Communications of the Association for Information Systems

Volume 43 Article 2

8-2018

Challenges to Cybersecurity: Current State of Affairs

Ravi Sen
Texas A M University, rsen@mays.tamu.edu

Communications of the Association for Information Systems

Research Paper DOI: 10.17705/1CAIS.04302 ISSN: 1529-3181

Challenges to Cybersecurity: Current State of Affairs

Ravi Sen

Department of Information and Operations Management

Mays Business School, Texas A&M University

rsen@mays.tamu.edu

Abstract:

Despite increasing investment in cybersecurity initiatives, incidents such as data breach, malware infections, and cyberattacks on cyberphysical systems show an upward trend. I identify the technical, economic, legal, and behavioral challenges that continue to obstruct any meaningful effort to achieve reasonable cybersecurity. I also summarize the recent initiatives that various stakeholders have taken to address these challenges and highlight the limitations of those initiatives.

Keywords: Cybersecurity, Software Security, Cybercrime, Cyberattack, Cyberwar, Economics of Security, Bad Code, Vulnerabilities, Exploits, Computer Security Laws.

This manuscript underwent editorial review. It was received 01/09/2018 and was with the authors for 1 month for 1 revision. Anthony Townsend served as Associate Editor.

1 Introduction

The frequency of cybersecurity incidents continues to grow. For example, according to data collected by Privacy Rights ClearingHouse (n.d.), more than 7,730 data breach incidents have been made public in the period from 2005 to October, 2017 (see Figure 1).

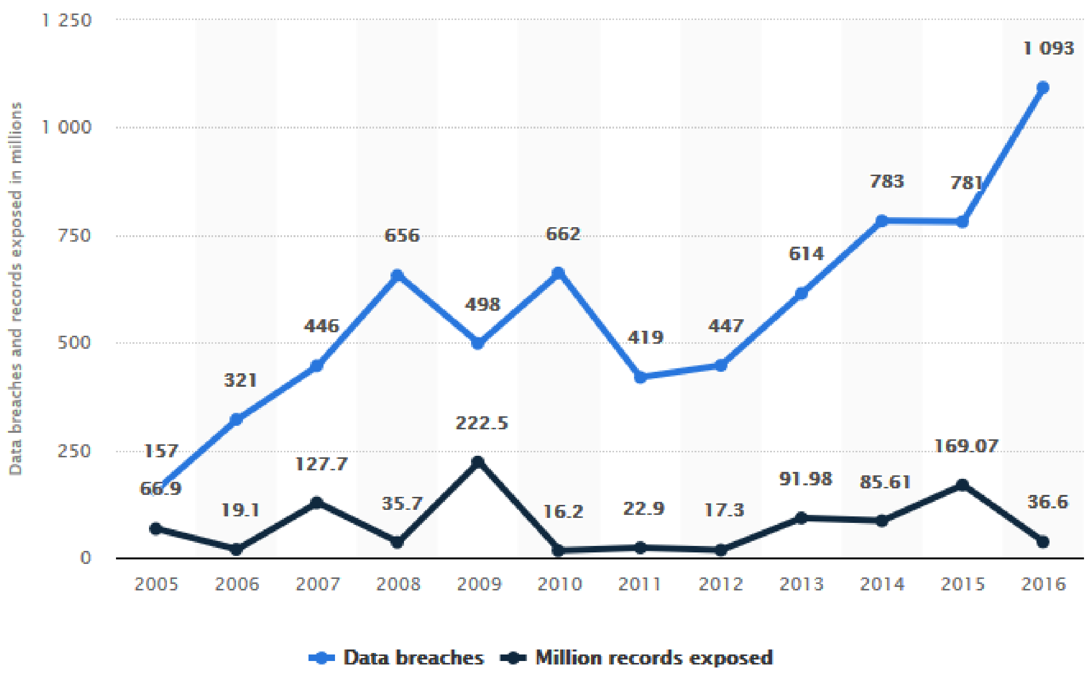

Figure 1. Annual Number of Data Breaches and Exposed Records in USA (2005-2016) (Identity Theft Resource Center, n.d.)

These data breach incidents have resulted in more than one trillion compromised individual records (Identity Theft Resource Center. (n.d.). In reality, this number should be much higher because 1) several incidents have an unknown number of breached records and 2) Privacy Rights Clearinghouse does not comprehensively compile all data breach incidents.

In addition to compromising protected data, threat agents can potentially target societally vital systems such as those that manage power grids, telecommunications networks, transportation networks, and transports (e.g., automobiles, airplanes). Such systems, often called cyberphysical systems (CPS), refer to networked systems of cyber (computation and communication) and physical (sensors and actuators) components that one uses to manage and operate critical infrastructure (Ashibani & Mahmoud 2017). CPS can offer a bigger attack surface to the threat agents (Shukla, 2016). Furthermore, CPS have increasingly begun to rely on remotely managed insecure Internet-of-things (IoT) devices, which makes these systems increasingly vulnerable (Colbert, 2017). For example, as the smart power grids begin to rely more on sensors such as synchrophasors (U.S. Department of Energy. (n.d.).) whose data are communicated over a communication network, they become vulnerable to attackers who can spoof sensor measurements. Incidents in which attackers have breached CPS include power blackouts in Brazil (Conti, 2010), the *StuxNe*t computer worm (Farwell & Rohozinski, 2011), and various other industrial security breaches (Pasqualetti, Dorfler, & Bullo, 2013).

More recently, in the beginning of 2018, NPR began reporting serious vulnerabilities in electronic chips that Intel, ARM, and AMD manufacture (Neuman 2018). Attackers could exploit these vulnerabilities to gain access to the protected memory on the chips. Going into the future, cybersecurity incidents will increasingly impact software, hardware, national infrastructure, defense, social life, and the daily operations of individuals and organizations.

In brief, cybersecurity continues to increase in importance. Indeed, the increasing budget that organizations allocate to it evidences this growing importance (see Figure 2).

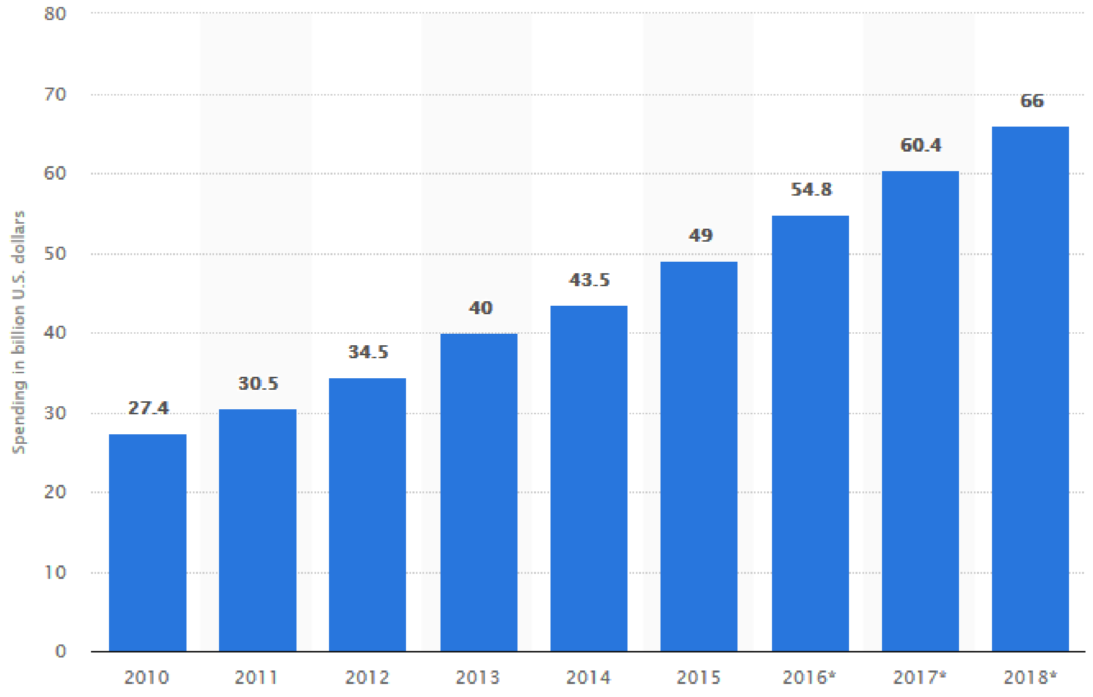

Figure 2. Spending on Cybersecurity in the USA from 2010 to 2018 (in Billion USD) (Identity Theft Resource Center, n.d.)

However, cybersecurity investments have an uncertain effectiveness. While one could argue that an increased security budget has slowed the growth rate of cybersecurity incidents, no empirical evidence supports this assertion. I propose that the state of cybersecurity is poor due to certain technical, economic, legal, and user behavioral challenges that researchers and organizations have yet to address. In this paper, I discuss some of these challenges, current initiatives to address them, and the extent to which these initiatives have succeeded in their goals.

2 Technical Challenge: Bad Code

The existence of vulnerabilities in popular software products constitutes one primary reason for poor cybersecurity (Schmidt & White, 2017). The National Institute of Standards and Technology (NIST)[1] defines vulnerability in software systems as:

> *A flaw or weakness in system security procedures, design, implementation, or internal controls that could be exercised (accidentally triggered or intentionally exploited) and result in a security breach or a violation of the system's security policy.* (Radack, 2012)

Attackers frequently exploit these vulnerabilities (Kaspersky Lab, 2017). For example, hackers stole EternalBlue, an exploit for the Server Message Block (SMB) vulnerability[2], from the US National Security Agency (NSA). They used it to launch the ransomware attacks WannaCry in May, 2017, and NotPetya in June, 2017 (Meyer, 2017). So how serious are the concerns about vulnerabilities in software? From searching the National Vulnerability Database (NVD)[3], I found a graph (see Figure 3) that shows the number of vulnerabilities disclosed per year between 2005 and 2017. The number of vulnerabilities disclosed by the end of 2017 already exceeded 14,000. Of course, all vulnerabilities do not pose the same risk. Qualys[4], a cloud security company, analyzed 40 million security scans and found that “just 10 percent of vulnerabilities are responsible for 90 percent of all cybersecurity exposures” (Chong, 2013). Still, vulnerabilities in general clearly represent a major cybersecurity concern.

[1] https://www.nist.gov/
[2] CVE Number: CVE-2017-0144
[3] https://nvd.nist.gov/vuln/search
[4] https://www.qualys.com/

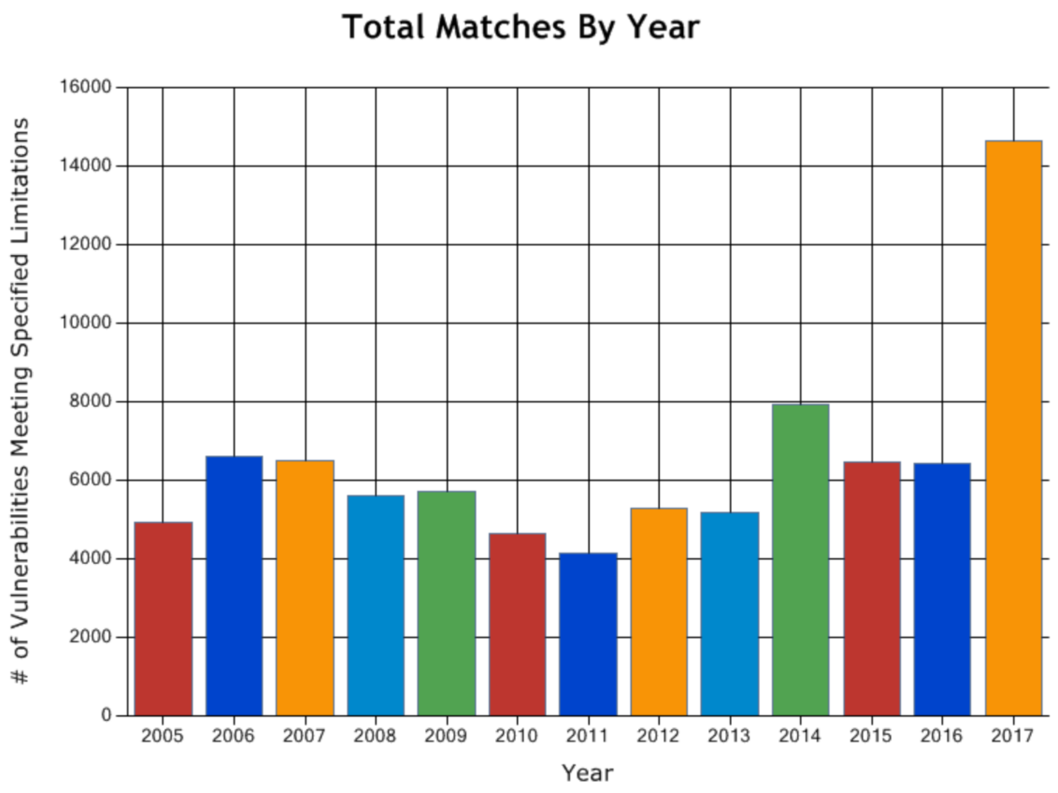

Figure 3. Number of Vulnerabilities Disclosed Per Year Since 2005 (Adapted from National Vulnerability Database)

Software producers are not unconcerned about vulnerabilities in their products; in fact, various sources have undertaken initiatives to eliminate vulnerabilities in software. For example:

- Microsoft started its Trustworthy Computing Initiative in 2002 to reduce the number of vulnerabilities at the design, coding, and documentation stages of their product development. As a result of this initiative, Microsoft created the security development lifecycle (SDL), which it introduced to software developers in 2004. Post SDL, Microsoft reported 45 percent fewer vulnerabilities a year after launching Windows Vista than in the same time after launching Windows XP, which the company developed before SDL. Similarly, SQL Server 2005 had 91 percent fewer reported vulnerabilities than the pre-SDL SQL Server 2000 (Ashford, 2012). While commendable, Microsoft's efforts do not represent the whole software industry. Unless all software producers develop and implement similar programs, we will continue to have vulnerable software.
- IEEE has published a set of guidelines to help companies establish a secure baseline for software developed for medical devices to reduce or eliminate vulnerabilities that attackers can exploit to gain access to medical devices (Haigh & Landwehr, 2015).
- The avionics industry has defined rigorous software security requirements for commercial aircraft and unmanned aerial vehicles in a certification document known as DO-178C (Howard, 2012).
- The National Institute of Standards and Technology (NIST) has developed a reference resource (see Rossman et al., 2008) to help U.S. federal government agencies in integrating essential information technology (IT) security steps into their IT system development lifecycle (SDLC).

However, in most cases, these best practices and guidelines are not mandatory.

- In 1976, Turing Award winners Edsger Dijkstra and Tony Hoare proposed a formally verified software approach to coding. This approach requires that each statement in the code should follow logically from the preceding one, which allows for one to test the whole program with the same certainty with which mathematicians prove theorems (Hartnett, 2016). In line with the above recommendation, a research group called DeepSpec[5] has begun to develop formally verified computer systems. Furthermore, tools such as TLA+[6], Coq[7] and Isabelle[8] support this approach to writing software. The promise that approach to coding shows has caused some journalists to claim that it will result in "perfect, hacker proof code" (Hartnett, 2016). However, developers who use this approach to software development face a significant challenge. Before a developer can use it, the developer must accurately and precisely 1) describe the software's correct behavior and 2) proscribe its incorrect behavior. However, doing so is difficult. For instance, Tony Hoare has acknowledged that the enormous effort required to exactly specify a software's functionality and performance under all conditions is not worth the perceived benefits (Tedre, 2015). Therefore, developers who use formal methods have moderated their goals. Instead of trying to create entire fully verified computer systems, most developers just focus on verifying smaller but especially vulnerable or critical pieces of a system (Hartnett, 2016).

- Eric Raymond, one of the most ardent proponents of open source software, claims that "given enough eyeballs, all bugs are shallow" (Raymond, 2000). The open source software development approach requires one to share the software code in the public domain, which allows other developers to analyze the code for any vulnerabilities and then fix the ones that they find. According to Raymond, an open source software development approach will more likely discover vulnerabilities compared to a closed source approach. Some evidence suggests that open source software development removes vulnerabilities at a faster pace than closed source software (Altinkemer, & Rees, & Sriidhar, 2008). However, Schryen (2011) recently found that open source and closed source software development do not significantly differ in terms of vulnerability disclosure and vendors' patching behavior. Furthermore, no convincing evidence suggests that an open source software development approach results in fewer vulnerabilities to start with. In fact, Levy (2000) asserts that: "Sure, the source code is available. But is anyone reading it?". He found no evidence that users carefully analyzed open source code to find and remove vulnerabilities.

All the above examples of corporate initiatives, development methods, and developer efforts have one thing in common: none are ambitious enough to shoot for software completely free from vulnerabilities. In fact, software developers widely believe that it is nearly impossible to write software with no vulnerabilities. Andrew Hunt (1999, p. 107) says:

> *You Can't Write Perfect Software. Did that hurt? It shouldn't. Accept it as an axiom of life. Embrace it. Celebrate it. Because perfect software doesn't exist. No one in the brief history of computing has ever written a piece of perfect software. It's unlikely that you'll be the first. And unless you accept this as a fact, you'll end up wasting time and energy chasing an impossible dream*.

They hold this belief for many reasons:

- Popular software applications are complex, and the number of lines a software application has represents one popular factor to measure that complexity (Weyuker, 1988). Carnegie Mellon University's CyLab Sustainable Computing Consortium estimates that commercial software contains 20 to 30 flaws for every 1,000 lines of code (Chong, 2013). Given that most popular software packages contain millions of lines of code (see Figure 4), and CyLab's estimate of at least 20 flaws per 1,000 lines of code (some of which could be vulnerabilities as NIST defines them), one can safely assume that all large applications contain vulnerabilities.
- One must compile software code before it can run in the production environment. Even if one assumes that a developer has written vulnerability-free code, the compiler may itself introduce vulnerabilities in the executable version of the application. For example, GCC, one of the most commonly used compilers, itself contains vulnerabilities[9].

[5] https://deepspec.org
[6] https://lamport.azurewebsites.net/tla/tla.html
[7] https://coq.inria.fr/
[8] https://isabelle.in.tum.de
[9] See https://gcc.gnu.org/bugzilla/buglist.cgi?component=c%2B%2B&product=gcc&resolution=---

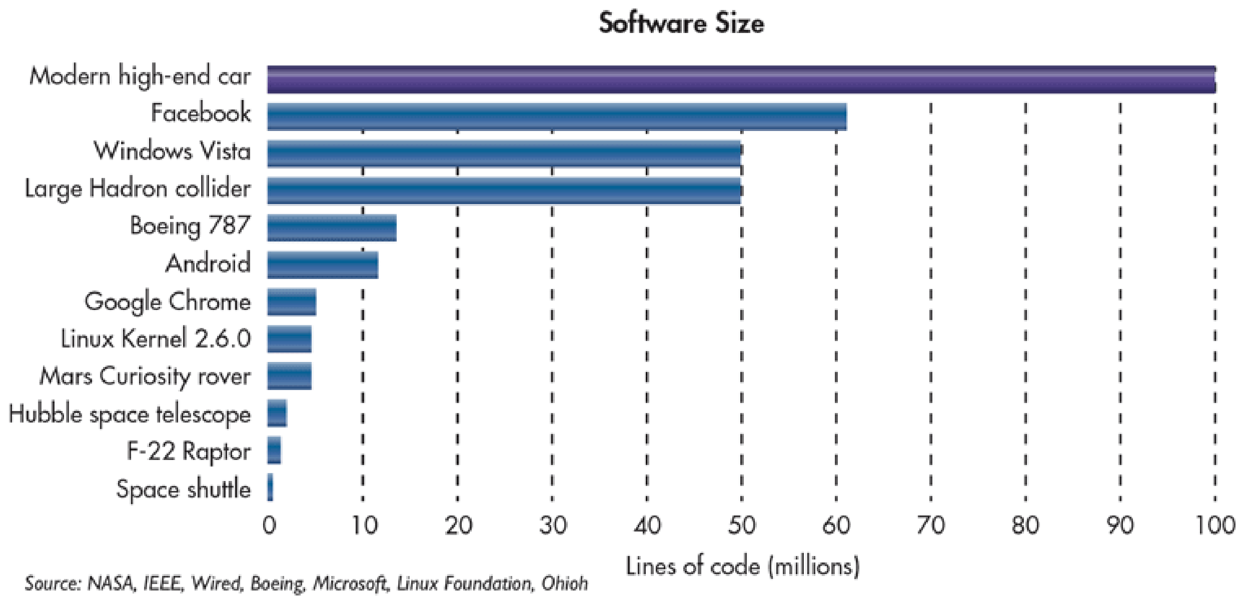

Figure 4. Millions of Lines of Code in Some Popular Software (Boldt, 2017)

- Software does not operate in isolation. Even the most basic program (written in any programming language) runs on an operating system, and, therefore, even simple programs are susceptible to vulnerabilities that may exist in the operating system.
- Software products often use pre-existing software modules (e.g., libraries, components), which may not be secure. For example, some string functions (included in the programing language C/C++ libraries) are vulnerable to buffer overflow attacks (Erickson, 2008). To write a code completely free from vulnerabilities, one would need to ensure that every library used in the code is also 100 percent secure. A similar problem plagues open source software. An average commercial application uses more than 100 open source components. From analyzing 31 billion open source components, retrieved from The Central Repository[10], Sonatype (2016) states that one sixteenth of the components included at least one vulnerability and that 69 new vulnerable components were added to the repository per quarter. Therefore, researchers have unsurprisingly found that two thirds of commercial software applications have open source components with known vulnerabilities in them (Korolov, 2017).
- To ensure that that software contains no design-related vulnerabilities, developers would need to examine every possible input to a program, the order of its commands, how data flows through the program, changes in the data as it flows through the program, and all possible execution paths that the program can take and to remove all race conditions (Weyuker, 1988). However, doing so requires skills and resources (e.g., time) that all software developers may not have. Even when they have the skills and the resources (e.g., as in the case of large software producers such as Microsoft and Apple), it might not be economically optimal for them to perform all these activities. In fact, they may be better off by releasing vulnerable software first and patching the vulnerabilities as and when they are discovered in the future (Arora, Nandkumar, & Telang, 2006).

As such, software developers typically do not focus on writing perfectly secure code but on writing code in a way that minimizes the number of vulnerabilities and removing those vulnerabilities as and when they are discovered. The most widely used approaches to minimize the risk of cybersecurity incident due to software vulnerabilities include: 1) developing and distributing patches to fix the vulnerabilities, 2) using technical security controls such as antivirus software and firewall on hosts and servers, and 3) monitoring and evaluating network traffic using intrusion-detection and prevention system (IDPS). However, these methods to address the risks associated with vulnerabilities in software products and systems have their own technical challenges to overcome:

[10] http://central.sonatype.org/

- Software patches and updates, being pieces of code, face the same development, design, and complexity challenges as any software. Therefore, they could (in theory) contain their own vulnerabilities that attackers could exploit. Furthermore, applying patches can cause “software regression” (i.e., the software stops working or performing as intended after a patch) (Nir, Tyszberowicz, & Yehudai, 2008). For example, software and firmware updates that Intel released to fix the recently discovered vulnerabilities Spectre[11] and Meltdown[12] vulnerabilities in their chips allegedly slowed down some Intel machines (Neuman, 2018). Finally, developers generally prioritize developing and distributing patches based on the public disclosure dates of vulnerabilities. This approach could leave software users vulnerable to exploits not yet publicly disclosed but that the software developer, security researchers, and the hacking community already know about (Sen & Heim, 2016).
- Organizations and individuals often use technical controls such as anti-malware and firewalls to implement the security principal “defense in depth”. This principle rests on the rationale that additional layers of security provided by controls such as firewalls and anti-malware applications make it more difficult and costly for the threat agents to breach the protected system. While this approach has successfully minimized the risks from vulnerable software, it also has its limitations. For example, anti-virus software does not work well against newer, more sophisticated viruses, worms, and trojans (Thornton 2017), and firewall rules may have anomalies that attackers can exploit to gain access to protected systems (Hu, Ann, & Kulkarni, 2012). Furthermore, users who rely on software security controls implicitly assume that these security applications are more secure than the software systems that they are trying to protect. One can test this assumption by searching for vulnerabilities discovered in two popular types of security software applications (i.e., antivirus and firewall). As such, I searched the National Vulnerability Database website and produced the graphs in Figures 5 and 6, which show the number of discovered vulnerabilities for antivirus and firewall software, respectively. As one can see, they still contain a significant number of vulnerabilities, an unsurprisingly result given that developers use the same principles to develop security software as any other type of software (whether open source or closed source). Therefore, one can safely assume that security software has the same security concerns (i.e., vulnerabilities) as any other software.

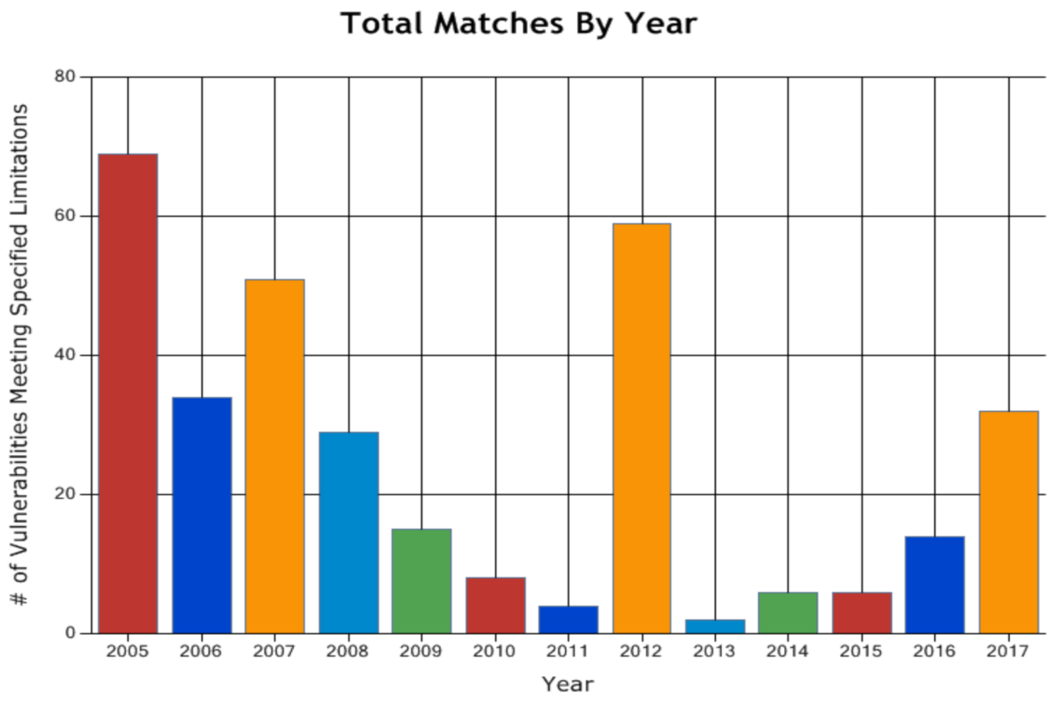

Figure 5. Number of Vulnerabilities Discovered in Antivirus Software (Adapted from National Vulnerability Database)

[11] See https://spectreattack.com/spectre.pdf
[12] See https://meltdownattack.com/meltdown.pdf

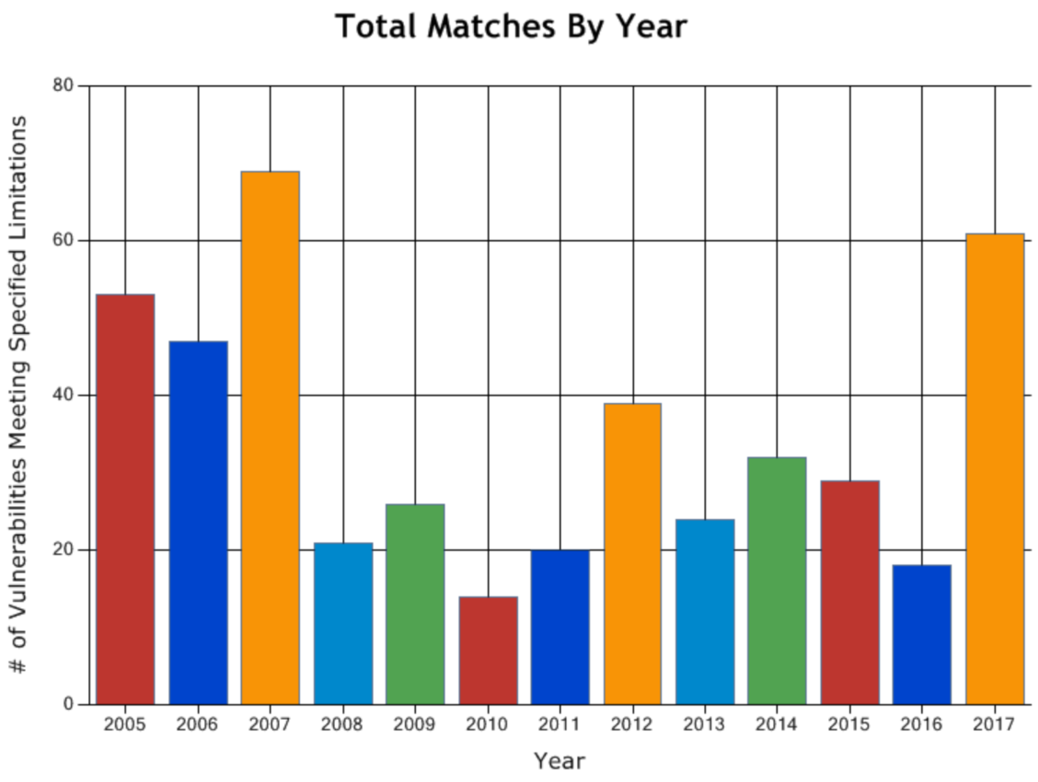

Figure 6. Number of Vulnerabilities Discovered in Firewall Software (Adapted from National Vulnerability Database)

- Intrusion-detection systems (IDS) and intrusion-detection and prevention systems (IDPS) analyze network traffic to detect cyberattacks and alert system administrators when they do. These systems often constitute a key component in efforts to implement the “defense in depth” strategy that security practitioners use to project critical systems from cyberattacks. In fact, IDS is the most commonly used mechanism to detect cyberattacks on cloud systems (Mehmood, Habiba, Shibli, & Masood, 2013). However, IDS and IDPS offer their own set of unique challenges; for example: 1) practitioners find it difficult to decide where to place an IDS and how to best configure it when used in a distributed environment with multiple stakeholders (Werlinger, Hawkey, Muldner, Jaferian, & Beznosov, 2008), 2) several IDS and IDPS have a poor design and/or have incomplete database of attack signatures (Beigh, Bashir, & Chachoo, 2013), 3) threat agents often target IDS (Corona, Giacinto, & Roli, 2013), and 4) IDS have their own vulnerabilities (see Figure 7).
- To summarize, 1) one needs to accept that bad code (i.e., software with vulnerabilities) is inevitable; 2) developers use patches, updates, and technical controls to minimize the risk of cybersecurity incidents due to software vulnerabilities; and 3) patches, updates, and technical controls are also software, which means they can have their own vulnerabilities. Therefore, we have yet to successfully address the key technical challenge to cybersecurity (i.e., bad code). In Section 3, I analyze the economic factors that contribute to poor cybersecurity.

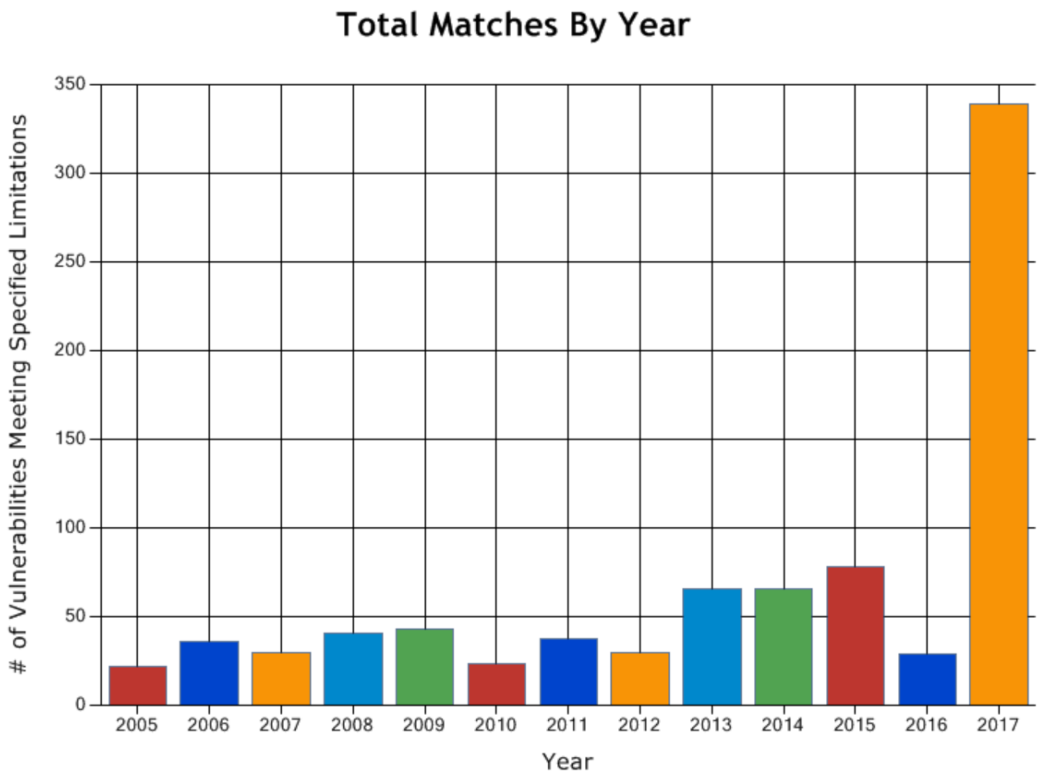

Figure 7. Number of Vulnerabilities Discovered in IDSs (Adapted from National Vulnerability Database)

3 Economic Challenges to Cybersecurity

The current literature on the economics of cyber security has identified several reasons that contribute to poor cybersecurity.

3.1 Cooperation and Coordination Problems

Competing firms all benefit from sharing cybersecurity information among themselves (Gal-Or & Ghosh, 2005). Government initiatives such as the National Vulnerability Database (NVD) have resulted in sharing of vulnerability information among developers and users. Similarly data breach disclosure laws have forced organizations to share information about data breach incidents with those affected by the incident within a certain time period. However, we have yet to see any effective voluntary initiatives among relevant stakeholders in this direction. For example, cooperation among nation states can minimize the likelihood that cyberwar will occur. All countries with cyberwar capabilities need to reach an agreement (e.g., as in the case of chemical and nuclear weapons) that limits and ultimately curbs their cyberwar program. However, for such an agreement to happen, each nation has to: 1) realize that their cyberwar capabilities are no better than that of their adversaries, 2) trust that their adversaries will adhere to the agreement, 3) believe that they are better off by adhering to the agreement instead of breaking it. The fact that we have yet to witness such an agreement suggests that the conditions required for it do not exist. Consider another case: Internet service providers (ISP). ISP are in the best position to stop (or at the least minimize) attack traffic (e.g., distributed denial of service (DDoS), malware) from traveling through their networks. However, to be effective, they need to cooperate and coordinate with other ISP. This cooperation and coordination would require ISP to invest more in resources required to closely scrutinize network traffic, punish those ISP that do not cooperate, and reward those that do. At present, most if not all ISP seem to have no economic rationale to cooperate and make the additional investments required for the subsequent coordination.

3.2 Perverse Incentives and Lack of Incentives

Researchers have often blamed perverse and/or a lack of incentives for poor cybersecurity (Anderson, 2001; Moore & Anderson, 2012). Some examples include:

- Some software producers reward their developers for how many bugs they have fixed (Maiguy, 2013). Without accountability for who created the flaw, this practice simply provides a perverse incentive to the developers to release vulnerable software that they will then subsequently be rewarded for "fixing". It is probably better to reward developers for "bug-free features" or "lowest bug count" (Maiguy, 2013).
- A lack of incentives plays a role in cases where the entity tasked with protecting information assets is not the one that primarily suffers from cybersecurity incidents (e.g., Anderson, 2001), such as in the case of organizations who operate critical infrastructure (Moore, 2010). Take the example of the U.S. Office of Personal Management (OPM). OPM acquires, stores, and manages employee data on millions of Americans. It suffered a massive data breach that it publicly disclosed in 2015 (Koerner, 2016). While the federal employees whose information was stolen faced possible identity theft as a result of this incident, it is not clear if OPM faced any serious sanctions for its failure to protect employee data. In the absence of such sanctions, OPM itself has minimal economic incentive to invest in its data-protection operations.

3.3 Asymmetric Information

When software users have less information than developers about software quality (in terms of number of vulnerabilities in the software), the market will likely comprise vulnerable software (Anderson, 2001). To evaluate software, a user organization would have to spend a significant amount of time and resources in testing the software to find its vulnerabilities. However, an organization is unlikely to do so for the same economic reasons that the software producer had when it released vulnerable software. That is, the investment in time and resources required to completely assess software for unknown vulnerabilities is not worth the effort. This situation leads to information asymmetry between the user organization and the software vendor, and, therefore, the market for enterprise software also likely suffers from quality concerns (Anderson, 2001).

3.4 Asymmetric Conflict

The conflict between parties who defend software systems and those who attack them favors the attackers. The defenders have to find out all the software vulnerabilities before anyone else does and patch them. On the other hand, attackers only have to find one vulnerability that the defender has not yet discovered/patched and exploit it (Anderson, 2001). Defenders respond to attacks by patching known vulnerabilities; yet, as soon as they fix one flaw, attackers identify and exploit another one (Böhme & Moore 2009). Chia, Chuang, and Chen (2016) compare this scenario with the game of whack-a-mole. They propose that, even with a resource disadvantage, an attacker can use information asymmetry (i.e., knowledge of vulnerabilities unknown to the defender) to its advantage.

3.5 Financial Gains

In the global vulnerability marketplace, the world's top "bug bounty hunters" can earn financial rewards for their discoveries (Roberts, 2015). Market makers such as HackerOne[13] and Zerodium[14] provide an easy-to-use platform to match sellers and buyers of vulnerabilities. Organizations as diverse as software developers (e.g., Microsoft, Google, Yahoo), automobile manufacturers (e.g., Fiat Chrysler Automobiles, General Motors), government organizations (e.g., Pentagon, U.S. Department of Health and Human Services), and security consultants (e.g., iDefense) offer bounty programs that invite researchers and hackers to find vulnerabilities in software and receive pay for their efforts[15]. In addition to the sponsors of bounty programs, the buyers of vulnerabilities also include nation states, military intelligence, and law enforcement agencies. Even if we ignore the ethical concerns associated with such markets, we still have to deal with the predicament that these unregulated markets are not optimal for social welfare in most cases (Kannan & Telang, 2005). In fact, federally funded mechanisms, such as CERT, for disclosing

[13] See https://www.hackerone.com/
[14] See https://zerodium.com/
[15] See http://securityledger.wpengine.com/?s=bug+bounty

vulnerability information always perform better (Kannan & Telang, 2005). Nevertheless, the vulnerability markets exist and are here to stay. These markets provide financial incentives to individuals and organizations to discover and sell vulnerabilities, which result in more discovered vulnerabilities. The proponents argue that these markets speed up the process of vulnerability discovery and, as a result: 1) motivate software developers to come up with relevant patches and 2) warn users so that they can take steps to protect against potential cyberattacks. On the other hand, more discovered vulnerabilities also provide more opportunities to threat agents and, therefore, translate into more attacks against vulnerable systems (Arora et al., 2006). One way to counter these vulnerability markets rests in developing better vulnerability-discovery and disclosure mechanisms. These mechanisms can provide incentives to individuals and organizations to discover vulnerabilities and share relevant information with users and software developers more quickly. By preempting the financially motivated "bug hunters", such mechanisms will weaken the existing markets for vulnerabilities. Google's Project Zero constitutes a good example. It's most recent success came in early 2018 when its researchers were among those who discovered the vulnerabilities Spectre[16] and Meltdown[17] in widely used chips from Intel, ARM, and AMD (Neuman, 2018). However, we need more such initiatives to counter or significantly weaken existing vulnerability markets.

3.6 Weak Market Forces

Market forces such as competition, consumer reaction, and investor responses are instrumental in encouraging companies to improve their products and services. These forces are, however, weak in the software industry.

3.6.1 Little Competition

A competitive software market requires transparent prices. In the software market, while most desktop and mobile applications have reasonably transparent pricing, enterprise software does not (e.g., enterprise resource planning (ERP) and customer relationship management (CRM)). For infrastructure and enterprise software, their owners do not often publish its price. When they do, they do so in in a manner that one cannot easily objectively compare with competing products (e.g., setup, maintenance, configuration, customizations, and total cost of ownership). A competitive market also requires a certain number of competitors; however, major software market segments suffer from a lack of serious competition. For example, Microsoft and Apple dominate the desktop operating system segment, Microsoft leads the office productivity segment (e.g., MS Office), Google's Android (and its derivatives) and Apple's iOS dominate the mobile operating system segment. The enterprise software segment has seen a trend toward growth by acquisition (e.g., SAP's acquisition of Business Object, and Oracle's acquisition of PeopleSoft), which has resulted in few dominant players. This lack of competition has an adverse impact on not only developers' motivation to write secure code but also the way software producers respond to vulnerabilities discovered in their products. For example, in a recent study, Jo (2017) found that a lack of competition among software vendors adversely impacts a software vendor's response to vulnerabilities in its product (i.e., releasing patches). One explanation could be that, in software market segments with a dominant vendor, software users have relatively weak bargaining power. Therefore, these users have minimal influence on vendor's patch release behavior, which, in turn, could result in delayed patch releases.

3.6.2 Unconcerned Investors

Multiple studies have shown that an announcement that a firm (other than software producer) has experienced a cybersecurity breach negatively impacts its stock price (Cavusoglu, Mishra, & Raghunathan, 2004; Acquisti, Friedman, & Telang, 2006; Telan & Wattal 2007; Kannan, Rees, & Sridhar, 2007). This impact is more severe in competitive markets. However, the cybersecurity breaches in these studies had no longer-term effect on the market valuation of the affected firms (Cavusoglu et al., 2004; Acquisti et al., 2006; Kannan et al., 2007). Thus, given that investors' tepid reaction to cyberattacks, victim organizations seem to have minimal incentive to demand secure software from their software vendors. In cases where a software producer suffers from a cyberattack, the same logic applies. In the absence of a strong disapproval from investors, software producers are more likely to continue doing business as usual.

[16] See https://spectreattack.com/spectre.pdf
[17]See https://meltdownattack.com/meltdown.pdf

3.6.3 Apathetic Customers

If customers respond to discoveries of vulnerabilities in software products by switching to competing products, then software producers might have an economic incentive to produce more secure software (at least in comparison to their competition). However, existing evidence suggests that customers are at best apathetic to security weaknesses either in the software that they themselves use or in the software that the vendors of the product and services that they purchase use. A survey that the Ponemon Institute conducted for Experian found that repeated announcements about data breaches and other cyberattacks has made consumers immune to these events (Experian, 2014). Found from conducting surveys with users, Ablon, Heaton, Lavery, and Romanosky (2016) found that they were unlikely to switch firms even after the firms have suffered from data breaches. In fact, only 11 percent of respondents stopped dealing with the affected company following a breach. Therefore, these firms have minimal incentive to force their software vendors to provide more secure code.

In brief, most software market segments feature a lack of serious competition, unfazed investors, and indifferent customers. As a result, market forces only weakly (at best) motivate firms to invest in cybersercuity. Furthermore, asymmetric information between software users and developers, the inherent advantage that attackers enjoy over defenders, and perverse incentives also have an adverse impact on cybersecurity. In Section 4, I investigate software users' contribution to poor cybersecurity.

4 Software Users

The number of people who use software and have access to high-speed Internet has continued to grow. According to Internet World Stats (Miniwatts Marketing Group, 2018), this number was 3.89 billion in 2017—almost half of the world population. According to researchers and security professionals, these users must share a large proportion of the blame for lack of cybersecurity (Krazit, 2016; Culp, 2016).

4.1 Unprepared and/or Irresponsible

Software users do not always follow security procedures such as encrypting sensitive data, enabling access controls, and changing default access credentials (e.g., username and password). Sometimes, they just need to upgrade their software to fix known vulnerabilities. However, they often continue to use old and often discontinued software. For example, people still use Windows XP, an operating system that Microsoft stopped supporting on 8 April, 2014 (Bell, 2016). Surprisingly, as of October, 2017, Windows XP was the fourth-most popular desktop operating system (OS) in use (see Figure 8). Further, even third-party developers such as Google have stopped providing applications for Windows XP (Kleinman, 2015). By continuously using unsupported software such as Windows XP, users put themselves at risk from cyberattacks such as the ransomware WannaCry (Erlanger, Bilefsky, & Chan, 2017).

Security professionals also fail to update their software in a timely manner. A 2015 survey by Google found that: 1) more than one third of security professionals do not keep their systems current, 2) only 64 percent of security experts update their software automatically or immediately when notified about the availability of a new version, and 3) just 38 percent of regular users do the same (Redmiles, 2017). Another study in 2012 that Skype sponsored found that 40 percent of users do not update their software when prompted and about a quarter skip the updates because they did not understand the benefits (McAllister, 2012).

4.2 Behind on Patch Application

As I establish in Section 2, one cannot develop software with zero vulnerabilities. Currently, software developers usually apply patches to fix vulnerabilities. However, this approach has a questionable effectiveness because many users do not update their systems. From the time a patch appears, it takes software engineers and regular users 24 days and 45 days on average, respectively, to apply it (Redmiles, 2017). One might ask why users do not patch their systems in a timely manner. The answer could lie in a 2016 survey by researchers from the University of Edinburgh and Indiana University (Redmiles, 2017). The researchers asked participants to discuss their experiences of installing software patches. Nearly half of them said the high frequency of patch releases frustrated them. Many users also expressed dissatisfaction with the time it took to apply patches, lost productivity during that time, and software-regression concerns (especially in cases of specialized and customized software). The companies that distribute patches do not always help as well. For example, Microsoft released 18 patches on 14 March, 2017. Among these patches included one that fixed the vulnerability that the ransomware WannaCry later exploited. However, the patch rating score that Microsoft provided did not clearly convey the severity of this vulnerability. Even security experts failed to identify the fix for WannaCry as a critic patch to apply (Redmiles, 2017). As a result, even though one could already obtain a patch that fixed it, WannaCry managed to infect hundreds of thousands of unpatched Windows machines in more than 150 countries in May, 2017.

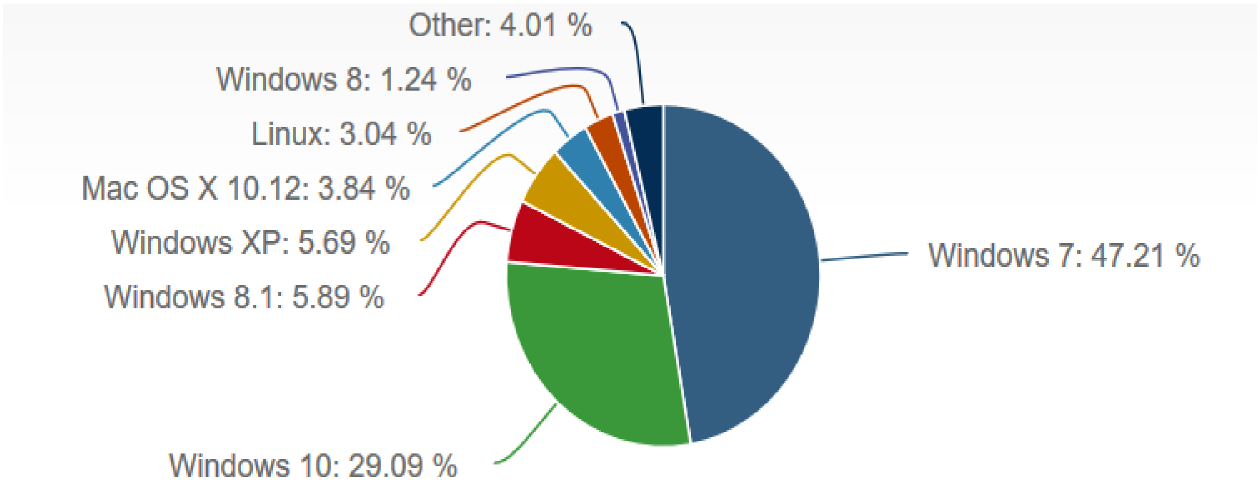

Figure 8. Desktop Operating System Market Share October, 2017 (Netapplications.com, 2017)

4.3 Unaware and/or Poorly Trained Users

According to the ACC Foundation's "State of Cybersecurity Report", cybersecurity incidents often result from employee error (ACC Foundation, 2015). BUPA, an international healthcare group headquartered in the United Kingdom that serves 32 million customers in 190 countries, revealed that their most recent security breach occurred when one of their employees inappropriately copied and removed information (Stanley, 2017). This incident is not isolated. In fact, many cybersecurity professionals believe that individuals (i.e., users and employees) share the most blame for cybersecurity incidents (Krazit, 2016; Culp, 2016). For example:

- Individuals often use weak passwords, share their passwords with friends and family, and use the same password for multiple accounts/applications. For instance, closely examining some major recent data breaches (e.g., Yahoo, U.S. Office of Personnel Management, Anthem, and the Democratic National Committee) reveals a common theme. In every incident, the attack vector has been a weak password (Chertoff, 2016). In organizational settings, one can easily address concerns related to weak passwords by enforcing strong password policies on employees. However, for personal systems, this approach might not be practical. In theory, software developers can implement strong password-based access controls or biometric-based access controls in their applications. However, doing so might adversely impact users from adopting the software. Therefore, developers might hesitate to enforce strong access-control mechanisms. Instead, they are more likely to offer various access control options (e.g., PIN code, password, biometric, etc.) and leave the decision to users while hoping that they will chose the strongest possible access control. However, no empirical evidence suggests that users chose the most secure access control option.
- Individuals frequently fall prey to social-engineering attacks such as phishing (e.g., Canfield, Fischhoff, & Davis, 2016). Vishwanath (2016) found several reasons as to why people fall victim to phishing attacks: 1) they naturally seek "cognitive efficiency" (i.e., maximal information for minimal brain effort), 2) when they see familiar logos and brand names, they assume that the (phishing) message is legitimate and act accordingly, 3) some wrongly assume that their online actions are inherently safe, and 4) some become complacent because they routinely use emails and do not spend enough time in scrutinizing the emails that they receive. To help alleviate these issues, Security experts often provide relevant training to employees, which should reflect the latest threats and occur as frequently as possible, (e.g., Drolet, 2016).

Vishwanath (2016) suggests that training should simulate phishing attacks and provide relevant feedback to users about their performance. However, this training has a debatable effectiveness. For instance, 32 percent of employees who received training at a major bank clicked on a phishing link in the weeks following a training class in comparison to 35 percent of those who received no training (Prince, 2017). In another case, JPMorgan boosted its cybersecurity spending after a data theft, but, several weeks after the training when it tested the staff with a fake phishing email, 20 percent of them clicked on it (Drolet, 2016). In a real scenario similar to the test, that action would have downloaded a malicious payload onto the bank's network (Drolet, 2016).

- Individuals do not seem to understand online threats even after continuous warnings from the system (Wolff, 2015) and frequently ignore these warnings (Bravo-Lillo, Cranor, Downs, & Komanduri, 2011). For example, Felt et al. (2015) found that people generally ignore SSL/TLS warnings: the messages that pop up in the browsers when a server cannot be authenticated or something is wrong with the encryption that protects the traffic between the browser and the server. However, many people continue with their browsing regardless of the warning. One could blame the presentation of these warning messages. However, Wolff (2015) found that changing the wording of warning seems to have minimal effect on user behavior.

In brief, software users do not update and patch their software in a timely manner, do not often know about best cybersecurity practices, and often do not change their behavior after training. As a result, they continue to be instrumental in cybersecurity attacks. In the next section, I assess the role that laws and regulations play in the area of cybersecurity.

5 Ineffective Laws and Regulations

Any cybersecurity incident has several key stakeholders are: 1) the victim(s), 2) the threat agent(s) blamed for the incident (e.g., hackers, cyber criminals, nation states etc.), 3) the producer of the vulnerable system, and 4) the custodian of the system responsible for maintaining the system. Figure 9 illustrates the different scenarios that represent the relationship between these stakeholders. The question is: how do the existing laws relevant to cybersecurity assign responsibility and assess liability among these stakeholders? I address this question by categorizing the laws into three types: 1) those that apply to threat agents, 2) those that apply to the custodian of legally protected data, and 3) those that apply to the software developer.

5.1 Laws Applicable to Threat Agents

Most countries have laws to target the threat agents of cybersecurity incidents. For example, in the US, several laws deter individuals and organizations from unauthorized intrusions into computer systems (e.g., the Computer Fraud and Abuse Act (CFAA), National Information Infrastructure Protection Act (NIIPA), Electronic Communications Privacy Act (ECPA), Economic Espionage Act (EEA), Wire Fraud Act (WFA), National Stolen Property Act (NSPA), and Identity Theft and Assumption Deterrence Act (ITADA)). While these laws have been somewhat effective in discouraging threat agents in domestic boundaries, they have often been useless against cyberattacks that have originated from outside their jurisdiction. Cyberwars exemplify such attacks. On 27 April, 2007, Estonia suffered a crippling cyberattack launched from outside its borders. In 2009, attackers—allegedly the US and Israel, targeted an Iranian nuclear reactor facility using Stuxnet. More recently in December, 2015, and then again in December, 2016, the suspected Russian attack on Ukraine's power grids, government department, business organizations illustrate the possibility and the capabilities of cyberwar (Greenberg, 2017). In all such cases, it is still unclear what legal rights a state or an organization has when it falls victim to cyberattacks from another country. The first hurdle in responding to such incidents is whether to classify them as merely cybercrimes or as cyberwar. Then one needs to attribute the attacks to the right threat agents. Even if one can identify the threat agents and one has solid evidence against them (e.g., as in the case of Russian cyberattack on Ukraine or North Korea's role in releasing WannaCry (Reuters Staff, 2018), no clear consensus on how the victims could legally respond exists (e.g., with armed force, its own cyberattack, or take some other measures). Compounding the problem, the international laws that one could potentially apply to cyberwar are weak to non-existent (Shackelford, 2009). Hathaway et al. (2012) have offered suggestions on how to apply a combination of domestic criminal laws and existing international laws and treaties, with some amendments and modifications, to meet the distinct challenges that cyberattacks from nation states pose. However, the effectiveness of this approach has not yet been established because most of the existing international laws and treaties that touch on cybersecurity do not carry practical enforcement provisions (Shackelford, 2009).

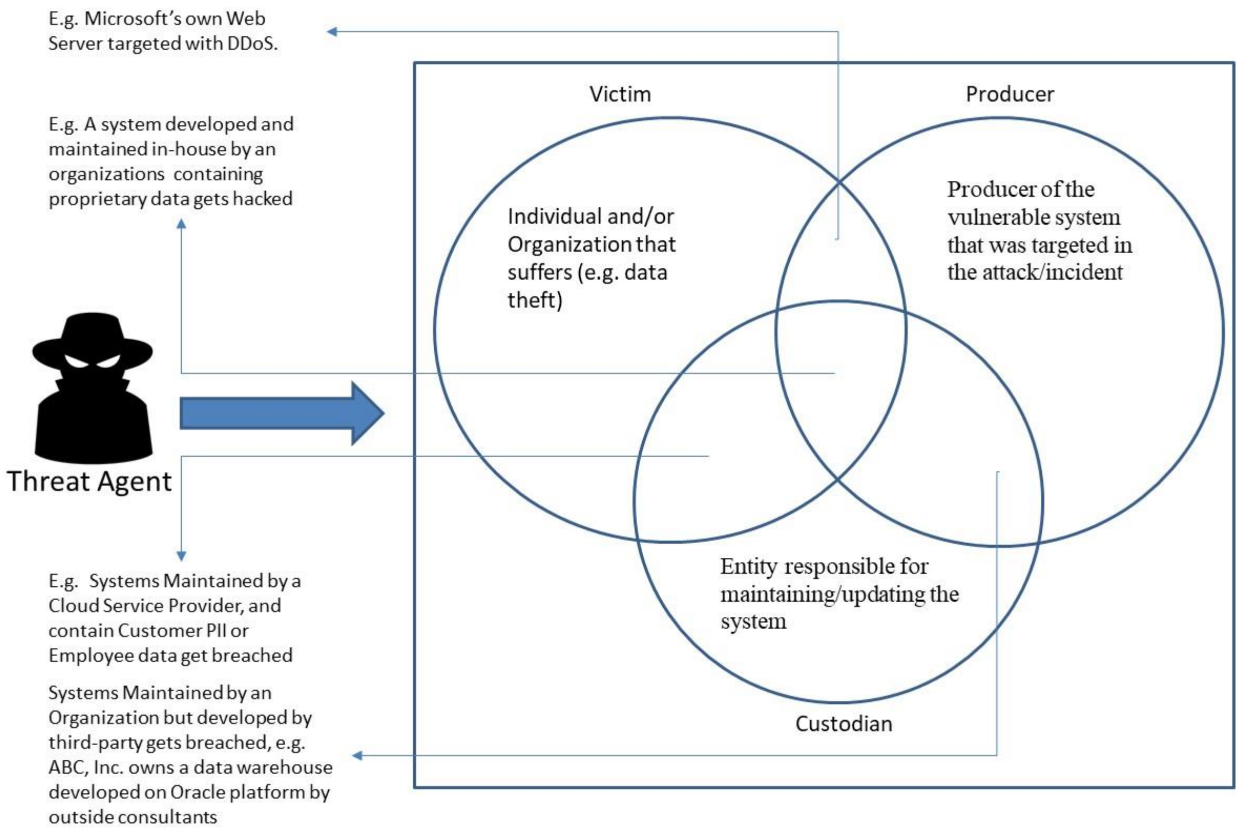
E.g. Microsoft's own Web Server targeted with DDoS.
E.g. A system developed and maintained in-house by an organizations containing proprietary data gets hacked
Threat Agent
E.g. Systems Maintained by a Cloud Service Provider, and contain Customer PII or Employee data get breached
Systems Maintained by an Organization but developed by third-party gets breached, e.g. ABC, Inc. owns a data warehouse developed on Oracle platform by outside consultants
Victim
Producer
Individual and/or Organization that suffers (e.g. data theft)
Producer of the vulnerable system that was targeted in the attack/incident
Entity responsible for maintaining/updating the system
Custodian

Figure 9. Stakeholders in a Cybersecurity Incident

5.2 Laws Applicable to Collectors and/or Custodians of Legally Protected Data

Some laws in the US (e.g., HIPPA, GLB, and Sarbanes-Oxley) require individuals and organizations to protect certain types of data stored on their systems (e.g., personally identifiable information (PII), sensitive personal information (SPI), health, financial). Furthermore, some federal and state data breach notification laws require organizations to disclose any data breach that they suffer (Bisogni, 2016). However, the effectiveness of these laws in reducing the risk of cybersecurity incidents (e.g., data breach) remains unproven. While, Romanosky, Telang, and Acquisti (2011) found that data breach disclosure laws reduce identity theft resulting from data breach disclosure, it does not necessarily mean that these laws result in a decrease in the number of data breach incidents. Sen and Borle (2015) found that state data breach disclosure laws did not significantly impact the risk of data breaches in those states. Furthermore, the trend towards outsourcing data storage (e.g., to cloud platforms operated and managed by a third party) also complicates the application of these laws. For example, does the cloud computing environment absolve business entities of their responsibility to ensure proper data security or do they share joint responsibility with cloud service providers (Kaufman, 2009)?

5.3 Laws Applicable to Software Producers

Academics and practitioners have long debated whether one should subject software producers and developers to strict product liability laws (e.g., Chong, 2013; Evans, 2015; Goertzel, 2016, Beard, Ford, Koutsky, & Spiwak, 2010). When applied, strict liability makes a manufacturer responsible for all physical injuries and associated losses caused by a defective product that it produces (HeinOnline, 1998). Strict liability has resulted in better-quality products in several industries (e.g., automobile, toys, and medical equipment). However, product liability laws do not seem to have had any impact on software's quality.

We have yet to see any successful lawsuit against software producers under the strict product liability laws, though a case against Toyota represents one exception. It took a decade-long investigation and experts' reviewing software code over 28 months to prove that faulty code caused unintended accelerations in Toyota vehicles. These unintended accelerations resulted in several accidents. Toyota had to recall nine million vehicles and pay billions in settlements and fines (Somers, 2017). Clearly, proving that software vulnerability causes physical injury is a major challenge to suing software developers under strict liability laws (Beard et al., 2010). In addition, end-user software licenses (EULA) protect software developers from legal consequences due to strict liability laws. Almost all commercial over-the-counter software (COTS) comes with a EULA. EULA also cover customized software developed by third parties. Their wording means that one cannot easily establish software as a product, a key requirement of product liability laws. EULA establish software as being a service that the software producer provides to end users. This service comprises the clauses that allow users to use the software under strict and specific conditions. The EULA effectively relieve the software developers from any liability due to harm that their product may cause (Beard et al., 2010; Rustad & Koenig, 2005).

One might ask whether one can counter EULA by suing software producers for breach of warranty under contract laws. The United States has contract laws that one could use to sue software producers for breach of implied and expressed warranties (Goertzel, 2016). For instance, if COTS software does not work as promised, promoted, and marketed, one could use contract laws to sue software producers (Rustad & Koenig, 2005). However, the liabilities under this approach involve only the cost of the software. Furthermore, U.S. court rulings (in cases brought under contract breach) interpret software license agreements (i.e., EULA) as fair contracts and have always sided with software developers. These rulings also presume that the threat agents, not software developers, are primarily responsible for cybersecurity incidents (Chong, 2013). Finally, it is unclear if the Uniform Commercial Code (UCC), which defines the widely accepted interpretation of civil contract law in the USA, applies to: 1) software licenses because they do not transfer ownership but only the right to use the developer-owned product from the producer to the user and 2) free software applications/platforms whose producers do not charge users to use the them (Chong, 2013).

One may also ask about using the tort theory of negligence to hold software producers legally accountable for their vulnerable products. This law applies to anyone under a contract to deliver the services of producing, maintaining, or supporting the software product or another service in which delivering the software product plays a role (Goertzel, 2016). The approach rests on the premise that: 1) the software vendor owes the user a duty to provide functioning software, 2) the software did not live up to that standard, 3) the user suffered harm, and 4) the software caused that harm. The software user faces a challenge in proving that software vulnerabilities resulted from the producer's failure to apply due care when designing or implementing the software. The software developers can successfully defend against negligence if they can prove that they could not have detected and corrected the vulnerability that caused the security breach through "reasonable" software development practices. A more rigorous form of tort negligence is tort theory of malpractice, which holds defendants in recognized licensed professions such as medicine, architecture, engineering, and so on liable. To use this approach, the plaintiff must prove that the software developers belong to a recognized, ideally government-licensed, profession and have failed to comply with the standards of that profession while engineering the defective software product (Goertzel, 2016). However, this approach suffers from the fact that a widely recognized system of professional licensure for software developers has not yet been established in the US or elsewhere.

To summarize, specific laws that target threat agents and custodians of protected data exist. While the laws targeting custodians of protected data have been somewhat effective in motivating organizations to better protect data, a lack of attribution and jurisdiction often constrain the laws that target threat agents. At present, no specific laws target software producers. Therefore, we must use existing laws on product liability, contract, and malpractice to take legal action against vulnerable software producers. However, EULA often successfully challenge these laws. The lack of consensus on whether software is a product or a service adds to the legal challenge. The opinion of some that software code expresses free speech further complicates the matter. For example, the Ninth U.S. Circuit Court of Appeals ruled in 1999 that the First Amendment protects source code (Balakrishnan, 2016). More recently, Apple, in a case against FBI, used the First Amendment as defense to prevent the FBI from forcing it to extract data from a confiscated iPhone. If the view that "software classifies as free speech" gains momentum, then it will become much more difficult to prosecute software producers.

6 Conclusion

Organizations continue to allocate increasing resources towards cybersecurity initiatives. Despite these efforts, the frequency of cyberattacks continues to grow. I propose that vulnerable software, certain economic factors, a lack of legal accountability for software producers and certain types of threat agents, and software users' behavior contribute to poor cybersecurity. Figure 10 illustrates some of these challenges to cybersecurity.

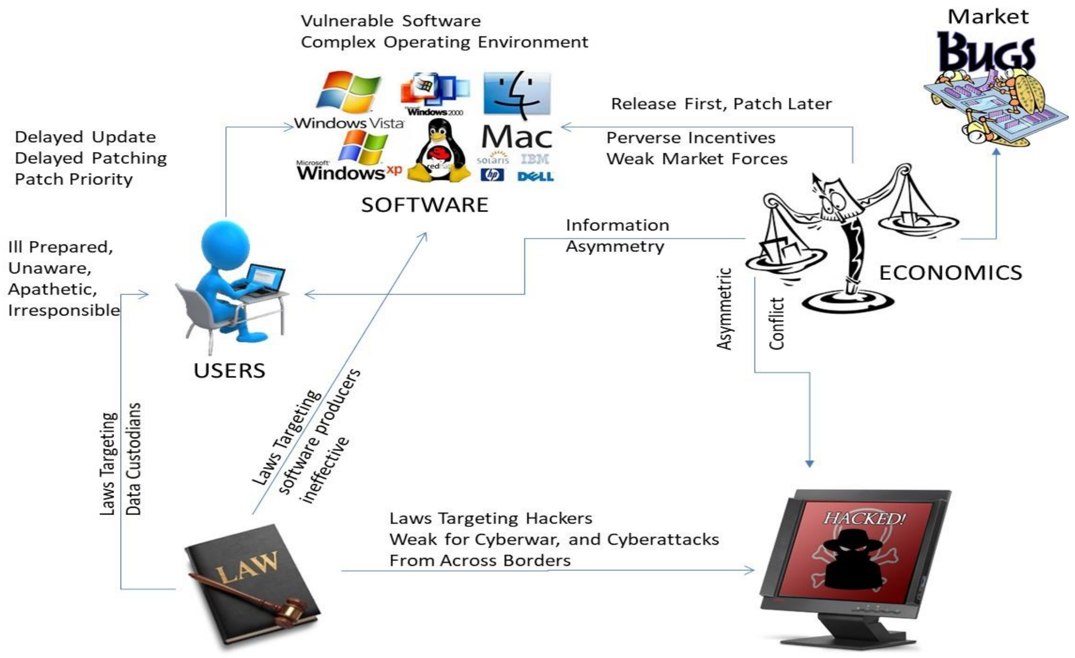

Figure 20. Factors Contributing Towards Poor Cybersecurity

I believe that vulnerable software constitutes the main reason for poor cybersecurity. However, due to certain technical (e.g., product complexity), economic (e.g., information asymmetry, release first patch later, perverse incentives), and legal (e.g., ineffective laws targeting software producers) reasons, software developers will continue to produce vulnerable software. While software developers use various development strategies to minimize the vulnerabilities in their software, they seem to have given up on the goal to produce vulnerability-free software. Appropriate public policies that eliminate perverse incentives and encourage software developers to spend more resources on eliminating bugs before releasing their software can address the economic reasons that contribute to the availability of vulnerable software. However, we have yet to see any meaningful steps taken in this direction. Finally, we also need to modify and update the laws relevant to cybersecurity to encourage software producers to write more secure code. Even if we assume, for a moment, that we have vulnerability-free software, effective laws, and relevant public polies, we still need to address the behavioral challenges that individual users of software pose. As long as these individuals remain careless, apathetic, and unaware about their role in improving cybersecurity, the state of cybersecurity will remain weak.

Given the current challenges to cybersecurity and our inability to overcome them so far, it would be unrealistic to expect reasonable cybersecurity anytime soon. One can only hope that, over a longer period, security researchers, software developers, legal scholars, public officials, and software users will better understand their role in improving cybersecurity. As a result, they might be more successful in addressing some of the challenges that I present in this paper.

References

Ablon, L., Heaton, P., Lavery, D., & Romanosky, S. (2016). *Consumer attitudes toward data breach notifications and loss of personal information*. Santa Monica, CA: The RAND Corporation.

ACC Foundation. (2015). *The state of cybersecurity*. Retrieved from http://www.acc.com/legalresources/resource.cfm?show=1416923

Acquisti, A., Friedman, A., & Telang, R. (2006). Is there a cost to privacy breaches? An event study. In *Proceedings of the International Conference on Information Systems.*

Altinkemer, K., Rees, J., & Sridhar, S. (2008). Vulnerabilities and patches of open source software: An empirical study. *Journal of Information System Security*, *4*(2), 3-25.

Anderson, R. (2001). Why information security is hard—an economic perspective. In *Proceedings of the 17th Annual Computer Security Applications Conference* (pp. 358-365).

Ashford, W. (2012). Microsoft: Is computing more trustworthy 10 years on? *ComputerWeekly.com*. Retrieved from http://www.computerweekly.com/feature/Microsoft-Is-computing-more-trustworthy-10-years-on

Ashibani, Y., & Mahmoud, Q. H. (2017). Cyber physical systems security: Analysis, challenges and solutions. *Computers & Security*, *68*, 81-97.

Arora, A., Nandkumar, A., & Telang, R. (2006). Does information security attack frequency increase with vulnerability disclosure? An empirical analysis. *Information Systems Frontier*, *8*(5), 350-362.

Balakrishnan, A. (2016). Does computer code count as free speech? *CNBC.* Retrieved from: https://www.cnbc.com/2016/03/01/apple-question-is-code-free-speech.html

Beard T. R., Ford, G. S., Koutsky, T. M., & Spiwak, L. J. (2010). Tort liability for software developers: A law and economics perspective. *John Marshall Journal of Computer and Information Law*, *1*(7), 199-234.

Beigh, B. M., Bashir, U., & Chachoo, M. (2013). Intrusion detection and prevention system: Issues and challenges. *International Journal of Computer Applications*, *76*(17), 26-30.

Bell K. (2016). Windows XP the third-most popular OS after 15 years. *TechnoBuffalo.* Retrieved from https://www.technobuffalo.com/2016/04/08/windows-xp-the-third-most-popular-os-after-15-years/

Bisogni, F. (2016). Proving limits of state data breach notification laws: Is a federal law the most adequate solution? *Journal of Information Policy*, *6*, 514-205.

Böhme, R., & Moore, T. (2009). The iterated weakest link—a model of adaptive security investment. In *Proceedings of the 8th Workshop on the Economics of Information Security.* Retrieved from http://weis09.infosecon.net/files/152/paper152.pdf

Boldt, B. (2017). Automotove security in a CAN. *ElectronicDesign.* Retrieved from http://www.electronicdesign.com/automotive/automotive-security-can

Bravo-LiLLo, C., Cranor, L. F., Downs, J. S., & Komanduri, D. S. (2011). Bridging the gap in computer security warnings a mental model approach. *IEEE Security and Privacy*, *9*(2), 18-26

Canfield, C. I., Fischhoff, B., & Davis, A. (2016). Quantifying phishing susceptibility for detection and behavior decisions. *The Journal of the Human Factors and Ergonomics Society*, *58*(8), 1158-1172.

Chertoff, M. (2016). Passwords are the weakest link in cybersecurity today. *CNBC*. Retrieved from https://www.cnbc.com/2016/10/06/passwords-are-the-weakest-link-in-cybersecurity-today-michael-chertoff-commentary.html

Chia, P. H., Chuang, J., & Chen, Y. (2016). Whack-a-mole: Asymmetric conflict & guerrilla warfare in Web security. In *Proceedings of the 15th Annual Workshop on the Economics of Information Security.*

Chong, J. (2013). Bad code: Should software makers pay? *New Republic*. Retrieved from https://www.newrepublic.com/paper/114973/bad-code-should-software-makers-pay-part-1/

Colbert, E. (2017). Security of cyber-physical systems. *Journal of Cyber Security and Information Systems*, *5*(1).

Conti, J. P. (2010). The day the samba stopped. *Engineering Technology*, *5*(4), 46-47.

Cavusoglu, H., Mishra, B., & Raghunathan, S. (2004). The effect of Internet security breach announcements on market value: Capital market reactions for breached firms and Internet security developers. *International Journal of Electronic Commerce*, *9*(1), 70-104.

Culp, S. (2016). Cyber risk: People are often the weakest link in the security chain. *Fortune.* Retrieved from https://www.forbes.com/sites/steveculp/2016/05/10/cyber-risk-people-are-often-the-weakest-link-in-the-security-chain/#41f74aca2167

Corona, I., Giacinto, G., & Roli, F. (2013). Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Information Sciences*, *239*(1), 201-225.

Drolet, M. (2016). Cybersecurity is only as strong as your weakest link—your employees. *CSO*. Retrieved from https://www.csoonline.com/paper/3095486/security/cybersecurity-is-only-as-strong-as-your-weakest-linkyour-employees.html

Erickson, J. (2008). *Hacking: The art of exploitation* (2nd ed.). San Francisco, CA: William Pollock.

Erlanger, S., Bilefsky, D., & Chan S. (2017). U.K. health service ignored warning for months. *The New York Times*. Retrieved from https://www.nytimes.com/2017/05/12/world/europe/nhs-cyberattack-warnings.html

Evans, J. (2015). Should software companies be legally liable for security breaches? *TechCrunch*. Retrieved from https://www.techcrunch.com/2015/08/06/ should-software-companies-be-legally-liable-for-security-breaches/

Experian. (2014). *Aftermath of a mega data breach: Consumer sentiment*. Retrieved from http://www.experian.com/data-breach/2014-aftermath-studyconsumersentiment.html

Farwell, J. P., & Rohozinski, R. (2011). Stuxnet and the future of cyber war. *Survival*, *53*(1), 23-40.

Felt, A., Ainslie, A., Reeder, R.W., Con, S., Tyagaraja, S., Bettes, A., Harris, H., & Grimes, J. (2015). Improving SSL warnings: Comprehension and adherence. In *Proceedings of the Conference on Human Factors in Computing Systems.*

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

Goertzel, K. M. (2016). Legal liability for bad software. *CrossTalk*, *29*(5), 23-27.

Greenberg, A. (2017). How and entire nation became Russia's test lab for cyberwar. *Wired*. Retrieved from https://www.wired.com/story/russian-hackers-attack-ukraine/

Haigh, T., & Landwehr, C. (2015). Building code for medical device software security. *IEEE Cybersecurity*. Retrieved from https://cybersecurity.ieee.org/blog/2015/11/15/building-code-for-medical-device-software-security/

Hartnett, K. (2016). Computer scientists close in on perfect, hack-proof code. *Science.* Retrieved from https://www.wired.com/2016/09/computer-scientists-close-perfect-hack-proof-code/

Hathaway, O. A., Crootof, R., Levitz, P., Nix, H., Nowlan, A., Perdue, W., & Spiegel, J. (2012). The law of cyber-attack. California Law Review, *100*(4), 817-885.

Howard, C. (2012). UAVs, software, and security: An interview with Robert Dewar of AdaCore. *Intelligent Aerospace.* Retrieved from http://www.intelligent-aerospace.com/articles/2012/06/uavs-software.html

Hu, H., Ann, G., & Kulkarni, K. (2012). Detecting and resolving firewall policy anomalies. *IEEE Transactions on Dependable and Secure Computing*, *9*(3), 318-331.

Hunt, A. & Thomas, D. (1999). *The pragmatic programmer: From journeyman to master*. Boston, MA: Addison-Wesley Professional.

Identity Theft Resource Center. (n.d.). Annual number of data breaches and exposed records in the United States from 2005 to 2017 (in millions). *Statista.* Retrieved from https://www.statista.com/statistics/273550/data-breaches-recorded-in-the-united-states-by-number-of-breaches-and-records-exposed/

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

Kannan. K., & Telang, R. (2005). Market for software vulnerabilities? Think again. *Management Science*, *51*(5), 726-740.

Kannan, K., Rees, J., & Sridhar, S. (2007). Market reactions to information security breach announcements: An empirical analysis. *International Journal of Electronic Commerce*, *12*(1), 69-91.

Kaspersky Lab. (2017). *Exploits: How great is the threat?* Retrieved from https://securelist.com/exploits-how-great-is-the-threat/78125/

Kaufman, L. M. (2009). Data security in the world of cloud computing. *IEEE Security and Privacy*, *7*(4), 61-64.

Kleinman, J. (2015). Windows XP users can kiss Chrome goodbye. Retrieved from https://www.technobuffalo.com/2015/11/12/windows-xp-users-can-kiss-chrome-goodbye/

Koerner, B. I. (2016). Inside the cyberattack that shocked the US Government. *Wired*. Retrieved from https://www.wired.com/2016/10/inside-cyberattack-shocked-us-government/

Krazit, T. (2016). Employees are the weakest link in computer security. *Fortune*. Retrieved from http://fortune.com/2016/06/20/employees-computer-security/.

Levy, E. (2000). Wide open source. *SecurityFocus*. Retrieved from http://www.securityfocus.com/news/19

Maiguy, M. (2013). Fixing perverse incentives in software development. *Agile Zone*. Retrieved from https://dzone.com/papers/fixing-perverse-incentives

Mehmood, Y. Habiba, U., Shibli, M. A., & Masood, R. (2013). Intrusion detection system in cloud computing: Challenges and opportunities. In *Proceedings of the National Conference on Information Assurance.*

Meyer, A. (2017). Leaked exploits and hacking tools enable the surge of cyber attacks in 2017. *BetaNews.com*. Retrieved from https://betanews.com/2017/08/09/leaked-exploits-&-hacking-tools-enable-the-surge-of-cyber-attacks-in-2017/

McAllister, N. (2012). Skype: Nearly half of adults don't install software updates. *The Register.* Retrieved from http://www.theregister.co.uk/2012/07/23/skype_software_update_survey

Miniwatts Marketing Group. (2018). *Internet world stats*. Retrieved from https://www.internetworldstats.com/stats.htm

Moore, T. (2010). The economics of cybersecurity: Principles and policy options. *The International Journal of Critical Infrastructure Protection*, *393*(4), 103-117.

Moore, T., & Anderson, R. (2012). Internet security. In M Peitz & J. Waldfogel (Eds.), *The Oxford handbook of the digital economy*. Oxford, UK: Oxford University Press.

Netapplications.com. (2017). *Operating system market share*. Retrieved from https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0)

Neuman, S. (2018). Intel acknowledges chip-level security vulnerability in processors. *NPR*. Retrieved from https://www.npr.org/sections/thetwo-way/2018/01/04/575573411/intel-acknowledges-chip-level-security-vulnerability-in-processors

Nir, D., Tyszberowicz, S., & Yehudai, A. (2008). Locating regression bugs. In K. Yorav (Ed.), *Hardware and software: Verification and testing* (LNCS vol. 4899, pp. 218-234). Berlin: Springer.

NIST. (n.d.). Search Vulnerability Database. Retrieved from https://nvd.nist.gov/vuln/search

Pasqualetti, F., Dorfler, F., & Bullo, F. (2013). Attack detection and identification in cyber-physical systems. *IEEE Transactions on Automatic Control*, *58*(11), 2715-2729.

Prince, T. (2017). When it comes to cybersecurity, employees are weakest. *Las Vegas Review*. Retrieved from https://www.reviewjournal.com/business/when-it-comes-to-cybersecurity-employees-are-weakest-link

Privacy Rights Clearinghouse. (n.d.). *Data breaches*. Retrieved from https://www.privacyrights.org/data-breaches

Radack, S. (2012). Conducting information security-related risk assessments: Updated guidelines for comprehensive risk management programs. *NIST*. Retrieved from https://www.nist.gov/publications/conducting-security-related-risk-assessments-updated-guidelines-comprehensive-risk

Raymond, E. S. (2001). The cathedral & the bazaar. Retrieved from http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/

Redmiles, E. (2017). Why installing software updates makes us WannaCry. *Scientific American*. Retrieved from https://www.scientificamerican.com/article/why-installing-software-updates-makes-us-wannacry/

Roberts, P. F. (2015). Glitches to riches: The hackers who make a killing off software flaws. *The Christian Science Monitor*. Retrieved from https://www.csmonitor.com/World/Passcode/2015/1030/Glitches-to-riches-The-hackers-who-make-a-killing-off-software-flaws

Romanosky, S., Telang, R., & Acquisti, A. (2011). Do data breach disclosure laws reduce identity theft? *Journal of Policy Analysis and Management*, *30*(2), 256-286.

Kissel, R., Stine, K., Scholl, M., Rossman, H., Fahlsing, J., & Gulick, J. (2008). Security considerations in the system development life cycle. *National Institute of Standards and Technology*. Retrieved from https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-64r2.pdf

Korolov, M. (2018). Open source software security challenges persist. *CSO*. Retrieved from https://www.csoonline.com/article/3157377/application-development/open-source-software-security-challenges-persist.html

HeinOnline. (1998). *Restatement (Second) of Torts, Section 402A.* Retrieved from https://home.heinonline.org/titles/American-Law-Institute-Library/Restatement-Second-Torts/

Reuters Staff. (2017). U.S. blames North Korea for “WannaCry” cyber attack. *Reuters*. Retrieved from https://www.reuters.com/article/us-usa-cyber-northkorea/u-s-blames-north-korea-for-wannacry-cyber-attack-idUSKBN1ED00Q

Rustad, M. L., & Koenig, T. H. (2005). The tort of negligent enablement of cybercrime. *Berkeley Technology Law Journal*, *20*(4), 1553-1611.

Schryen, G. (2011). Is open source security a myth? *Communications of the ACM*, *54*(5), 130-140.

Sen, R., & Borle, S. (2015). Estimating the contextual risk of data breach: An empirical approach. *Journal of Management Information Systems*, *32*(2), 314-341.

Sen, R., & Heim, G. R. (2016). Managing enterprise risks of technological systems: An exploratory empirical analysis of vulnerability characteristics as drivers of exploit publication. *Decision Sciences*, *47*(6), 1073-1102.

Shackelford, S. (2009). From nuclear war to net war: Analogizing cyber attacks in international law. *Berkley Journal of International Law*, 2*5*(3), 191-251.

Schmidt, D. C., & White, J. (2017). Why don’t big companies keep their computer systems up-to-date? *The Conversation*. Retrieved from http://theconversation.com/why-dont-big-companies-keep-their-computer-systems-up-to-date-84250

Shukla, S. K. (2016). Cyber security of cyber physical systems: Cyber threats and defense of critical infrastructures. In *Proceedings of the 29th International Conference on VLSI Design.*

Somers, J. (2017). The coming software apocalypse. *The Atlantic*. Retrieved from https://www.theatlantic.com/technology/archive/2017/09/saving-the-world-from-code/540393/

Sonatype. (2016). *2016 state of the software supply chain*. Retrieved from https://cdn2.hubspot.net/hubfs/1958393/SSC/2016_State_of_the_Software_Supply_Chain_Report.pdf

Stanley, M. (2017). *Humans: The weakest link in cybersecurity.* Retrieved from https://www.foursys.co.uk/pages/paper/humans-the-weakest-link-in-cybersecurity

Tedre, M. (2015). *The science of computing—shaping a discipline.* New York, NY: CRC Press.

Telang, R., & Wattal, S. (2007). Impact of software vulnerability announcements on the market value of software vendors—an empirical investigation. *IEEE Transactions on Software Engineering*, *33*(8), 544-557.

U.S. Department of Energy. (n.d.). *Synchrophasor applications in transmission systems.* Retrieved from https://www.smartgrid.gov/recovery_act/program_impacts/applications_synchrophasor_technology.html

Vishwanath, A. (2016). Why do people fall victim to social engineering attacks? *The Conversation*. Retrieved from https://theconversation.com/cybersecuritys-weakest-link-humans-57455

Werlinger, R., Hawkey, K., Muldner, K., Jaferian, P., & Beznosov, K. (2008). The challenges of using an intrusion detection system: Is it worth the effort? In *Proceedings of the Symposium on Usable Privacy and Security*.

Weyuker, E. (1988). Evaluating software complexity measures. *IEEE Transactions Software Engineering*, *14*(9), 1357-1365.

Wolff, J. (2015). Ignoring the warning signs—users don't understand alerts about insecure browsing. Can we fix that? *Slate.* Retrieved from http://www.slate.com/articles/technology/future_tense/2015/02/ssl_warnings_users_ignore_them_can_we_fix_that.html

About the Authors

Ravi Sen is an Associate Professor at Mays Business School, Texas A&M. He received his PhD (Business Administration) in 2003 from University of Illinois at Urbana-Champaign. His research interests include cybersecurity, open source software, and economics of electronic commerce. He has published in the *Journal of management Information Systems* (*JMIS*), *Decision Support Systems* (*DSS), Decision Sciences* (*DS*), *International Journal of Electronic Commerce* (*IJEC*), *Communications of AIS* (*CAIS), Electronic Markets* (*EM*), and *Journal of Electronic Commerce Research* (*JECR*).