$ more of those vulnerabilities. Although the estimates for $monetization_i$ are less significant, the positive direction of these estimates is consistent with the hypothesis that the cost considerations of updating and making software secure outweigh the considerations regarding value at risk.

The results also suggest that the value at risk does not have a strong association with the prevalence of fixed severe vulnerabilities. Although the finance, healthcare, and public administration sectors are all likely to handle sensitive personal data, firms in these areas do not consistently display superior vulnerability prevalence. Data breach disclosures within the same state or industry have an economically

insignificant impact with inconsistent directionality. We have considered a range of organizational characteristics, such as revenue and whether the firm is publicly listed. Appendix Table A4 includes various characteristics of publicly listed firms, such as income and cash flow. However, none of these variables appear to be significant. The only notable finding is that larger organizations are more likely to use Apache versions with severe vulnerabilities already fixed in newer versions and to have a greater number of them.

The estimates suggest that a significant amount of unobserved organizational factors also influenced vulnerability prevalence. While the cost of updating is a strong predictor of prevalence, the magnitude of each individual estimate is small compared to the overall prevalence of vulnerabilities. Moreover, the explained variance in the outcome variable $severeBugFixed_{it}$, as indicated by the R-squared value, is only moderate. In Column (3) of Table 4, the inclusion of organization-fixed effects substantially increased the R-squared value. These results suggest that unobserved variations between organizations are likely essential drivers of differences in prevalence. Given the effectiveness of organization-fixed effects in improving model fit, these unobserved variations likely stem from persistent organizational differences, such as routines or management culture regarding cybersecurity.

6.4. Policy implications

These results carry substantial policy implications. First, they relate to theoretical models concerning user incentives for patching and optimal patch management, such as those by August and Tunca (2006) and Cavusoglu, Cavusoglu, and Zhang (2008). The cost of patching and the value at risk are crucial elements of these models that affect the optimality of different incentive schemes. The findings suggest that organizations with lower patching costs are more likely to patch. At the same time, those with a higher value at risk do not necessarily exhibit a greater likelihood of patching.

The results could assist policymakers in developing targeted policies. The findings imply that policies aimed at increasing the value at risk, such as a usage tax, may not be the most effective. Similarly, data breach disclosures or marketing campaigns designed to raise awareness of the value at risk from cyberattacks are unlikely to be effective. In contrast, policies that reduce the cost of patching for

users, such as cost-sharing, patching rebates, and providing IT training, may prove more effective. Among these policies, workshops or subsidies that help organizations migrate their software services to third-party professionally managed services and the cloud could reduce the prevalence of vulnerabilities.

The organizational effect is also an important consideration for policymakers, as organizations may exhibit persistence in their security management practices. This has implications for the relative effectiveness of policies aimed at incentivizing the installation of updates through cost and benefit mechanisms versus those targeting changes in the long-run organizational routines and managerial culture. Similar to the ideas of Cavusoglu, Cavusoglu, and Zhang (2008) that update releases should be aligned with firm updating cycles; our results regarding persistence in firm updating practices suggest that policy interventions that overlook organizational-level factors could have limited effectiveness.

7. Responsiveness

7.1. Sample Construction

Section 6 does not provide insights into the determinants of how quickly organizations respond to new updates post-release. The speed of updating deserves its own investigation. It could uncover additional factors that enable organizations to address severe vulnerabilities with software updates more promptly at certain times than at others—for example, the characteristics of the updates themselves. The approach involves analyzing the effects of characteristics of updates and characteristics of an organization and its website on the time taken to adopt fixes. To construct the sample, we identify the relevant time periods following the release of each critical update for the software versions used by organizations.

Figure 3 provides a visual explanation of the construction of the responsiveness sample. Panel (a) displays observations from the determinants of vulnerabilities sample for the firm Adobe. Each observation details the organization's identifier i ("adobe.com"), the month of IA capture t, and the version of Apache. The sample also includes other organizational and website characteristics not displayed in Panel (a). The dataset records Adobe's use of Apache from March 2001 to December 2002. These observations will be utilized to construct observations in the responsiveness sample pertaining to

Adobe. It should be noted that, due to the irregular capture frequency of the IA, there are gaps in our observations, such as in February 2002 and April 2002 for Adobe.

It is possible to know whenever a new version that fixes vulnerabilities in a particular Apache version becomes available based on the data on Apache security vulnerabilities. For example, between March 2001 and December 2002, two releases addressed the severe vulnerabilities in the Apache versions Adobe used. One of them was the release of Apache 1.3.24 in March 2002, which fixed a severe vulnerability (identifier: CVE-2002-0061) in the version in use, Apache 1.3.19, that allowed remote attackers to execute arbitrary commands. Panel (a) shows that by July 2002, Adobe.com had been updated to a version above 1.3.24, which fixed the vulnerability.

Using the relevant observations from Panel (a) for this updating event, as shown in the red box in Panel (a), we can construct one observation in the responsiveness sample. This observation is shown in the red box in Panel (b). The $timeToFix_{ir}$ variable represents the number of months it took the organization to adopt the fix, where the r subscript indexes the release (version 1.3.24 in March 2002). The $timeToFix_{ir}$ variable in this case equals 4. The $fixed_{ir}$ variable is a binary indicator, where it equals 1 if the organization adopts the fix and equals 0 if it does not adopt the fix by the final recorded observation (i.e., this updating cycle is right-censored). $fixed_{ir}$ in this case is 1. The variable $severeFixed_{ir}$ represents the number of severe vulnerabilities that the new update could fix for the version in use, which equals 1 in this case.

Another release that addressed severe vulnerabilities for Adobe was the release of Apache 1.3.27 in October 2002, which fixed two severe vulnerabilities (identifiers: CVE-2002-0839 and CVE-2002-0843) in the version Apache 1.3.26. We observed Adobe's Apache usage up to December 2002, and no updates occurred until then, as shown in the green box in Panel (a). This allows us to construct an observation in Panel (b), where $timeToFix_{ir}$ is recorded as 2, and $fixed_{ir}$ is set to 0 to reflect that the updating cycle is right-censored. The number of severe vulnerabilities fixed in that release, $severeFixed_{ir}$, is 2.

Another point to consider in the sample construction is handling gaps in IA captures. Although Figure 3 indicates gaps in IA captures for Adobe in February 2002 and April 2002, these did not affect the sample construction. We were able to deduce the time to fix precisely. Such precision is not attainable when significant gaps in an updating cycle occur. An updating cycle is observed to start with the release that fixed severe vulnerabilities in the version in use and is deemed to end with the first observed use of a version above that release. Suppose significant gaps are present within this cycle. In that case, it is conceivable that the organization might have been using the updated version for a substantial period before it was recorded in the data. Hence, only updating cycles for which there are IA captures at least once every two months on average are used to construct observations for the responsiveness sample.

We include additional variables. These capture the characteristics of the updates, as well as the organization and website characteristics. Table 5 presents the summary statistics for this sample.

Numerous variables represent the characteristics of the updates for each release that fixed severe vulnerabilities. The variable $nonSevereFixed_{ir}$ represents the number of non-severe vulnerabilities release r fixed in organization i's version in use at the time of the release. Meanwhile, $featureChanges_{ir}$ represents the cumulative number of feature changes between the version organization i used and release r. Fixes for minor vulnerabilities and feature changes can be valuable, but they can also add complexity and cost to software updates.

The variable $sameMajorVersion_{ir}$ is a binary indicator that equals one if the version in use and release r belong to the same major version (e.g., the version in use is 1.3.19, and the fix is released in 1.3.24) and equals 0 otherwise (e.g., the version in use is 1.3.19 and the fix is released in 2.0.37). Updates within the same major version represent small incremental changes that are less costly to install.

The binary indicator variables $highImpact_{ir}$ and $highExploitability_{ir}$ indicate that the update fixed vulnerabilities with high impact and high exploitability, respectively. High-impact vulnerabilities, which scored 10 out of 10 in impact according to NIST's scoring rubric, could result in total disclosure of information, a total compromise of system integrity that allows attackers to modify any files, and a

complete shutdown of the system. High exploitability vulnerabilities, which scored 10 out of 10 in exploitability by the same rubric, would allow attackers to attack over the network without authentication requirements, and the attack has low complexity once an attacker has gained access to the target system.

The variable $notOSSpecific_{ir}$ is a binary indicator that indicates that at least one of the severe vulnerabilities fixed by the release is not specific to a particular operating system, making the vulnerability more general.

The variable $prevTimeToFix_{ir}$ represents the time the organization took to update to the preceding release that addressed severe vulnerabilities. It investigates the persistence of organizations in their vulnerability patching practices. If organizations were not persistent, those slow to update in the previous updating cycle should aim to rapidly update in the current cycle to compensate. Conversely, suppose organizations are burdened with consistent organizational costs, such as a poorly organized IT department. Then, they will exhibit delays in the current cycle if they were slow previously.

The responsiveness sample also includes the range of organization and website characteristics, for example, $newUser_{it}$ and PCs_i. Whenever these variables carry a time subscript, the value of the variable on the release date is used to fill that variable in the responsiveness sample.

7.2. Empirical Strategy

Because the outcome of interest is a time-to-event variable—the amount of time from the release until the organization adopts the release—survival models can estimate what predicts faster or slower updating. We first estimate the standard Cox proportional hazards regression model, specified as follows:

$$h_{ir}(t) = h_0(t) exp\left(b_1 X_{1,ir} + b_2 X_{2,ir} + \cdots + b_j X_{j,ir}\right), \qquad (3)$$

where $h_{ir}(t)$ is the hazard function, representing the expected number of updates to the fixed version given that the vulnerable version has survived for t months. $h_0(t)$ is the baseline hazard and represents the hazard when all the explanatory variables $X_{1,ir}, \ldots, X_{j,ir}$ are equal to zero. If some explanatory variables carry a time subscript t or y that varies within an updating cycle, the value of the variable on the release date is used in the regressions.

The model described above does not account for unobserved differences in the organizations' security management practices, which could lead to reverse causality issues. Specifically, the model in Equation (3) assumes the baseline hazard. $h_0(t)$ is the same for all organizations. However, organizations that are persistently slow to update for unobserved reasons are more likely to use older versions of Apache, which have more severe security vulnerabilities that need fixing. If we assume a uniform baseline hazard across all organizations and estimate a Cox model, the estimate for $severeFixed_{ir}$ would likely be biased downward. We might incorrectly conclude that a release addressing a greater number of severe security vulnerabilities would be associated with a longer time to adopt the release.

To control for unobservable differences in organizations' security management practices, the preferred specification is a stratified Cox model. Each organization has a different baseline hazard:

$$h_{ir}(t) = h_{0i}(t)exp\left(b_1X_{1,ir} + b_2X_{2,ir} + \cdots + b_jX_{j,ir}\right), \qquad (4)$$

where the i subscript in $h_{0i}(t)$ denotes stratum for organization i. Under this model, the effect of a variable is identified by the changes to that variable within an organization across different releases.

The estimated effect of update characteristics on the time-to-fix from Equation (5) is causal because the reporting, disclosures, and release of fixes for severe security vulnerabilities and the characteristics of each update released are plausibly exogenous to an organization's IT staff. The crucial assumption in this context is intuitive. Most web developers and IT professionals are not directly involved in developing Apache server software or in its vulnerability handling process. The influence of any single organization (aside from the ASF) is negligible compared to the total global interactions. To ensure that individual organizations' actions do not disproportionately affect vulnerability handling, we examined the data on who was credited with reporting each vulnerability to the ASF. Aside from the ASF Security Team staff, there is almost no overlap in the names and organizational affiliations of the reporters.

7.3. Results

Estimation results are presented in Table 5. They provide evidence that the stratified Cox model is preferable. Column (1) displays the standard Cox regression results, using only update characteristics as

explanatory variables. As expected, the estimate for $severeFixed_{ir}$ is biased downward due to unaccounted-for organizational effects. Column (2) incorporates more comprehensive organization and website characteristics, and Column (3) includes additional, less populated organizational and website characteristics. As these controls are added, the coefficient becomes increasingly positive. This suggests that accounting for organizational effects is crucial for mitigating reverse causality issues and ensuring correct inferences. Column (4) presents the estimates from the stratified Cox model, stratified at the organization level, using only release characteristics as explanatory variables. Column (5) presents the stratified Cox model with additional controls for time-varying organization characteristics. Now, the estimates for $severeFixed_{ir}$ are large, positive, and highly significant. This suggests that organizations respond more quickly to an increase in the number of severe vulnerabilities a release fixes.

Another piece of evidence supporting the effectiveness of the stratified Cox model in controlling for unobserved organizational effects comes from the estimates of the variable $prevTimeToFix_{ir}$. This variable captures the persistence of organizations in their behavior between adjacent updating cycles. The estimates for this variable are large, negative, and highly significant in the standard Cox models. They suggest that a one-month increase in the time to fix in the previous updating cycle for an organization would be associated with at least a $1 - exp(-0.018) = 1.8\%$ decrease in the expected hazard of updating in the current cycle. The estimates become economically negligible in the stratified Cox models.

Based on the preferred specification, the stratified Cox models, organizations have varying levels of responsiveness to different characteristics of updates. The estimates show that organizations are responsive to fixing severe vulnerabilities that enhance the security of their software. An additional severe vulnerability being fixed in a new release would result in a $exp(0.073) - 1 = 7.6\%$ to $exp(0.170) - 1 = 18.5\%$ increase in the expected hazard of updating to that release.

Organizations are averse to characteristics of updates that increase the complexity of updating, even when those characteristics offer benefits, such as minor bug fixes and feature improvements. An additional non-severe vulnerability being fixed in the new release would result in at least a $1 -$

$exp(-0.066) = 6.4\%$ decrease in the expected hazard of updating to the release. Furthermore, a 10% increase in the number of feature changes in the release, relative to the version in use, would result in at least a $1 - exp\left(-\frac{0.075}{100} * 10\right) = 0.7\%$ decrease in the expected hazard of updating. This contrast suggests that organizations perceive the benefit of updating to fix severe vulnerabilities to outweigh the cost, which is not the case for non-severe bugs and feature improvements.

Organizations are more responsive to new releases that consist of small incremental changes, in contrast to major upgrades in performance and features. When the release is a minor new version within the same major version as the version in use, organizations exhibit at least a $exp(1.150) - 1 = 215.8\%$ higher expected hazard of updating compared to releases that are part of a different major version.

Organizations are more responsive to highly exploitable severe vulnerabilities than those that address highly impactful vulnerabilities. While the estimates for high-impact vulnerabilities are not consistently significant, a release that addresses highly exploitable severe vulnerabilities is associated with at least a $exp(0.233) - 1 = 26.2\%$ increase in the expected hazard of updating to that release.

7.4. Policy implications

The attributes of updates influence their adoption (F. Li et al. 2019). Our results offer important empirical evidence for the policy debate on designing effective vulnerability disclosure policies. These estimates demonstrate that full disclosure is unlikely to be optimal. While organizations respond to releases that announce fixes for severe vulnerabilities, the evidence suggests they avoid releases disclosing fixes for non-severe ones due to the greater costs and complexity of adoption, which outweigh the perceived benefits. Thus, a robust vulnerability disclosure policy that aims to minimize vulnerability exposure, maximize patching behavior, and enhance societal welfare could consider disclosing fixes for severe vulnerabilities but not minor ones.

Moreover, the empirical evidence suggests that organizations are very cost-sensitive when making vulnerability patching decisions. Announcements of feature improvements could increase the perceived

costs and complexity of installation. Software developers should reconsider how much information about feature improvements to disclose to avoid discouraging patching behavior.

Additionally, developers should consider releasing small incremental updates or standalone patches for highly severe vulnerabilities to encourage patching rather than bundling the patch with numerous new features in a major release, which could serve as a deterrent. Furthermore, developers should emphasize exploitability when disclosing fixes for severe vulnerabilities in new releases. This is likely more effective than simply discussing the potential impact if the vulnerability were to be exploited.

Our findings also highlight the significant role of persistent, unobserved organizational effects in shaping organizations' responsiveness to adopting fixes. This suggests that even optimally crafted disclosure policies may achieve limited success for the laggards unless there is a fundamental transformation in organizational routines and culture regarding cybersecurity.

8. Conclusion and Discussion

This study examined security vulnerabilities in open-source server software used by over 150,000 organizations in the United States between 2000 and 2018. This is the largest data set assembled on security vulnerabilities and updates. The study sought to understand previously underexplored research questions at the heart of assessing which cybersecurity policies will likely be effective. Specifically, we examined how prevalent severe vulnerabilities are in server software, what determines the variance in the installation distribution, and how fast software users respond to the availability of secure versions.

The empirical analysis revealed four critical findings. First, the study finds widespread usage of server software with known vulnerabilities. While the exploitability of these vulnerabilities depends on various factors, the high prevalence suggests that there may be many opportunities for malicious actors to exploit organizations' web servers. Second, the prevalence of vulnerabilities in the software used to host companies' websites is associated more with factors related to the cost of updating than factors related to the value of security. Third, observables cannot easily explain the large variation in the prevalence of vulnerabilities in company server software. Instead, persistent, unobservable aspects of organizations, such as organizational routines, explain much of the variation in the presence of vulnerabilities. Finally,

firms are responsive to updates with security fixes, but they are slower to install multifaceted and complex updates.

These findings inform previous theoretical work on cybersecurity policy. First, scholars and policymakers have contemplated mandating the disclosure of software vulnerabilities to instigate faster releases of software updates patching vulnerabilities (Arora, Telang, and Xu 2008). Our data revealed, however, that organizations are slow and unthorough when installing updates. This finding gives reason to be cautious about such a policy. Second, policymakers have considered either providing patching rebates to defray the cost of installing updates or taxing software users to dissuade low-value firms from using software in insecure ways (August and Tunca 2006). Our finding that the presence of vulnerabilities in server software is associated with high costs of updating implies that software update rebates may be more effective at increasing cybersecurity practices than policies aimed at increasing the value of updating or highlighting the risk of vulnerability. In addition, organizations can take actions that reduce the cost of installing updates, such as hosting their website on a cloud-based platform that assists with installing updates and decreasing the technological complexity of their website. Furthermore, the finding that much of the variation in the presence of vulnerabilities is explained by persistent unobservable attributes of firms implies that policymakers and managers may wish to focus on organizational routines and culture to improve cybersecurity. For example, managers should consider if a routine that includes updating at regular intervals is beneficial for their organization. Finally, a long-running theme in cybersecurity literature questions the amount of information software vendors should release about vulnerabilities in their software (Mitra and Ransbotham 2015). The results of the hazard model analysis reinforce that software vendors can also design the release of software updates in ways that are more likely to have those updates adopted. Specifically, when patches fixing severe vulnerabilities are packaged in an update alone, they are more likely to be installed in a timely manner than if those updates contain feature updates or minor bug fixes.

This study does have limitations. First, it only examines Apache web servers. While open-source server software operates on most servers today (Greenstein and Nagle 2014), installing updates on

proprietary server software, especially from software vendors that automatically send updates to users, may be different. Whereas extensive and detailed data were available for this study because Apache server software is open source, proprietary software vendors are less transparent about their products and users. Future research on the usage of proprietary software would enable a more complete picture of software user behavior in general.

Second, there are many challenges matching variance in server software decisions at organizations with variance in 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 the data with information in the World Management Survey, the overlap in samples provides limited statistical power for analysis, and the select sample of matching firms constrains the external validity of any findings. Future researchers should seek to find ways to expand on the insights by attempting to understand how managerial practices more broadly influence IT and cybersecurity investments.

Third, we cannot definitively say if firms are making rational, calculated decisions or if the firms' decisions are the reflection of inattention. For example, a firm that forgoes installing an available software update may be aware that their firm is unlikely to be targeted by malicious actors or knows that the idiosyncratic software configuration on their server mitigates the vulnerabilities fixed in that update. While we include a measure of the traffic to a firm's website in our regressions, as this is likely to be correlated with attention from malicious actors, we cannot observe the same information that firm managers possess. Thus, we cannot judge their decisions. Instead, our data highlights that a large share of organizations operate server software with vulnerabilities that could be exploited, which policymakers may find to be socially less than ideal. Future researchers could collect additional data to illuminate this topic further.

This study sets the stage for future research on a variety of cybersecurity-related topics. First, more research can be done on factors that instigate changes in cybersecurity practices. For example, future research should examine how executive leadership changes, labor market changes for IT professionals, and business cycles impact firms' cybersecurity postures. Second, more research should be done on the

effect of cybersecurity service vendors on vulnerabilities. While this seems like a rich area for research, additional data collection on the timing and usage of these services will be required. Future research should better understand the interaction of firm strategy, the competitiveness of firms' markets, and cybersecurity investment decisions. Finally, additional data on the cybersecurity threats faced by firms, including the probability of an attack and the damages from actual incidents, could provide further insight into the determinants of updating decisions and the rationality of organizations regarding these decisions.

References

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/.

AimPoint Group. 2020. "Cyber Hygiene Report." https://patch.automox.com/rs/923-VQX-349/images/Automox_2020_Cyber_Hygiene_Report-What_You_Need_to_Know_Now.pdf.

Arbaugh, William A, William L Fithen, and John McHugh. 2000. "Windows of Vulnerability: A Case Study Analysis." *Computer* 33 (12): 52–58. https://doi.org/10.1109/2.889093.

Arora, Ashish, Jonathan P. Caulkins, and Rahul Telang. 2006. "Research Note: Sell First, Fix Later: Impact of Patching on Software Quality." *Management Science* 52 (3): 465–71.

Arora, Ashish, Ramayya Krishnan, Rahul Telang, and Yubao Yang. 2010. "An Empirical Analysis of Software Vendors' Patch Release Behavior: Impact of Vulnerability Disclosure." *Information Systems Research* 21 (1): 115–32. https://doi.org/10.1287/isre.1080.0226.

Arora, Ashish, Anand Nandkumar, and Rahul Telang. 2006. "Does Information Security Attack Frequency Increase with Vulnerability Disclosure? An Empirical Analysis." *Information Systems Frontiers* 8 (5): 350–62. https://doi.org/10.1007/s10796-006-9012-5.

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

August, Terrence, Marius Florin Niculescu, and Hyoduk Shin. 2014. "Cloud Implications on Software Network Structure and Security Risks." *Information Systems Research* 25 (3): 489–510.

August, Terrence, and Tunay I. Tunca. 2006. "Network Software Security and User Incentives." *Management Science* 52 (11): 1703–20. https://doi.org/10.1287/mnsc.1060.0568.

———. 2011. "Who Should Be Responsible for Software Security? A Comparative Analysis of Liability Policies in Network Environments." *Management Science* 57 (5): 934–59.

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, Jay Pil, Chaim Fershtman, and Neil Gandal. 2010. "Network Security: Vulnerabilities and Disclosure Policy." *The Journal of Industrial Economics* 58 (4): 868–94.

Dey, Debabrata, Atanu Lahiri, and Guoying Zhang. 2015. "Optimal Policies for Security Patch Management." *INFORMS Journal on Computing* 27 (3): 462–77.

Dissanayake, Nesara, Asangi Jayatilaka, Mansooreh Zahedi, and M. Ali Babar. 2022. "Software Security Patch Management - A Systematic Literature Review of Challenges, Approaches, Tools and Practices." *Information and Software Technology* 144 (April):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." *Proceedings of the ACM on Human-Computer Interaction* 6 (CSCW2): 1–29. https://doi.org/10.1145/3555087.

EY Americas. 2021. "Cybersecurity: How Do You Rise above the Waves of a Perfect Storm?" https://www.ey.com/en_in/cybersecurity/cybersecurity-how-do-you-rise-above-the-waves-of-a-perfect-storm.

Goldenberg, Amit, and Julian Zlatev. 2022. "Atlanta Ransomware Attack (A)." *Harvard Business School Case Collection*, August. https://www.hbs.edu/faculty/Pages/item.aspx?num=62893.

Goodin, Dan. 2017. "Failure to Patch Two-Month-Old Bug Led to Massive Equifax Breach." Ars Technica. September 14, 2017. https://arstechnica.com/information-technology/2017/09/massive-equifax-breach-caused-by-failure-to-patch-two-month-old-bug/.

Greenstein, Shane, and Frank Nagle. 2014. "Digital Dark Matter and the Economic Contribution of Apache." *Research Policy* 43 (4): 623–31. https://doi.org/10.1016/j.respol.2014.01.003.

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

Jenkins, Adam D G, Linsen Liu, Maria K Wolters, and Kami Vaniea. 2024. "Not as Easy as Just Update: Survey of System Administrators and Patching Behaviours." In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 1–17. Honolulu HI USA: ACM. https://doi.org/10.1145/3613904.3642456.

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. https://doi.org/10.1093/icc/dtac039.

Li, Frank, Lisa Rogers, Arunesh Mathur, Nathan Malkin, and Marshini Chetty. 2019. "Keepers of the Machines: Examining How System Administrators Manage Software Updates for Multiple Machines." In *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*, 273–88. Santa Clara, CA: USENIX Association. https://www.usenix.org/conference/soups2019/presentation/li.

Li, He, Sungjin Yoo, and William J. Kettinger. 2021. "The Roles of IT Strategies and Security Investments in Reducing Organizational Security Breaches." *Journal of Management Information Systems* 38 (1): 222–45. https://doi.org/10.1080/07421222.2021.1870390.

Liu, Che-Wei, Peng Huang, and Henry Lucas. 2017. "IT Centralization, Security Outsourcing, and Cybersecurity Breaches: Evidence from the U.S. Higher Education." *ICIS 2017 Proceedings*, December. https://aisel.aisnet.org/icis2017/Security/Presentations/1.

Mell, Peter M., Karen A. Scarfone, and Sasha Romanosky. 2007. "A Complete Guide to the Common Vulnerability Scoring System Version 2.0." *NIST*, July. https://www.nist.gov/publications/complete-guide-common-vulnerability-scoring-system-version-20.

Mitra, Sabyasachi, and Sam Ransbotham. 2015. "Information Disclosure and the Diffusion of Information Security Attacks." *Information Systems Research* 26 (3): 565–84.

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, and Shane Greenstein. 2021. "Hidden Software and Veiled Value Creation: Illustrations from Server Software Usage." *Research Policy* 50 (9): 104333.

National Institute of Standards and Technology. 2018. "Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1." NIST CSWP 04162018. Gaithersburg, MD: National Institute of Standards and Technology. https://doi.org/10.6028/NIST.CSWP.04162018.

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/.

Positive Technologies. 2020. “Vulnerabilities on the Corporate Network Perimeter.” https://www.ptsecurity.com/ww-en/analytics/vulnerabilities-corporate-networks-2020/.

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, Mitra, and Ramsey. 2012. “Are Markets for Vulnerabilities Effective?” *MIS Quarterly* 36 (1): 43. https://doi.org/10.2307/41410405.

Rescorla, E. 2005. “Is Finding Security Holes a Good Idea?” *IEEE Security and Privacy Magazine* 3 (1): 14–19. https://doi.org/10.1109/MSP.2005.17.

Tajalizadehkhoob, Samaneh, Tom Van Goethem, Maciej Korczyński, Arman Noroozian, Rainer Böhme, Tyler Moore, Wouter Joosen, and Michel Van Eeten. 2017. “Herding Vulnerable Cats: A Statistical Approach to Disentangle Joint Responsibility for Web Security in Shared Hosting.” In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 553–67. Dallas Texas USA: ACM. https://doi.org/10.1145/3133956.3133971.

Tiefenau, Christian, Maximilian Häring, Katharina Krombholz, and Emanuel von Zezschwitz. 2020. “Security, Availability, and Multiple Information Sources: Exploring Update Behavior of System Administrators.” In *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*, 239–58. USENIX Association. https://www.usenix.org/conference/soups2020/presentation/tiefenau.

Vasek, Marie, John Wadleigh, and Tyler Moore. 2016. “Hacking Is Not Random: A Case-Control Study of Webserver-Compromise Risk.” *IEEE Transactions on Dependable and Secure Computing* 13 (2): 206–19. https://doi.org/10.1109/TDSC.2015.2427847.

West, Jonathan Codi, and Tyler Moore. 2022. “Longitudinal Study of Internet-Facing OpenSSH Update Patterns.” In *Passive and Active Measurement*, edited by Oliver Hohlfeld, Giovane Moura, and Cristel Pelsser, 13210:675–89. Lecture Notes in Computer Science. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-98785-5_30.

Wooldridge, Jeffrey M. 1999. “Distribution-Free Estimation of Some Nonlinear Panel Data Models.” *Journal of Econometrics* 90 (1): 77–97. https://doi.org/10.1016/S0304-4076(98)00033-5.

Tables and Figures

Table 1 Summary of Data Sources

	Data	Variables	Coverage	Source
Key data source				
(i)	Server software usage panel of U.S. organizations	Server software name, version number, IP address for the homepages of 200,000+ U.S. organizations with 50 or more employees	Jan 1, 2000--August 1, 2018; irregular capture frequency; 17+ million observations	IA
Apache version characteristics				
(ii)	Apache security vulnerabilities	Each vulnerability’s reporting date, version(s) affected, disclosure date,	158 vulnerabilities reported before or on August 1, 2018	ASF

		fix date, and version, and (when available) the reporting entity		
(iii)	Severity of Apache security vulnerabilities	Each vulnerability's severity rating (high, medium, or low), and breakdown scores including impact and exploitability	28 vulnerabilities rated high, 123 rated medium, and 7 rated low	NIST
(iv)	Apache version release dates	Each Apache version's release date	115 releases between 2000 and 2018	Authors' compilation
(v)	Apache version feature improvements	Each Apache version's new and improved features from Apache change logs		ASF
Organization characteristics				
(vi)	Basic organization characteristics	State, industry (NAICS), the estimated number of employees and estimated revenue	Cross-sectional snapshot on August 28, 2018	Mint Global by Bureau van Dijk
(vii)	Public firm characteristics	Total assets, capital expenditure, depreciation and amortization, number of employees, cash flow, and net income	Yearly panel between 2000 and 2018	Compustat
(viii)	Organization's IT operations	Number of personal computers, number of IT staff, IT budget, and software budget	Cross-sectional snapshot for the year 2017	Harte Hanks
(iv)	Organization's IT outsourcing	Whether the organization outsourced its IT operations during 2005-09	IT outsourcing variable during 2005-09	Harte Hanks
(x)	Data breach disclosures	Disclosing organization, disclosure date, state, disclosing organization industry, and (when available) number of affected records	2,366 data breaches disclosed 2005-18	Privacy Rights Clearing house's Data Breach Chronology
Website Characteristics				
(xi)	Website traffic	Alexa's ranking of top one million websites by traffic	Yearly between 2010 and 2018	Alexa
(xii)	Cloud-hosting of website	Whether an organization's server software used an IP address associated with AWS, Microsoft Azure, or Google Cloud	IP addresses associated with AWS (1/25/20), Azure (8/13/23), and Google (8/14/23).	Amazon, Microsoft, and Google
(xiii)	Website's technology use	Website's technology use in 47 technology categories, including	Data captured between 2016 and 2018; a website used a particular	HTTP Archive

	analytics, e-commerce, and web frameworks	technology if we observe the usage during 2016—18

Table 2 Summary Statistics of the Analysis Sample for Prevalence

	Count	Mean	STD	Min	25%	50%	75%	Max
$severeBugReported_{it}$	4861871	0.679	0.467	0	0	1	1	1
$severeBugDisclosed_{it}$	4861871	0.599	0.490	0	0	1	1	1
$severeBugFixed_{it}$	4861871	0.571	0.495	0	0	1	1	1
$numSevereBugReported_{it}$	4861871	2.362	2.314	0	0	2	4	12
$numSevereBugDisclosed_{it}$	4861871	2.177	2.366	0	0	1	4	12
$numSevereBugFixed_{it}$	4861871	2.012	2.312	0	0	1	4	12

Notes: The i subscript indexes organizations. The t subscript indexes months. $severeBugReported_{it}$, $severeBugDisclosed_{it}$, $severeBugFixed_{it}$ are binary indicator variables, each indicating whether there are severe security vulnerabilities that have been reported, disclosed, or fixed for the Apache version observed for the organization-month. $numSevereBugReported_{it}$, $numSevereBugDisclosed_{it}$, $numSevereBugFixed_{it}$ are count variables, each counting the number of severe security vulnerabilities reported, disclosed, or fixed for the Apache version observed for the organization-month. Appendix Table A1 shows the correlation between the variables.

Table 3 Summary Statistics of the Analysis Sample for Determinants of Vulnerabilities

	Count	Mean	STD	Min	50%	Max
$severeBugFixed_{it}$	4861871	0.571	0.495	0	1	1
$numSevereBugFixed_{it}$	4861871	2.012	2.312	0	1	12
$newUser_{it}$	4861871	0.041	0.198	0	0	1
PCs_i	3772082	226.842	1347.049	0	56	94659
$ITBudget_i$	3772082	4.481	114.361	0	0.342	17320.639
$softwareBudget_i$	3772082	0.957	28.86	0	0.06	4476.575
$techCategories_i$	845817	11.708	4.99	1	11	38
$techs_i$	845817	18.853	14.285	1	16	174
$outsourced_i$	1014195	0.192	0.394	0	0	1
$outsourcedMissing_i$	4861871	0.791	0.406	0	1	1
$cloud_{it}$	4861871	0.020	0.14	0	0	1
$highTraffic_i$	4861871	0.083	0.276	0	0	1
$monetization_i$	845817	0.905	0.293	0	1	1
$finance_i$	4606217	0.032	0.177	0	0	1
$healthcare_i$	4606217	0.095	0.293	0	0	1
$govt_i$	4606217	0.016	0.125	0	0	1
$countBreachState_{it}$	4861871	0.449	1.299	0	0	18
$noAffectedBreachState_{it}$	4861871	2101.489	51360.515	0	0	3000814.02
$countBreachIndustry_{it}$	4861871	2.04	3.883	0	0	31
$noAffectedBreachIndustry_{it}$	4861871	10051.199	108648.505	0	0	3000001.61
$employmentMint_i$	4722177	0.953	16.665	0.05	0.175	2458.775

$revenueMint_i$	4722177	0.185	2.617	-0.077	0.018	500.362
$isPublic_{iy}$	4861871	0.022	0.145	0	0	1
$capxCompustat_{iy}$	104601	0.154	0.88	-0.001	0.008	40.145
$employmentCompustat_{iy}$	104601	8.948	44.567	0	0.73	2200
$totalAssetsCompustat_{iy}$	104601	7125.214	66296.384	0	404.427	2209974
$depreciationCompustat_{iy}$	104601	127.295	695.069	0	8.843	27595
$incomeCompustat_{iy}$	104601	154.235	1156.658	-21244	5.369	41733
$cashflowCompustat_{iy}$	101723	-0.023	0.663	-63.667	0.055	5.2
$state_i$	4710191					
$naics_i$	4606217					

Notes: All dollar amounts are in millions. All headcounts are in thousands of people. The y subscript indexes years. For public firms, we observe their characteristics at the yearly level. Appendix Table A2 shows the correlation between the variables. The rest of Table 2's notes apply.

Table 4: Regression Results for the Determinants of Vulnerabilities Across Organizations

	(1)	(2)	(3)	(4)	(5)	(6)
Outcome	$severeBugFixed_{it}$			$numSevereBugFixed_{it}$		
Model	LPM	LPM	LPM	Poisson	Poisson	Poisson
$newUser_{it}$	-0.047***	-0.047***	-0.033***	-0.194***	-0.158***	-0.124***
	(0.005)	(0.008)	(0.003)	(0.015)	(0.024)	(0.007)
log $(PCs_i + 1)$	0.014***	0.011***		0.027***	0.018***	
	(0.002)	(0.002)		(0.003)	(0.003)	
log $(softwareBudget_i + 1)$	0.009	0.000		0.019*	0.009	
	(0.005)	(0.004)		(0.011)	(0.014)	
$techCategories_i$		-0.002**			-0.003	
		(0.001)			(0.004)	
log $(techs_i + 1)$		0.009			-0.017	
		(0.010)			(0.035)	
$cloud_{it}$	-0.109***	-0.199***	-0.148***	-0.414***	-0.524***	-0.532***
	(0.013)	(0.017)	(0.011)	(0.048)	(0.045)	(0.049)
$outsourced_i = 1$ &		-0.001			0.002	
$outsourcedMissing_i = 0$		(0.011)			(0.059)	
$outsourced_i = 0$ &		0.016***			0.046***	
$outsourcedMissing_i = 0$		(0.005)			(0.015)	
$highTraffic_i$	0.050***	0.026***		0.048***	0.036**	
	(0.005)	(0.005)		(0.014)	(0.018)	
$monetization_i$		0.019*			0.047	
		(0.010)			(0.034)	
$finance_i$	0.029***	0.023***		0.050***	0.038**	
	(0.004)	(0.004)		(0.007)	(0.016)	
$healthcare_i$	-0.021***	-0.009*		-0.031***	-0.036*	
	(0.004)	(0.005)		(0.009)	(0.020)	
$govt_i$	-0.005	0.008		0.013	0.060**	
	(0.007)	(0.009)		(0.016)	(0.029)	

$countbBeachState_{it}$	-0.001**	0.002**	0.001***	0.001	0.005	0.006***
	(0.001)	(0.001)	(0.000)	(0.002)	(0.003)	(0.001)
$countBreachIndustry_{it}$	-0.002*	-0.001	-0.001***	-0.005**	-0.004	-0.003***
	(0.001)	(0.001)	(0.000)	(0.002)	(0.003)	(0.001)
log $(employmentMint_i + 1)$	0.022***	0.014***		0.047***	0.045***	
	(0.003)	(0.004)		(0.007)	(0.010)	
$revenueMint_i$	-0.001***	-0.000		-0.001	0.000	
	(0.000)	(0.000)		(0.001)	(0.001)	
$isPublic_{iy}$	0.007	0.007		-0.003	0.008	
	(0.007)	(0.008)		(0.013)	(0.021)	
Constant	0.513***	0.548***	0.577***	0.763***	0.871***	1.149***
	(0.007)	(0.016)	(0.001)	(0.017)	(0.028)	(0.002)
Month Fixed Effects	Y	Y	Y	Y	Y	Y
Organization Fixed Effects			Y			Y
Observations	3,669,972	763,648	4,854,454	3,669,972	763,552	4,667,073
R-squared	0.266	0.281	0.547			

Notes: We perform a log transformation for large nonnegative variables, such as employment by adding one to the variable and then taking the logarithm. Standard errors are clustered at the state, industry, and month level for Columns (1), (2), (4), and (5). Standard errors are clustered at the organization and month level for Columns (3) and (6). Standard errors are shown in parentheses. *** p<0.01, ** p<0.05, * p<0.1. For regression results specific to public firms, please see Appendix Table A4.

Table 5 Summary Statistics of the Analysis Sample for Responsiveness

	Count	Mean	STD	Min	50%	Max
$timeToFix_{ir}$	161649	20.451	23.349	0	12	197
$fixed_{ir}$	161649	0.57	0.495	0	1	1
$severeFixed_{ir}$	161649	1.475	0.805	1	1	4
$nonSevereFixed_{ir}$	161649	0.735	1.355	0	0	9
$featureChanges_{ir}$	161649	164.111	167.43	0	92	1110
$sameMajorVersion_{ir}$	161649	0.716	0.451	0	1	1
$highImpact_{ir}$	161649	0.446	0.497	0	0	1
$highExploitability_{ir}$	161649	0.549	0.498	0	1	1
$notOSSpecific_{ir}$	161649	0.9	0.3	0	1	1
$prevTimeToFix_{ir}$	92543	24.127	22.389	1	17	197
$newUser_{it}$	161649	0.032	0.175	0	0	1
PCs_i	125847	219.124	1242.448	0	56	94659
$ITBudget_i$	125847	3.912	50.931	0	0.343	6660.01
$softwareBudget_i$	125847	0.822	12.067	0	0.061	1419.834
$techCategories_i$	27886	11.066	4.661	1	11	36
$techs_i$	27886	17.251	11.62	1	15	157
$outsourced_i$	34713	0.19	0.392	0	0	1

$outsourcedMissing_i$	161649	0.785	0.411	0	1	1
$cloud_{it}$	161649	0.012	0.111	0	0	1
$highTraffic_i$	161649	0.078	0.268	0	0	1
$monetization_i$	27886	0.898	0.303	0	1	1
$finance_i$	153342	0.037	0.188	0	0	1
$healthcare_i$	153342	0.092	0.29	0	0	1
$govt_i$	153342	0.018	0.132	0	0	1
$countBreachState_{it}$	161649	0.322	1.016	0	0	7
$noAffectedBreachState_{it}$	161649	4.485	36.924	0	0	573
$countBreachIndustry_{it}$	161649	1.521	3.559	0	0	14
$noAffectedBreachIndustry_{it}$	161649	31.16	106.217	0	0	636.274
$employmentMint_i$	157211	0.999	19.182	0.05	0.175	2458.775
$revenueMint_i$	157211	0.202	2.358	-0.077	0.018	158.869
$isPublic_{iy}$	161649	0.022	0.147	0	0	1
$capxCompustat_{iy}$	3590	0.144	0.719	0	0.007	18.237
$employmentCompustat_{iy}$	3590	8.864	31.568	0	0.8	428
$totalAssetsCompustat_{iy}$	3590	9283.29	86803.4	0	407.677	2209174
$depreciationCompustat_{iy}$	3590	129.323	595.803	0	7.992	11974
$incomeCompustat_{iy}$	3590	171.174	988.796	-2186.659	5.337	22017
$cashflowCompustat_{iy}$	3494	-0.002	0.367	-7.969	0.056	1.995
$state_i$	156872					
$naics_i$	153342					

Notes: The r subscript indexes fix releases as depicted in Figure 2. $timeToFix_{ir}$ counts the number of months it takes organization i to update their vulnerable Apache version to the fixed version r after its release. $fixed_{ir}$ is an indicator variable equal to 1 if the organization has updated to the fixed version, and 0 if the observation is right-censored. Appendix Table A3 shows the correlation between the variables. The rest of notes of Tables 1 and 2 apply.

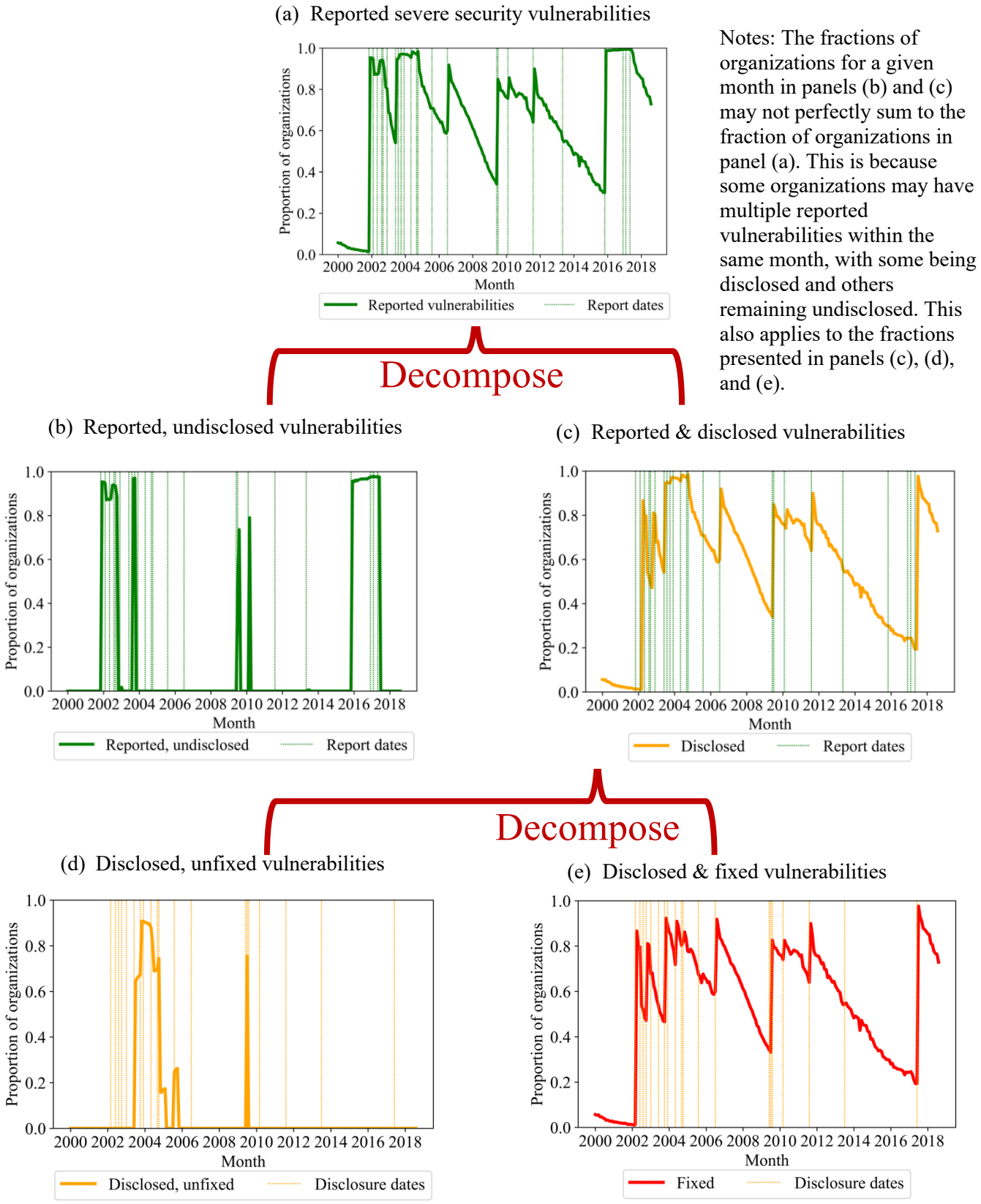
Table 6: Cox Proportional Hazards Regression Results for Responsiveness

	(1)	(2)	(3)	(4)	(5)
	Cox	Cox	Cox	Stratified	Stratified
$severeFixed_{ir}$	-0.070***	0.000	0.150***	0.073***	0.170***
	(0.011)	(0.013)	(0.028)	(0.017)	(0.019)
$nonSevereFixed_{ir}$	-0.017***	-0.034***	-0.053***	-0.066***	-0.098***
	(0.004)	(0.005)	(0.009)	(0.006)	(0.007)
$log(featureChanges_{ir} + 1)$	-0.208***	-0.195***	-0.203***	-0.094***	-0.075***
	(0.004)	(0.005)	(0.011)	(0.007)	(0.007)
$sameMajorVersion_{ir}$	1.043***	1.055***	0.700***	1.150***	1.184***
	(0.017)	(0.020)	(0.038)	(0.021)	(0.022)
$highImpact_{ir}$	0.075***	0.045***	0.100***	0.074***	0.014
	(0.012)	(0.014)	(0.025)	(0.014)	(0.015)
$highExploitability_{ir}$	0.207***	0.179***	0.188***	0.274***	0.233***
	(0.013)	(0.014)	(0.027)	(0.015)	(0.015)

$notOSSpecific_{ir}$	0.152***	0.047	0.045	0.037	-0.062*
	(0.027)	(0.031)	(0.050)	(0.032)	(0.032)
$prevTimeToFix_{ir}$	-0.020***	-0.020***	-0.018***	-0.003***	-0.003***
	(0.000)	(0.000)	(0.001)	(0.000)	(0.000)
$newUser_{it}$		-0.287***	-0.299*		-0.320**
		(0.108)	(0.179)		(0.141)
log $(PCs_i + 1)$		-0.020***	-0.013		
		(0.006)	(0.012)		
log $(softwareBudget_i + 1)$		0.000	-0.028		
		(0.019)	(0.026)		
$techCategories_i$			0.015**		
			(0.007)		
log $(techs_i + 1)$			-0.049		
			(0.063)		
$cloud_{it}$		0.129*	0.139		-0.410**
		(0.078)	(0.142)		(0.168)
$outsourced_i = 1$ &		0.037	0.148*		
$outsourcedMissing_i = 0$		(0.030)	(0.076)		
$outsourced_i = 0$ &		-0.025	-0.039		
$outsourcedMissing_i = 0$		(0.016)	(0.032)		
$highTraffic_i$		-0.058**	-0.048		
		(0.023)	(0.032)		
$monetization_i$			0.030		
			(0.056)		
$finance_i$		-0.064*	-0.020		
		(0.035)	(0.066)		
$healthcare_i$		0.058**	-0.106		
		(0.023)	(0.068)		
$govt_i$		-0.013	-0.034		
		(0.052)	(0.102)		
$countbBeachState_{it}$		-0.052***	-0.064***		-0.044***
		(0.009)	(0.017)		(0.014)
$countBreachIndustry_{it}$		-0.034***	-0.036***		-0.052***
		(0.004)	(0.008)		(0.006)
log $(employmentMint_i + 1)$		-0.048***	-0.019		
		(0.016)	(0.022)		
$revenueMint_i$		-0.002	0.001		
		(0.005)	(0.005)		
$isPublic_{iy}$		0.017	0.013		0.261
		(0.043)	(0.063)		(0.205)
Observations	92543	71343	16883	92543	92543

Notes: Stratification and standard error clustering are at the organization level. Clustering at the month level is not performed due to the perfect mapping between releases and the months they occurred. Public firm characteristics are not included in the regressions to avoid a significant reduction in the number of observations. The rest of the notes of Table 4 apply.

Figure 1 Proportion of Organizations with Reported, Disclosed, and Fixed Severe Security Vulnerabilities



Notes: The fractions of organizations for a given month in panels (b) and (c) may not perfectly sum to the fraction of organizations in panel (a). This is because some organizations may have multiple reported vulnerabilities within the same month, with some being disclosed and others remaining undisclosed. This also applies to the fractions presented in panels (c), (d), and (e).

Figure 2 Proportion of Organizations Operating with Multiple Fixed Severe Security Vulnerabilities

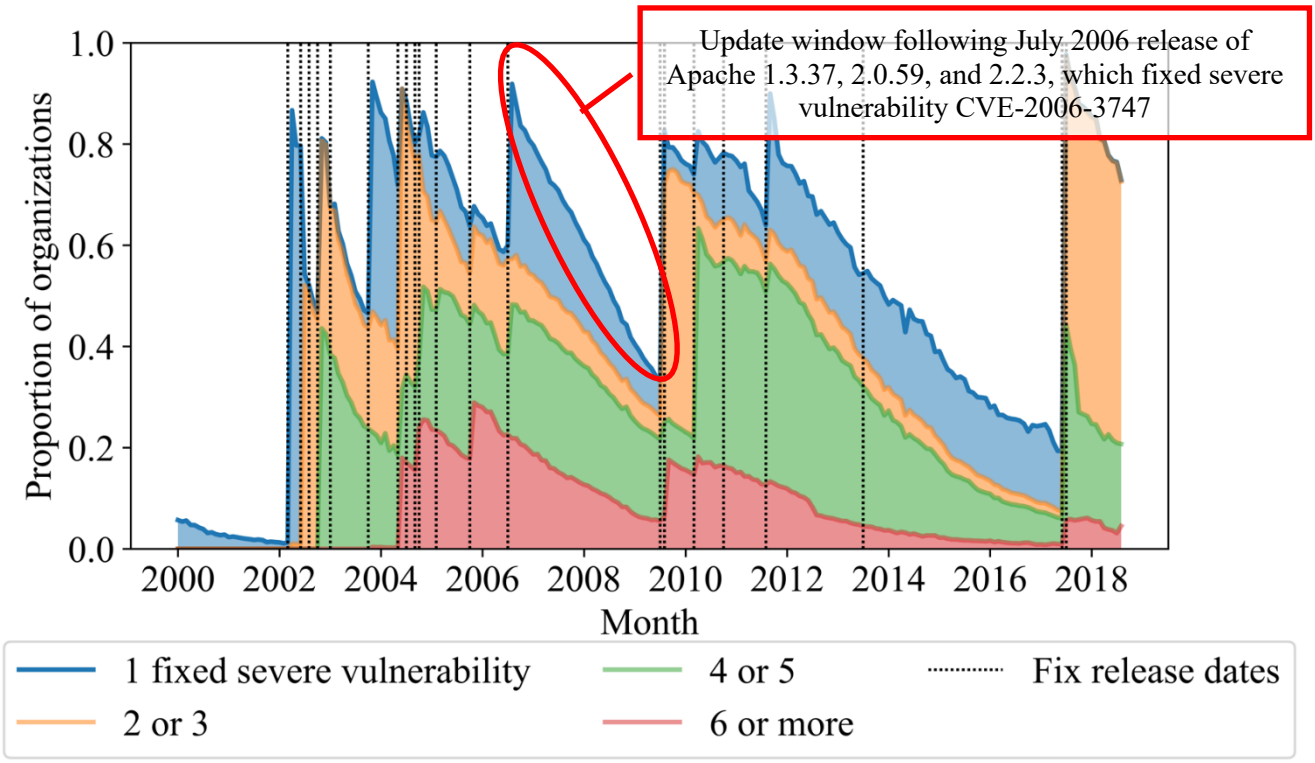


Figure 3 Construction of the Analysis Sample for Responsiveness

(a) Analysis sample for determinants of vulnerabilities, restricted to Adobe

Organization i	Month t	Apache Version	…
adobe.com	03/2001	1.3.14	…
adobe.com	04/2001	1.3.19	…
adobe.com	05/2001	1.3.19	…
adobe.com	06/2001	1.3.19	…
adobe.com	07/2001	1.3.19	…
adobe.com	08/2001	1.3.19	…
adobe.com	09/2001	1.3.19	…
adobe.com	10/2001	1.3.19	…
adobe.com	11/2001	1.3.19	…
adobe.com	12/2001	1.3.19	
adobe.com	01/2002	1.3.19	
adobe.com	03/2002	1.3.19	
adobe.com	05/2002	1.3.23	
adobe.com	06/2002	1.3.23	
adobe.com	07/2002	1.3.26	
adobe.com	08/2002	1.3.26	
adobe.com	09/2002	1.3.26	
adobe.com	10/2002	1.3.26	
adobe.com	11/2002	1.3.26	
adobe.com	12/2002	1.3.26	

(b) Analysis sample for responsiveness

Organization i	Release r	$timeToFix_{ir}$	$fixed_{ir}$	$severeFixed_{ir}$	…
adobe.com	1.3.24, 03/2002	4	1	1	…
adobe.com	1.3.26, 10/2002	2	0	2	…

Apache 1.3.24 released in March 2002, which fixed severe vulnerability CVE-2002-0061 in Apache 1.3.19

Firm used version above 1.3.24 by July 2002

Apache 1.3.27 released in October 2002, which fixed severe vulnerabilities CVE-2002-0839 and CVE-2002-0843 in Apache 1.3.26

Firm used 1.3.26 until the final recorded observation of Apache usage; therefore this update cycle is right-censored

Appendices

A.1. Correlation Tables

Table A1: Correlation Table of the Analysis Sample for Prevalence

	(1)	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**
(1)$severeBugReported_{it}$	1	0.84	0.79	0.7	0.63	0.6
(2)$severeBugDisclosed_{it}$	0.84	1	0.94	0.74	0.75	0.71
(3)$severeBugFixed_{it}$	0.79	0.94	1	0.74	0.76	0.75
(4)$numSevereBugReported_{it}$	0.7	0.74	0.74	1	0.98	0.95
(5)$numSevereBugDisclosed_{it}$	0.63	0.75	0.76	0.98	1	0.97
(6)$numSevereBugFixed_{it}$	0.6	0.71	0.75	0.95	0.97	1

Table A2: Correlation Table of the Analysis Sample for Determinants of Vulnerabilities

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)	(21)	(22)	(23)
(1)$severeBugFixed_{it}$	1	0.75	-0.07	0.02	0.01	0.01	0.02	0.02	-0.04	-0.01	-0.05	0.05	0.02	0.01	-0.01	0.01	-0.04	-0.02	-0.09	-0.04	0.01	0.01	0.02
(2)$numSevereBugFixed_{it}$	0.75	1	-0.07	0.02	0.01	0.01	0.01	0.01	-0.03	-0.01	-0.05	0.04	0.01	0.01	-0.01	0.01	-0.02	-0.02	-0.06	-0.04	0.02	0.02	0.02
(3)$newUser_{it}$	-0.07	-0.07	1	0.01	0	0	0.02	0.04	0.01	0.01	0.01	0	0	0.01	0	0	-0.02	0	-0.04	-0.01	0	0.01	0
(4)PCs_i	0.02	0.02	0.01	1	0.15	0.11	0.21	0.32	-0.06	-0.03	0	0.21	0.04	0	0.02	0.03	0	0	-0.01	0	0.11	0.12	0.07
(5)$ITBudget_i$	0.01	0.01	0	0.15	1	0.99	0.05	0.09	-0.04	0	0	0.08	0.02	0.01	-0.01	0	0	0	0	0	0.17	0.4	0.1
(6)$softwareBudget_i$	0.01	0.01	0	0.11	0.99	1	0.03	0.07	-0.03	0	0	0.07	0.01	0.02	-0.01	0	0	0	0	0	0.13	0.34	0.09
(7)$techCategories_i$	0.02	0.01	0.02	0.21	0.05	0.03	1	0.85	-0.04	-0.03	0.03	0.34	0.43	-0.07	0.03	0	0.03	0.01	0.02	0	0.09	0.07	0.05
(8)$techs_i$	0.02	0.01	0.04	0.32	0.09	0.07	0.85	1	-0.04	-0.04	0.01	0.38	0.28	-0.07	0.03	0.02	0.01	0	-0.01	-0.01	0.13	0.11	0.06
(9)$outsourced_i$	-0.04	-0.03	0.01	-0.06	-0.04	-0.03	-0.04	-0.04	1		-0.02	-0.09	-0.02	-0.02	0.03	-0.01	0	0	0.01	0	-0.01	-0.03	-0.03
(10)$outsourcedMissing_i$	-0.01	-0.01	0.01	-0.03	0	0	-0.03	-0.04		1	0.01	-0.05	-0.01	-0.02	-0.03	0	0.01	0	-0.01	0	-0.01	0	-0.01
(11)$cloud_{it}$	-0.05	-0.05	0.01	0	0	0	0.03	0.01	-0.02	0.01	1	0.04	0.01	0	0	-0.01	0.11	0.02	0.13	0.04	0	0.01	0.03
(12)$highTraffic_i$	0.05	0.04	0	0.21	0.08	0.07	0.34	0.38	-0.09	-0.05	0.04	1	0.19	0.01	-0.06	0.01	0.02	0.01	-0.01	0	0.09	0.11	0.15
(13)$monetization_i$	0.02	0.01	0	0.04	0.02	0.01	0.43	0.28	-0.02	-0.01	0.01	0.19	1	0.01	0.02	-0.02	-0.01	0	-0.01	0	0.02	0.02	0.02
(14)$finance_i$	0.01	0.01	0.01	0	0.01	0.02	-0.07	-0.07	-0.02	-0.02	0	0.01	0.01	1	-0.06	-0.02	0	0	-0.06	-0.01	0	0.03	0.05
(15)$healthcare_i$	-0.01	-0.01	0	0.02	-0.01	-0.01	0.03	0.03	0.03	-0.03	0	-0.06	0.02	-0.06	1	-0.04	0	0	0.14	-0.03	0	-0.01	-0.04
(16)$govt_i$	0.01	0.01	0	0.03	0	0	0	0.02	-0.01	0	-0.01	0.01	-0.02	-0.02	-0.04	1	-0.01	0	-0.07	-0.01	0	-0.01	-0.02
(17)$countBreachState_{it}$	-0.04	-0.02	-0.02	0	0	0	0.03	0.01	0	0.01	0.11	0.02	-0.01	0	0	-0.01	1	0.2	0.31	0.08	0	0	0.02
(18)$noAffectedBreachState_{it}$	-0.02	-0.02	0	0	0	0	0.01	0	0	0	0.02	0.01	0	0	0	0	0.2	1	0.06	0.27	0	0	0
(19)$countBreachIndustry_{it}$	-0.09	-0.06	-0.04	-0.01	0	0	0.02	-0.01	0.01	-0.01	0.13	-0.01	-0.01	-0.06	0.14	-0.07	0.31	0.06	1	0.17	0	0	0.01
(20)$noAffectedBreachIndustry_{it}$	-0.04	-0.04	-0.01	0	0	0	0	-0.01	0	0	0.04	0	0	-0.01	-0.03	-0.01	0.08	0.27	0.17	1	0	0	0.01
(21)$employmentMint_i$	0.01	0.02	0	0.11	0.17	0.13	0.09	0.13	-0.01	-0.01	0	0.09	0.02	0	0	0	0	0	0	0	1	0.54	0.12
(22)$revenueMint_i$	0.01	0.02	0.01	0.12	0.4	0.34	0.07	0.11	-0.03	0	0.01	0.11	0.02	0.03	-0.01	-0.01	0	0	0	0	0.54	1	0.2
(23)$isPublic_{iy}$	0.02	0.02	0	0.07	0.1	0.09	0.05	0.06	-0.03	-0.01	0.03	0.15	0.02	0.05	-0.04	-0.02	0.02	0	0.01	0.01	0.12	0.2	1

Notes: We dropped public firm characteristics from Compustat to reduce table size.

Table A3: Correlation Table of the Analysis Sample for Responsiveness

	(1)	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**	**(7)**	**(8)**	**(9)**	**(10)**
(1)$timeToFix_{ir}$	1	-0.02	-0.23	-0.05	0.18	-0.28	0.16	-0.01	0.05	0.51
(2)$fixed_{ir}$	-0.02	1	-0.24	0.04	-0.27	0.23	0.08	0.18	-0.09	-0.02
(3)$severeFixed_{ir}$	-0.23	-0.24	1	0.32	0.02	0.15	-0.36	-0.22	0.2	-0.04
(4)$nonSevereFixed_{ir}$	-0.05	0.04	0.32	1	-0.11	0.32	-0.14	0.28	0.02	0.04
(5)$featureChanges_{ir}$	0.18	-0.27	0.02	-0.11	1	-0.54	0.04	-0.15	0.01	0.29
(6)$sameMajorVersion_{ir}$	-0.28	0.23	0.15	0.32	-0.54	1	-0.16	0.27	-0.13	-0.06
(7)$highImpact_{ir}$	0.16	0.08	-0.36	-0.14	0.04	-0.16	1	-0.39	0.04	0.07
(8)$highExploitability_{ir}$	-0.01	0.18	-0.22	0.28	-0.15	0.27	-0.39	1	-0.3	0
(9)$notOSSpecific_{ir}$	0.05	-0.09	0.2	0.02	0.01	-0.13	0.04	-0.3	1	-0.04
(10)$prevTimeToFix_{ir}$	0.51	-0.02	-0.04	0.04	0.29	-0.06	0.07	0	-0.04	1

Notes: To reduce the table size, we have kept only the outcome variable, the event variable, and the characteristics of the updates.

A.2. Determinants of Vulnerabilities Analysis for Public Firms

Table A4: Regression Results for the Determinants of Vulnerabilities Across Public Firms

	(1)	(2)	(3)	(4)
Outcome	$severeBugFixed_{it}$		$numbSevereBugFixed_{it}$	
Model	LPM	LPM	Poisson	Poisson
$newUser_{it}$	-0.068***	-0.038***	-0.229***	-0.102***
	(0.016)	(0.009)	(0.054)	(0.025)
log $(PCs_i + 1)$	0.009		0.020	
	(0.007)		(0.019)	
log $(softwareBudget_i + 1)$	-0.001		0.001	
	(0.006)		(0.021)	
$techCategories_i$	0.005		0.019**	
	(0.004)		(0.008)	
log $(techs_i + 1)$	-0.044		-0.165***	
	(0.031)		(0.060)	
$cloud_{it}$	-0.179***	-0.165***	-0.429***	-0.525***
	(0.037)	(0.037)	(0.085)	(0.094)
$outsourced_i = 1$ &	0.057		0.235	
$outsourcedMissing_i = 0$	(0.035)		(0.178)	
$outsourced_i = 0$ &	0.000		0.008	
$outsourcedMissing_i = 0$	(0.017)		(0.064)	
$highTraffic_i$	0.031		0.001	
	(0.019)		(0.052)	
$monetization_i$	0.060*		0.098	
	(0.030)		(0.098)	
$finance_i$	-0.008		-0.075	
	(0.023)		(0.058)	
$healthcare_i$	-0.004		0.081	
	(0.058)		(0.131)	
$countbBeachState_{it}$	0.001	-0.003	-0.001	-0.004
	(0.004)	(0.002)	(0.011)	(0.005)
$countBreachIndustry_{it}$	-0.000	0.001	0.005	0.009*
	(0.003)	(0.002)	(0.006)	(0.005)
log $(employmentMint_i + 1)$	0.005		0.036*	
	(0.008)		(0.021)	
r$evenueMint_i$	-0.000		-0.001	
	(0.000)		(0.001)	
$capxCompustat_{iy}$	-0.004	-0.010	0.029	0.032*
	(0.009)	(0.014)	(0.021)	(0.019)
log $(employmentCompu_{iy} + 1)$	0.012	-0.003	0.032	0.008
	(0.010)	(0.015)	(0.033)	(0.035)

log $(totalAssetsCompu_{iy} + 1)$	0.010	0.001	0.049**	0.016
	(0.007)	(0.013)	(0.020)	(0.033)
log $(depreciationCompu_{iy} + 1)$	-0.006	-0.010	-0.049***	0.011
	(0.009)	(0.012)	(0.017)	(0.030)
$incomeCompustat_{iy}$	0.000	0.000	-0.000	0.000
	(0.000)	(0.000)	(0.000)	(0.000)
$cashflowCompustat_{iy}$	0.017	-0.000	0.097	0.001
	(0.025)	(0.007)	(0.067)	(0.035)
Constant	0.541***	0.661***	0.811***	1.072***
	(0.057)	(0.065)	(0.134)	(0.181)
Month Fixed Effects	Y	Y	Y	Y
Organization Fixed Effects		Y		Y
Observations	56,077	101,555	55,837	96,266
R-squared	0.339	0.593		

Notes: Standard errors are clustered at the state, industry, and month level for Columns (1) and (3), and are clustered at the organization and month level for Columns (2) and (4). The rest of the notes of Table 4 apply.

A.3. Results for 2013—2018

Table A5: Regression Results for the Determinants of Vulnerabilities Across Organizations, 2013—2018

	(1)	(2)	(3)	(4)	(5)	(6)
Outcome	$severeBugFixed_{it}$			$numSevereBugFixed_{it}$		
Model	LPM	LPM	LPM	Poisson	Poisson	Poisson
$newUser_{it}$	-0.067***	-0.055***	-0.035***	-0.204***	-0.133**	-0.108***
	(0.012)	(0.017)	(0.004)	(0.047)	(0.055)	(0.017)
log $(PCs_i + 1)$	0.020***	0.019***		0.035***	0.037***	
	(0.004)	(0.004)		(0.010)	(0.008)	
log $(softwareBudget_i + 1)$	0.019	0.002		0.016	-0.007	
	(0.013)	(0.009)		(0.029)	(0.029)	
$techCategories_i$		-0.006**			-0.015**	
		(0.002)			(0.006)	
log $(techs_i + 1)$		0.028			0.016	
		(0.020)			(0.057)	
$cloud_{it}$	-0.137***	-0.226***	-0.181***	-0.483***	-0.591***	-0.597***
	(0.013)	(0.018)	(0.015)	(0.063)	(0.061)	(0.061)
$outsourced_i = 1$ &		-0.049			-0.174**	
$outsourcedMissing_i = 0$		(0.031)			(0.081)	
$outsourced_i = 0$ &		-0.001			0.025	
$outsourcedMissing_i = 0$		(0.011)			(0.021)	
$highTraffic_i$	0.125***	0.070***		0.172***	0.135***	
	(0.009)	(0.009)		(0.031)	(0.035)	
$monetization_i$		0.036			0.075*	
		(0.021)			(0.045)	
$finance_i$	0.052***	0.020		0.032**	0.019	
	(0.008)	(0.015)		(0.015)	(0.034)	
$healthcare_i$	-0.033**	0.017		-0.046	-0.047	
	(0.013)	(0.015)		(0.030)	(0.052)	
$govt_i$	-0.001	-0.002		0.115***	0.134*	
	(0.014)	(0.020)		(0.039)	(0.069)	
$countbBeachState_{it}$	0.000	0.004***	0.000	0.005**	0.010***	0.002*
	(0.001)	(0.001)	(0.000)	(0.002)	(0.003)	(0.001)
$countBreachIndustry_{it}$	-0.002	-0.001	-0.000	-0.005*	-0.003	-0.001
	(0.001)	(0.001)	(0.000)	(0.003)	(0.004)	(0.001)
log $(employmentMint_i + 1)$	0.035***	0.022***		0.067***	0.069***	
	(0.005)	(0.008)		(0.017)	(0.023)	
$revenueMint_i$	-0.002**	-0.001		-0.004*	-0.004	
	(0.001)	(0.001)		(0.002)	(0.003)	
$isPublic_{iy}$	-0.006	-0.015		-0.038	-0.046	
	(0.013)	(0.019)		(0.025)	(0.048)	

Constant	0.364***	0.429***	0.460***	0.407***	0.528***	1.189***
	(0.012)	(0.040)	(0.001)	(0.029)	(0.076)	(0.005)
Month Fixed Effects	Y	Y	Y	Y	Y	Y
Organization Fixed Effects			Y			Y
Observations	983,461	203,694	1,263,331	983,461	203,694	993,032
R-squared	0.194	0.170	0.789			

Notes of Table 4 apply.

Table A6: Cox Proportional Hazards Regression Results for Responsiveness, 2013—2018

	(1)	(2)	(3)	(4)	(5)
	Cox	Cox	Cox	Stratified	Stratified
$severeFixed_{ir}$	1.173***	1.286***	2.217***	1.751	1.530
	(0.083)	(0.093)	(0.215)	(1.125)	(1.187)
$nonSevereFixed_{ir}$	0.412***	0.524***	0.349*	0.116	0.215
	(0.110)	(0.143)	(0.188)	(0.238)	(0.301)
$log(featureChanges_{ir} + 1)$	-0.182***	-0.164***	-0.113*	-0.209	-0.236
	(0.022)	(0.023)	(0.060)	(0.170)	(0.175)
$sameMajorVersion_{ir}$	0.013	-0.157	-0.329*	3.928***	3.515***
	(0.109)	(0.131)	(0.197)	(0.716)	(0.852)
$highImpact_{ir}$	0.681	0.308	3.511***	4.643	3.134
	(0.621)	(0.806)	(1.055)	(3.281)	(3.800)
$highExploitability_{ir}$	-0.022***	-0.023***	-0.027***	-0.050**	-0.055**
	(0.002)	(0.002)	(0.004)	(0.024)	(0.027)
$notOSSpecific_{ir}$	1.173***	1.286***	2.217***	1.751	1.530
	(0.083)	(0.093)	(0.215)	(1.125)	(1.187)
$prevTimeToFix_{ir}$	0.412***	0.524***	0.349*	0.116	0.215
	(0.110)	(0.143)	(0.188)	(0.238)	(0.301)
$newUser_{it}$		0.147	-0.380		-13.997***
		(0.357)	(0.606)		(1.252)
log $(PCs_i + 1)$		-0.040	-0.006		
		(0.025)	(0.054)		
log $(softwareBudget_i + 1)$		0.101	-0.033		
		(0.089)	(0.124)		
$techCategories_i$			0.024		
			(0.032)		
log $(techs_i + 1)$			0.093		
			(0.278)		
$cloud_{it}$		0.551***	0.452**		0.772
		(0.087)	(0.178)		(1.760)
$outsourced_i = 1$ & $outsourcedMissing_i = 0$		-0.104	-0.542		
		(0.144)	(0.348)		
$outsourced_i = 0$ & $outsourcedMissing_i = 0$		0.080	0.082		
		(0.069)	(0.148)		
$highTraffic_i$		-0.109	-0.305**		
		(0.120)	(0.139)		
$monetization_i$			0.043		
			(0.217)		
$finance_i$		0.051	0.252		
		(0.146)	(0.258)		
$healthcare_i$		0.103	-0.340		
		(0.100)	(0.342)		

$govt_i$		-0.057	-0.660*		
		(0.210)	(0.391)		
$countbBeachState_{it}$		-0.021	0.028		-0.191
		(0.017)	(0.038)		(0.143)
$countBreachIndustry_{it}$		-0.011	-0.020		-0.068
		(0.008)	(0.015)		(0.074)
log $(employmentMint_i + 1)$		-0.019	-0.092		
		(0.074)	(0.104)		
$revenueMint_i$		0.015	0.021		
		(0.015)	(0.013)		
$isPublic_{iy}$		-0.045	0.010		-17.180***
		(0.188)	(0.248)		(1.651)
Observations	9941	8041	1673	9941	9941

Notes of Table 6 apply.

A.4. Backporting

Backporting, a practice of taking security fixes from a more recent version of software and applying them to older versions, creates a measurement problem for our analysis of vulnerability prevalence. Backporting is often done to ensure that systems which for various reasons cannot upgrade to the latest version of software can still benefit from the latest security patches. The challenge for our analysis arises from the fact that backported versions of Apache, despite including fixes, often retain their original version number in server headers. This can lead to the misinterpretation that an organization is neglecting security fixes when, in reality, they are applying fixes through backports.[13] Additionally, our dataset is unable to detect when a firm uses a backported version of Apache. Despite the measurement issue, we believe that this does not significantly undermine the validity of our results for several reasons.

First, backports are not always available. The development of backported versions of Apache server software is not through the ASF but is at the discretion of the developers of a system's operating system. To the best of our knowledge, the most popular operating system for hosting websites, Windows Server, which comprises 42% of market share for website hosting as of 2024,[14] does not practice backporting of Apache server software. Among the other operating systems that are popular for hosting server software, backporting of Apache server software has not always been available and has been practiced in an uneven manner.[15] For example, to the best of our knowledge, Ubuntu, the second most popular operating system for websites which comprises 32% of the website market share as of 2024,[16] has consistently provided backports of Apache vulnerabilities starting from Ubuntu 6.06 (Dapper Drake), which was released in June 2006.[17] Debian, the third most popular operating system comprising 9% of website market share as of 2024, only began officially backporting in 2010.[18] We do not observe substantial changes in vulnerability prevalence or organizations' responsiveness before or after 2006 or before or after 2010 when Ubuntu and Debian started backporting.

Second, backports do not provide long-term security or stability. According to developers of Ubuntu, the most popular operating system for website hosting that offers backporting, "Unlike the packages released with Ubuntu, backports do not come with any security support guarantee. The Ubuntu Security Team does not update packages in backports…" and backports may interact negatively with other older software on the user's system in ways that the developers have not anticipated.[19] In addition, operating system distributors only backport a subset of known vulnerabilities to previous software versions for a limited duration. For example, Ubuntu focuses on backporting only the fixes for the most severe

[13] A similar concern about such false-positive mismeasurement has been discussed regarding "scanners," which examine the code running on an operating system and provide the owner with alerts regarding the presence of potentially insecure software.
[14] https://www.wappalyzer.com/technologies/operating-systems/
[15] This could be due to the practice being viewed negatively by some in early years. For example, in 2004, the CTO of SUSE remarked that backporting was bad for the open-source ecosystem since it prevented standardization (https://www.datamation.com/applications/linux-creator-calls-backporting-good-thing/).
[16] See https://www.wappalyzer.com/technologies/operating-systems/. In addition, OpenLogic has conducted a survey of enterprises and has also found that Ubuntu and Debian are the two most popular Linux distributions used. See https://www.openlogic.com/blog/top-enterprise-linux-distributions#:~:text=Amazon%20Linux%2C%20Rocky%20Linux%2C%20and,than%20ever%20to%20choose%20from!
[17] https://ubuntu.com/security/cves?q=&package=apache2&priority=&version=dapper&status=
[18] https://wiki.debian.org/Backports
[19] See https://help.ubuntu.com/community/UbuntuBackports. Similar warnings have been posted by Debian's developers, "Backports cannot be tested as extensively as Debian stable, and backports are provided on an as-is basis, with risk of incompatibilities with other components in Debian stable. Use with care!" (https://backports.debian.org/).

vulnerabilities, and the support lasts between five to ten years for each Ubuntu version, depending on the support tier to which a user belongs. It is fair to say that users of older software, even when diligently applying backports, should expect a continued degradation of the software's security, stability, and performance, as well as that of the system.

Third, the availability of backports does not guarantee that users will apply them or do so promptly. Backported versions of Apache are not installed automatically or by default on the popular operating systems we have studied. For example, although Ubuntu and Debian provide backported versions of Apache that address severe vulnerabilities, these are not installed by default for their users. Users must either manually install the backports or modify their system settings to allow automatic installation of these backported versions. West and Moore (2022) studied user behavior regarding the application of Ubuntu backports of patches for OpenSSH, an open-source connectivity tool for remote login with the SSH protocol, which allows observation of the status of backport application through OpenSSH banners. They found that a significant fraction of machines did not apply all available security backports. Throughout their sample period from 2015 to 2019, during times when relatively few backports were issued, the population was able to catch up, with the unpatched share falling to around 20% at one point in early 2018. However, in instances when multiple backports were issued consecutively, the servers could not keep up with applying all the patches, and the unpatched share consistently remained above 60% in the entire 2016.

Lastly, the practice of backporting does not alter the fact that users are employing outdated server software, often reflecting the obsolescence of their entire technology stack, which can present a multitude of problems. For the popular Linux distributions we have reviewed, once developers release a new version of the operating system, it is generally fixed at that point and does not allow for newer versions of the included software packages.[20] This means that updating to a newer Apache version is only possible when the entire operating system is updated. Hence, using an outdated Apache version, even if it is patched via backports, often implies the continued use of an outdated operating system and other legacy packages, further compounding the deterioration of the server software's security, stability, and performance, as well as that of the entire system.

Overall, while we believe backporting can alleviate some security concerns, it does not alter the core findings of our paper regarding firms' security practices in using server software. The overall prevalence of vulnerabilities within the user base is not likely to be fully explained by backporting.[21] Within the subset of users employing backporting, many continue to use older versions of Apache software for extended periods, which may harbor multiple vulnerabilities susceptible to attacks, and may also experience diminished security and stability over time. However, considering that our findings on the vulnerability prevalence among Apache users represent an upper bound of the actual attack surface, we have been careful in our discussion not to equate the vulnerability frequency in the base versions of Apache with the attack surface, due to the possibility that some vulnerabilities may have been addressed in the short run through backports.

[20] https://help.ubuntu.com/community/UbuntuBackports

[21] Tajalizadehkhoob et al. (2017) adopt the generous assumption that systems are always patched when users operate popular operating systems within the support period for backporting. Despite this assumption, they found that 144,637 (71%) out of the 203,455 domains, for which they had server software version information in March 2016, were unpatched. Although their study encompasses various types of server software and considers both severe and non-severe vulnerabilities within a short measurement period—differing from this paper—their findings corroborate that backporting does not fully account for the observed vulnerability prevalence.