Characterizing and Modeling Patching Practices of Industrial Control Systems

BRANDON WANG, The University of Iowa, USA
XIAOYE LI, The University of Iowa, USA
LEANDRO P. DE AGUIAR, Siemens Corporation, USA
DANIEL S. MENASCHE, Federal University of Rio de Janeiro (UFRJ), Brazil
ZUBAIR SHAFIQ, The University of Iowa, USA

Industrial Control Systems (ICS) are widely deployed in mission critical infrastructures such as manufacturing, energy, and transportation. The mission critical nature of ICS devices poses important security challenges for ICS vendors and asset owners. In particular, the patching of ICS devices is usually deferred to scheduled production outages so as to prevent potential operational disruption of critical systems. Unfortunately, anecdotal evidence suggests that ICS devices are riddled with security vulnerabilities that are not patched in a timely manner, which leaves them vulnerable to exploitation by hackers, nation states, and hacktivist organizations.

In this paper, we present the results from our longitudinal measurement and characterization study of ICS patching behavior. Our study is based on IP scan data collected from Shodan over the duration of three years for more than 500 known industrial ICS protocols and products. Our longitudinal measurements reveal the impact of vulnerability disclosures on ICS patching. Our analysis of more than 100 thousand Internet-exposed ICS devices reveals that about 50% upgrade to newer patched versions within 60 days of a vulnerability disclosure. Based on our measurement and analysis, we further propose a variation of the Bass model to forecast the patching behavior of ICS devices. The evaluation shows that our proposed models have comparable prediction accuracy when contrasted against traditional ARIMA timeseries forecasting models, while requiring less parameters and being amenable to direct physical interpretation.

Additional Key Words and Phrases: Industrial Control Systems (ICS); Shodan; Vulnerability Patching

ACM Reference format:
Brandon Wang, Xiaoye Li, Leandro P. de Aguiar, Daniel S. Menasche, and Zubair Shafiq. 2017. Characterizing and Modeling Patching Practices of Industrial Control Systems. *Proc. ACM Meas. Anal. Comput. Syst.* 1, 1, Article 1 (January 2017), 23 pages.
DOI: http://dx.doi.org/10.1145/3084455

1 INTRODUCTION

Background and Motivation. Industrial Control Systems (ICS) are increasingly being used for a wide range of applications such as industrial automation, utility monitoring, and asset tracking [12, 34]. As any piece of software, ICS products are subject to bugs and vulnerabilities. However, unlike ordinary systems, ICS devices pose a unique set of challenges. Bugs and vulnerabilities in ICS software modules can cause serious consequences but frequent patching of such systems may lead to intolerable unavailability of mission critical infrastructures.

DOI: http://dx.doi.org/10.1145/3084455

Unpatched vulnerabilities represent one of the most likely attack vectors to the critical infrastructure in the U.S. and beyond [30] as publicly-disclosed vulnerabilities are heavily targeted by attackers. The critical manufacturing sector, for instance, takes an average of 51 days since disclosure until released patches get installed [47]. There are a number of reasons why critical systems such as PLCs (Programmable Logic Controllers), RTUs (Remote Terminal Units), SCADA (Supervisory Control And Data Acquisition) systems, and HMIs (Human Machine Interfaces) is typically not patched immediately after the vulnerability disclosure and patch release by ICS vendors. For instance, vulnerability patches have to be exhaustively tested as a general rule both by the vendor and by the asset owner. In addition, some patches require a complete system reboot, which might have to be synchronized with plant maintenance schedules where a production outage is already expected.

There are several challenges behind ICS patch management. The basic tradeoff consists of balancing between security needs and availability. Understanding current practices is key to parameterize models to assess risk scores. As the space of potential patching policies is vast, the knowledge of common practices can help to guide *what if* analysis [45]. A global perspective on the distribution of software versions among different devices assists asset owners to set their patching priorities. It is well-known that nowadays the whole IPv4 space can be scanned in a few hours using off-the-shelf resources [10, 42]. Therefore, when a vulnerability is weaponized, even if most devices are patched, the ones that are Internet-exposed and remain vulnerable may become likely targets. In some cases, an attacker may not even need Internet access to explore vulnerable devices, as insider and privilege misuse of resources are common causes of attacks [47]. A better understanding of patching practices can also provide insights into the usage of ICS devices. An ICS device which is patched is likely to be active. Then, the number of recently patched devices is likely a conservative estimate of active ICS device population.

Limitations of Prior Art. Although there have been a number of reports indicating that ICS patching behavior is erratic [20], such reports are typically conducted under restrictive non-disclosure agreements, and the methodology and data used to obtain the results is not generally available to the scientific community and the general public. Our goal, in contrast, is to analyze the ICS ecosystem using a reproducible methodology and publicly available data. This way, we envision that our results can be reproduced and extended by the scientific community, which is a fundamental step towards a better understanding of the security landscape of ICS devices.

Key Findings. We summarize our proposed approach and key findings as follows.

- First, we conduct a longitudinal measurement study to systematically analyze ICS patching behavior. We analyze 3 years worth of IPv4 scanning data from Shodan [28] for more than 500 known ICS protocols and products. We conduct a longitudinal study to analyze the impact of vulnerability disclosures on ICS device patching. *Our key insight consists of cross correlating the reports provided by Shodan with public data available in other sites such as the National Vulnerability Database (NVD) and the websites of ICS device vendors.* By jointly analyzing these data, we are able not only to assess whether the disclosure of vulnerabilities is correlated with patch application, but also if and when different features associated to the disclosed vulnerabilities, such as their Common Vulnerability Scoring System (CVSS) scores, impact patching behavior. Our analysis of more than 100 thousand ICS devices reveals that about 50% upgrade to newer patched versions within 60 days of vulnerability disclosure.
- Second, we propose a variation of the Bass model to capture and forecast the dynamics of ICS population and patching behavior. Our results show that the Bass model provides comparable accuracy as traditional ARIMA based time series prediction models, while requiring less parameters whose meaning is amenable to direct physical interpretation. Our population forecasting models can be helpful for asset owners and vendors to inform strategic patch management policies for ICS devices.

Paper Organization. The remainder of this paper is organized as follows. In the next section, we introduce basic concepts and terminology associated to the vulnerability lifecycle and to the vulnerability databases considered

in this work. Then, we present the proposed data collection methodology in Section 3. In Section 4, we report our measurement and analysis results. In Section 5, we present our proposed models to forecast ICS patching behavior. Section 6 discusses the limitations of our method and findings. Section 7 compares and contrasts our work against prior literature. Finally, we conclude in Section 8.

2 BACKGROUND

Next, we introduce background on the lifecycle of vulnerabilities and briefly describe the public vulnerability database considered in this work.

2.1 Vulnerability Lifecycle

A *vulnerability* in a computer system is a weakness that allows a hacker to exploit and attack the system. The lifecycle of a vulnerability is marked by three major events: discovery, disclosure, and patch. The lifecycle begins when a vulnerability is discovered by the vendor, hackers, or third-party institutions. The next event is the public disclosure by the vendor, security researchers, or third-party institutions. The time duration between the discovery and disclosure is called *black risk*. During this time period, only a closed group of people know about the vulnerability's existence. The next event is the patch release by the vendor. The time duration between vulnerability disclosure and patch release date is called *gray risk*. However, even when the vendor releases the patch, not all users will install it immediately. The time period between the patch date and the date when all users install the patch is called *white risk*. The lifecycle of a vulnerability ends when all users install the patch. Note that hackers can exploit the vulnerability anytime during the lifecycle.

2.2 Vulnerability Database

The National Vulnerability Database (NVD) [37], operated by the National Institute of Standards and Technology (NIST), is the one of the most comprehensive open database of vulnerabilities. At the time of writing, there are 81,292 vulnerabilities listed on the NVD. Each vulnerability has an ID given by the Common Vulnerability and Exposures Identifier (CVE-ID). For example, a vulnerability with an ID of CVE-2016-0001 was the first vulnerability to be published in 2016. For each CVE-ID entry, NVD contains the disclosure date, source information, brief overview of the vulnerability, Common Vulnerability Scoring System (CVSS) score that measures the risk of the vulnerability, and a list of all affected products. The overall CVSS score is quantified in the range of $[0, 10]$ with 0 being the least severe and 10 being the most severe. CVSS scores are computed from base, temporal and environmental metric groups. For base metrics, exploitability metrics of access vector (local, adjacent network, network), access complexity (low, medium, high), and authentication (none, single, multiple) and impact metrics of confidentiality (none, partial, complete), integrity (none, partial, complete), and availability (none, partial, complete) are used. For temporal metrics, exploitability (not defined, unproven, proof of concept, functional, high), remediation (not defined, official, temporary, workaround, unavailable), and report confidence (not defined, unconfirmed, uncorroborated, confirmed) are used. For environment metrics, general modifiers of collateral damage potential (none, low, low-medium, medium-high, high) and target distribution (not defined, none, low, medium, high) are used, as well as impact modifiers of confidentiality, integrity and availability requirements [33].

3 DATA

We rely on a popular device search engine called Shodan [28] to measure and analyze ICS devices. Shodan uses a geographically distributed set of crawlers (USA, China, Iceland, France, Taiwan, Vietnam, Romania, and Czech Republic) to scan Internet-connected devices. To ensure uniform scanning, Shodan's crawlers generate a random pair of IPv4 address and port (from a subset of ports).[1] The crawlers then scan the IP address and port pair

[1]Note that IPv6 space is too large to be reasonably covered by random scanning. At the time of writing, Shodan does not scan the IPv6 space.

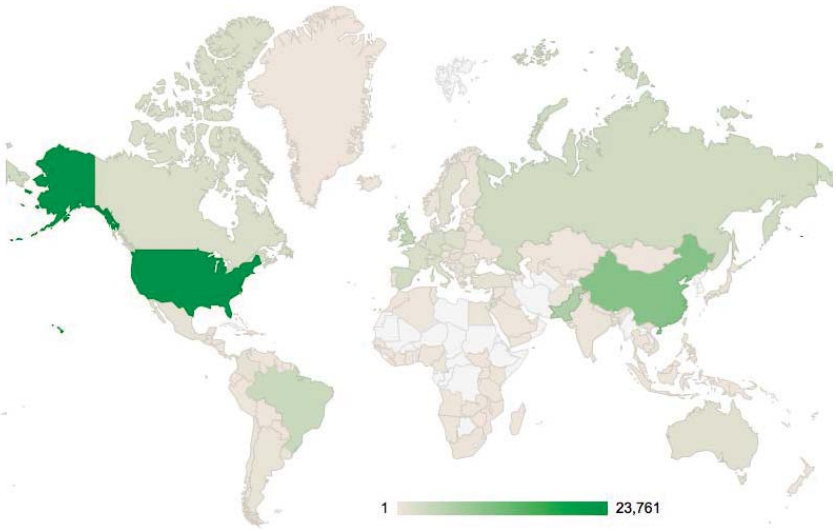


Fig. 1. World map of the global distribution of ICS devices. We note that most ICS devices are located in USA.

and gather relevant header information such as protocol/service type (HTTP, FTP, ICCP, DNP3, BACNet, etc.) and software/firmware version. Shodan provides parsed metadata for the following protocols/services: HTTP, HTTPS, FTP, SFTP, Telnet, SSH, SNMP, SQL, ICCP, MODBUS, DNP3, Ethernet, and BACNet. It is noteworthy that Shodan's scans may miss uncommon protocols/services and well-known protocols/services hosted on uncommon ports.

We searched Shodan for relevant search terms to analyze mission critical ICS devices. We identified relevant search filter terms from the project SHINE report and recent work on ICS devices [12, 34, 40]. Our list of search filter terms included popular ICS device manufacturers, products, and protocols such as "Siemens S7", "3s-Smart Software Solutions", and "VxWorks". We used the `Shodan.search` API method to retrieve the list of devices for each of the search filter terms. From Shodan, we identified a total of 121,078 matching ICS devices and 514 products. In this paper, we use the terms *products* and *software modules* interchangeably.

A device may make use of multiple software modules. The presence of certain modules at a given device, such as the Siemens SIMATIC S7-1200 module, is indicative of the nature of that specific ICS device. The presence of other modules, such as MySQL, for instance, may not provide much information about the kind of ICS device being considered. In this paper we consider both ICS and non-ICS specific software modules. Note, however, that irrespective of the software module under consideration, all the analysis presented in this work focuses on ICS devices, filtered using the terms from the project SHINE as described earlier.

Prior work has reported that ICS honeypots are increasingly deployed on the Internet [39, 43]. To ensure that our analysis represents real-world ICS devices, we aim to identify and filter suspected honeypots. There are several characteristics that are used to identify honeypots. First, many honeypots use default or randomly generated device/firmware serial numbers. For example, several Siemens S7 devices use the same serial number of 88111222. Second, honeypots are commonly hosted on popular cloud services such as Amazon EC2 and Digital Ocean. Shodan uses these criteria to quantify *honeyscore*, which represents the likelihood ($\in [0,1]$) that a device is a honeypot. Using honeyscore, we filtered 1,174 ICS devices with the likelihood of honeypot being more than 0.5. We also filtered 19,138 duplicate IP addresses.

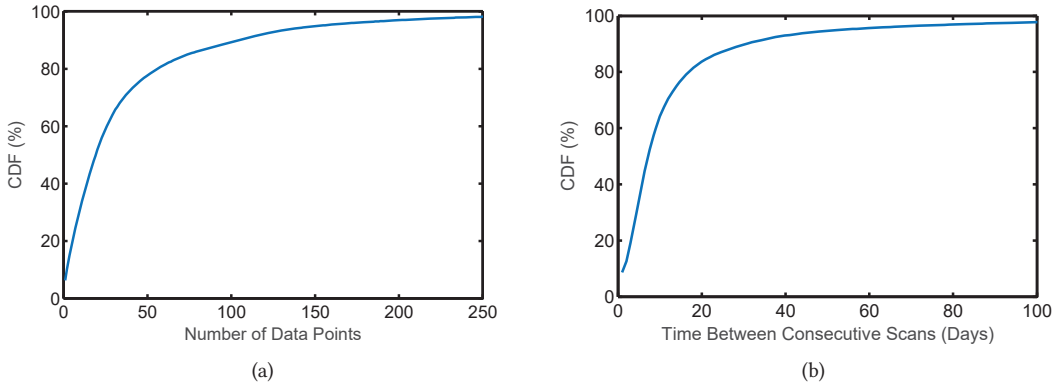


Fig. 2. (a) CDF of historical scan count of ICS devices in our data. We collected at least 10 scans for more than 90% of the considered ICS devices. (b) CDF of average time interval between consecutive scans by Shodan. The median of the time between scans is roughly 1 week.

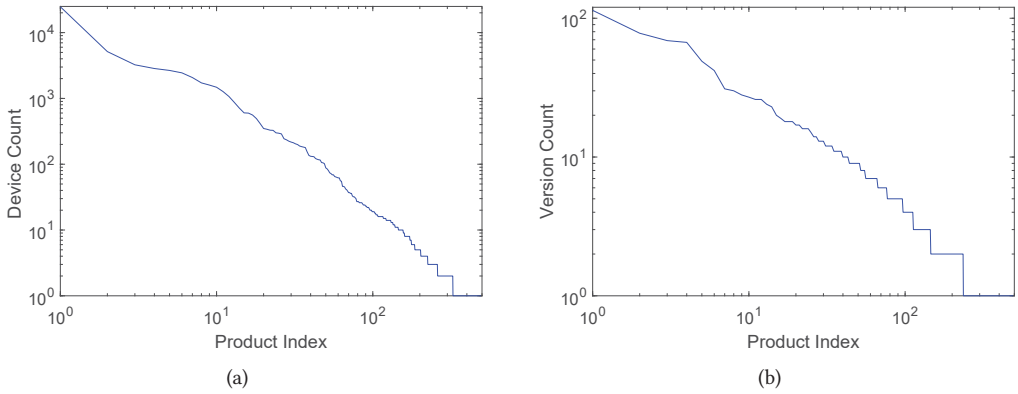


Fig. 3. (a) Distribution of most popular products. The most popular product (software module) is VxWorks ftpd with 24,711 devices. (b) Distribution of version count for different products. The products with most distinct versions are MySQL with 114 versions and MikroTik router with 78 versions.

The filtered data set contains 100,766 ICS devices. We then localize IP addresses of ICS devices using the publicly available MaxMind IP geolocation database. Figure 1 plots the distribution of the devices around the world. The top five countries with the most ICS devices in our data are the United States, China, Korea, Pakistan, and the United Kingdom. We note that USA has the most ICS devices, with a total of 23,761 devices. The other countries have less: China with about 10,000, Korea with 7,000, Pakistan with 6,000, and UK with 4,000.

We next retrieve historical scan data for each ICS device from Shodan using the `api.host` method. Shodan periodically scans the IP space and records the historical scan data. We need the historical data for longitudinal analysis of ICS device patching. The number of historical scan data points vary across ICS devices. Figure 2(a) plots the distribution of historical scan count for ICS devices in our data. We note that more than 90% of ICS devices have at least 10 scans and 5% of the devices have at least 150 scan data points. Figure 2(b) plots the

distribution of average time difference between consecutive scans by Shodan. We note that the median time difference between consecutive scans is around one week.

We parsed data retrieved from Shodan to identify product names and different version numbers for each of the products. For some devices, we could not find product names or version numbers. We were able to identify product names and version numbers for more than 64,072 devices belonging to about 500 different ICS products. Figure 3(a) plots the distribution of different ICS products in our data. The x-axis represents product index sorted in the descending order of number of matching devices. The most popular product in our data is VxWorks ftpd with 24,711 devices. Each ICS product contains one or more distinct versions. We analyzed each product and compiled a list of different product versions. Figure 3(b) plots the distribution of number of versions for different products. The x-axis represents product index sorted in the descending order of number of distinct versions. The products with most distinct versions are MySQL with 114 versions and MikroTik router with 78 versions.

4 MEASUREMENT AND ANALYSIS

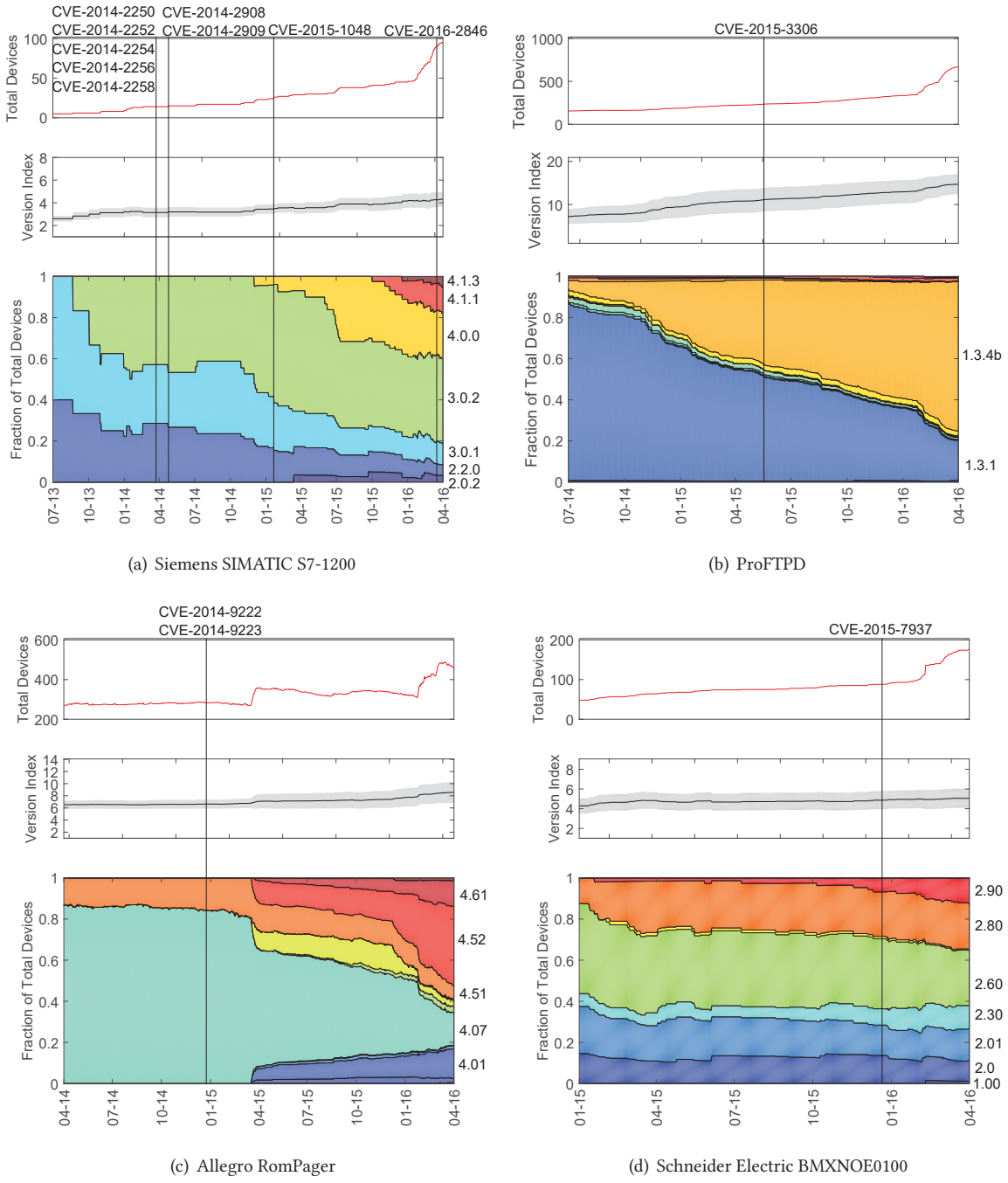
Next, we report our measurement results. We start with a longitudinal analysis of patching practices (Section 4.1), followed by an assessment of the impact of vulnerability disclosures on patching behavior (Section 4.2).

4.1 Longitudinal Analysis

We first conduct longitudinal analysis of scan data for ICS devices to understand their patching behavior. Using the detailed information about vendor name, product name, and version number, Figure 4 illustrates the temporal dynamics of different product versions for popular ICS products in our data. Each subfigure contains three plots for device count (top), version index (middle), and fractional area (bottom). The top plot represents the time series of total devices in our data. The middle plot represents the time series of an intuitive metric, *version index*, to analyze commonly used version numbers for each product. The oldest version is assigned the index of 1 and subsequent versions are assigned indexes incrementally. The solid line represents the average version index and the gray area surrounding the solid line represents the standard deviation of version index for all devices in our data. The bottom plot represents the time series of fraction of different versions in our data. Each color in the area plot represents a distinct version number. Version numbers are sorted in ascending order. Note that we do not label versions which constitute a small fraction of total devices in our data.

Figure 4 reports results of our cross correlation analysis between data from Shodan and NVD for four software modules: Siemens SIMATIC S7-1200, ProFTPD, Allegro RomPager, and Schneider Electric BMXNOE0100. We select these four modules because they represent diverse ICS patching behavior. Siemens SIMATIC S7-1200 is a controller used for industrial automation. ProFTPD is an open source FTP server. Allegro RomPager is an embedded web server toolkit. Schneider Electric BMXNOE0100 is an Ethernet module used in energy management and automation solutions. Note that whereas Siemens and Schneider's modules are typically installed at ICS devices, Allegro RomPager is commonly present at embedded systems and ProFTP is a generic software module. In all cases, we restrict our measurements to ICS devices that use such modules. Recall from Section 3 that this is accomplished by filtering devices whose descriptions (as provided by their owners or manufacturers) included ICS terms curated by project SHINE.

Figure 4(a) illustrates the patching behavior for Siemens SIMATIC S7-1200 over the duration of three years. We note that the number of devices monotonically increases over time, with a sharp increase in the last two months of our data collection. Before March 2014, we note that the share of version 3.0.2 devices increases while that of two older versions (2.2.0 and 3.0.1) decreases. This is followed by five vulnerability disclosures in March 2014 (CVSS scores ranging between 6.1-8.3) and two more vulnerability disclosures in April 2014 (CVSS scores ranging between 4.3, 7.5). The disclosed vulnerabilities mostly impact the integrated web server on Siemens SIMATIC S7-1200 devices by allowing hackers to launch denial of service and Cross-Site Scripting (XSS) attacks. Version 3.0.2 and earlier versions were affected by these vulnerabilities. Siemens released version 4.0.0 in March 2014 to



(a) Siemens SIMATIC S7-1200

(b) ProFTPD

(c) Allegro RomPager

(d) Schneider Electric BMXNOE0100

Fig. 4. Fractional share of different versions over time for popular ICS products in our data.

patch these vulnerabilities. However, it is not until late 2014 when version 4.0.0 starts to appear in our scan. These findings indicate a white risk period of at least 8 months between patch release and earliest patch installation dates. In January 2015, CVE-2015-1048 disclosed a new redirect vulnerability in the integrated web server on Siemens SIMATIC S7-1200 4.0.0. Siemens released version 4.1 in December 2014, but the fraction of users utilizing version 4.0.0 continues to increase throughout 2015 (including newly discovered devices). This could potentially denote a lack of security awareness during the first product installation (assuming newly installed devices come up-to-date from factory but might remain boxed without updates for an extended period of time). Version 4.1 finally starts to appear around October 2015, again confirming a white risk period around 8 to 10 months.

Figure 4(b) illustrates the patching behavior for ProFTPD over the duration of two years. We again note that the number of devices monotonically increases over time. We find two dominant versions in our data: version 1.3.1 declines and version 1.3.4b rises over time. The increase in share of version 1.3.4b is due to users upgrading from version 1.3.1 to version 1.3.4b. As a result, we observe a significant increase in the mean version index. CVE-2015-3306 (CVSS score of 10.0) was disclosed in May 2015 and it affected version 1.3.5 by allowing hackers to read/write to arbitrary files. We note approximately 2% ProFTPD version 1.3.5 devices in our data and their share does not substantially change over time. This vulnerability was patched in version 1.3.5a, which was released later in May 2015. We first observed version 1.3.5a in November 2015 in our data but its maximum share rose up to less than 0.5%.

Figure 4(c) illustrates the patching behavior for Allegro RomPager over the duration of two years. We note a non-monotonic change in the number of devices over time. More than 80% of devices used version 4.07 till April 2015. Two vulnerability disclosures (CVE-2014-9222 and CVE-2014-9223 with CVSS scores of 10.0) in December 2014 affected 4.07 and previous versions. These vulnerabilities allowed remote attackers to gain privileges and possibly execute arbitrary code. Allegro RomPager 4.34 and later versions released in mid 2015 fixed these vulnerabilities. We observe a sharp increase in the share of newer versions soon after the patch release, which corresponds to version 4.07 devices being patched to the newer versions.

Figure 4(d) illustrates the patching behavior for Schneider Electric's BMXNOE0100 product over the duration of one year. We note that the number of devices monotonically increases over time. There is a significant increase in the share of version 2.80 in the first quarter of 2015. Since only insignificant changes in the number of devices are prevalent, this indicates users from older versions most likely updated to version 2.80. CVE-2015-7937 (CVSS score of 10.0) was disclosed in December 2015 and it affected all versions prior to 3.10. The vulnerability allowed attackers to remotely execute arbitrary code. Despite that, we observe an increase in share of the vulnerable version 2.90 mainly because of an increase in the number of new devices. This observation confirms our previous finding about limited awareness for the need of patching new devices before connecting them to the network.

4.2 Impact Analysis of Vulnerability Disclosures

Next, our goal is to study the impact of vulnerability disclosures on ICS patching practices. To this end, we first analyze the impact of the age of vulnerabilities on the average version index of ICS products. Figure 5(a) plots the CDF of the change in average version index after vulnerability disclosure for the products with at least one relevant vulnerability disclosure. We condition the distributions on different time durations of 7 days, 30 days, and 60 days. We observe that approximately 20% ICS products exhibit an increase in average version index 7 days after vulnerability disclosure. We observe that approximately 40% ICS products exhibit an increase in average version index 30 days after vulnerability disclosure. We also observe that approximately 50% ICS products exhibit an increase in average version index 60 days after vulnerability disclosure. To our surprise, approximately 10-15% ICS products exhibit a decline in average version index after vulnerability. Therefore, we conclude that a substantial fraction of ICS products are slow to patch — they do not install patches up to several months after vulnerability disclosures.

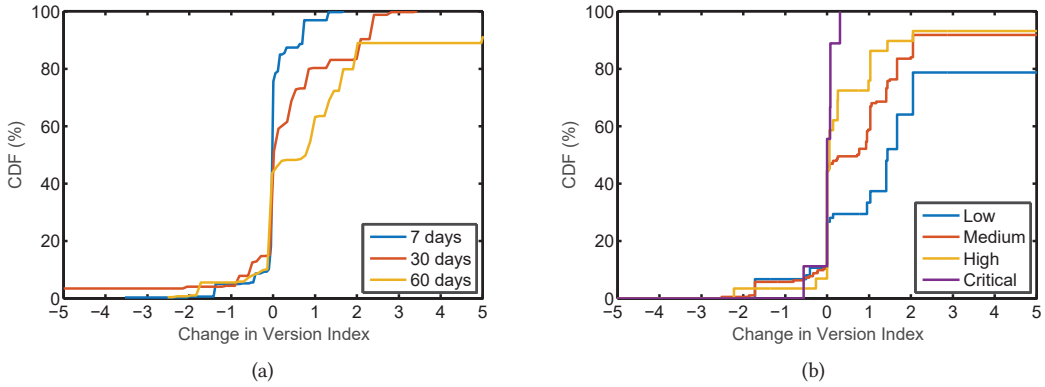


Fig. 5. Impact of vulnerability age (days since disclosure) and CVSS score (60 days since disclosure) on change in version index. (a) Approximately 20% ICS products exhibit an increase in average version index 7 days after vulnerability disclosure. (b) As a counter-intuitive finding, we note that the overall CVSS score is negatively correlated to patch deployment time.

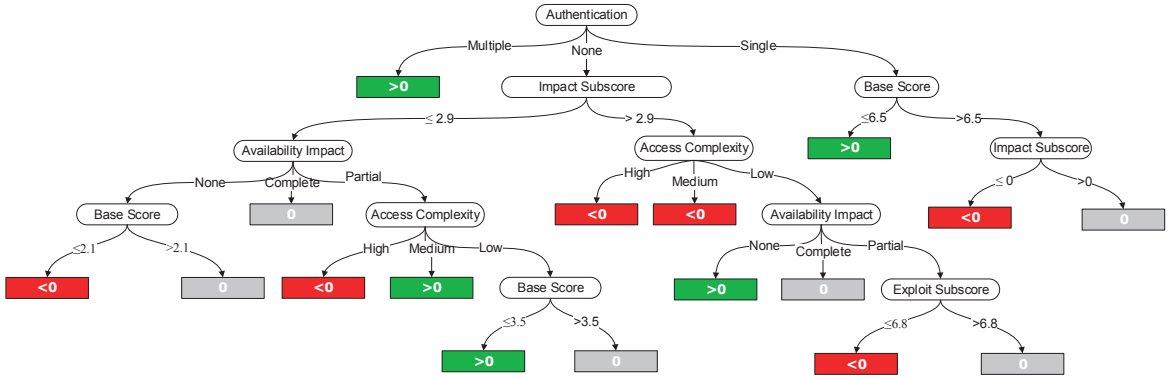


Fig. 6. Impact of vulnerability attributes (CVSS features) on change in version index. We note that lower CVSS exploitability and impact subscores are correlated with increased patch deferral times.

We next analyze the impact of vulnerability characteristics on the average version index of ICS products. We plot the change in average version index as a function of different CVSS scores in Figure 5(b). We condition the distributions on CVSS score ranges of low, medium, high, and critical severity. We observe that approximately 70% ICS products exhibit an increase in average version index 60 days after a low severity vulnerability is disclosed. We also observe that approximately 50% ICS products exhibit an increase in average version index 60 days after a medium severity vulnerability is disclosed. To our surprise, fewer ICS products exhibit average version index increases 60 days after a high or critical severity vulnerability disclosure. In fact, the 10% of ICS products that exhibit an average version increase 60 days after a critical severity vulnerability disclosure increase by less than 1 version index. Therefore, at surface, it seems that the severity of a disclosed vulnerability has little impact on the patch rate for a vast number of ICS products.

We next breakdown CVSS scores into their constituent metric groups (discussed in Section 2) and study their impact on the average version index of ICS products. To systematically quantify the impact of these vulnerability features on ICS patching, we train a C4.5 decision tree model to predict the change in version index as a function of CVSS vulnerability features. Figure 6 visualizes a pruned version of the decision tree model for positive (>0), negative (<0), and no change (0) in the average version index of ICS products, 60 days after vulnerability disclosures (same as in Figure 5(b)). The vulnerability features at the higher levels of the decision tree tend to have more distinguishing power than lower level features. The root node is the authentication requirement for vulnerability exploitation. However, it is noteworthy that the ordering of the features does not strictly determine their importance because of the interdependencies among them. For example, the authentication requirement is incorporated into the base score. Decision tree feature splits provide insights into how these features may impact ICS patching. For example, the exploitability and impact subscore splits show that lower scores are lead to a decrease in the average version index of ICS products.

In summary, Figure 5(b) presents a counter-intuitive finding indicating that the *overall CVSS score* is negatively correlated to patch deployment time. Figure 6, in contrast, shows that if we look at the *constituents of the CVSS score*, we observe more expected patterns. In particular, we note that lower CVSS exploitability and impact subscores are correlated with increased patch deferral times.

5 MODELING POPULATION DYNAMICS

In this section, we present a model to capture the patching behavior of ICS devices. The goal of the model is to predict and explain the adoption and substitution for successive software versions. As we will show, a simple variation of the Bass model, which has been previously used to describe the evolution of innovations, is able to accurately capture ICS population dynamics in our measurements.

To appreciate the applicability of a predictive model, we consider the problem of patch management. Suppose that a firm is able to infer that a significant share of the population will be protected against a given vulnerability by a certain day. Then, the firm may want to apply the patch sufficiently before that day, so as to avoid becoming an easy target of network-based malware, which typically find vulnerable devices by scanning the IP space [24, 27]. Conversely, suppose that the number of users adopting a given version of a product is predicted to remain large, and there are no known exploits against the vulnerabilities that affect that product. In addition, the CVSS score associated to the vulnerabilities is low. In that case, the firm may decide to defer the patching of its products up until a pre-scheduled convenient date. Predictive models of patching practices provide estimates about the extent at which vulnerabilities can impact the ICS ecosystem and the horizon during which weapons against existing vulnerabilities are useful. From the standpoint of security economics, such estimates are helpful to understand the incentives for exploit and patch development by hackers [1] and vendors [3], respectively.

5.1 Bass Model Primer

Next, we introduce the model proposed by Norton and Bass [35, 36] to capture the evolution of technological generations. We consider the model in its simplest form, to characterize the dynamics of the number of adopters of a given product. The model considers a sequence of N generations of a given product. The number of devices of a given generation at any point in time increases due to newcomers (exogenous adoptions) as well as due to upgrades from previous generations (endogenous adoptions). The key simplifying assumption consists of considering constant diffusion parameters, p and q, across generations of the same product. Parameters p and q determine the rate and shape of the growth curves of the number of adopters of each version of the product. Let m_i be the incremental market potential for the i-th generation. Then, the population dynamics of a given product is characterized by $N + 2$ parameters, $m_1, \ldots, m_N, p, q$ (Table 1 summarizes notation).

Let $t_{0,i}$ be the time of introduction of the i-th generation, and let t_i be the time since the introduction of the i-th generation,

$$t_i = \begin{cases} t - t_{0,i}, & \text{if } t > t_{0,i} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

The model assumes that each generation is introduced before its predecessor has been fully diffused. Therefore, some of the users that have adopted a previous version of the software will migrate to new ones, and some of the users that would have adopted a previous version will effectively only adopt a new version.

5.1.1 Single Generation. Next, let us consider the single generation Bass model. Let $F(t;p,q)$ be the fraction of users, out of the total potential number of adopters, that adopted a given product by time t. The key assumption of the model is that the rate of new adoptions of a product, $\frac{d}{dt}F(t;p,q)/(1-F(t;p,q))$, equals a constant term, p, that captures the number of new external arrivals, also referred to as *innovators*, plus a term which increases as the number of adopters increases, $qF(t;p,q)$, which captures *imitation*. $\frac{d}{dt}F(t;p,q)$ is given by

$$\frac{d}{dt}F(t;p,q) = (p + qF(t;p,q))(1 - F(t;p,q)) \tag{2}$$

The solution of the equation above is

$$F(t;p,q) = \frac{1 - e^{-(p+q)t}}{1 + (p/q)e^{-(p+q)t}} \tag{3}$$

In prior literature, different shape functions have been considered as alternatives to (3) (see [26]). In the remainder of this paper, we rely on (2) and its solution (3), and consider alternative diffusion models in Section 5.6.

5.1.2 Multiple Generations. Next, we extend the results in the previous section to multiple generations (versions) of a product. Let $\tilde{m}_i$ be the upper limit on the number of users adopting the product at the i-th generation. Then,

$$\tilde{m}_i = \sum_{j=1}^{i} m_j \tag{4}$$

Whereas m_i, $i = 1, \ldots, N$, is the incremental market potential of the i-th version of the product, $\tilde{m}_i$ is the total market potential of that version. We let M be the total market potential of all versions,

$$M = \tilde{m}_N \tag{5}$$

Variable	Description
N	number of versions
m_i	incremental market potential of version i
$\tilde{m}_i$	total market potential of version i, $i = 1, \ldots, N$
M	$= \tilde{m}_N$, total market potential of software module
p	innovation parameter
q	imitation parameter
t_i	time since introduction of i-th generation
$t_{0,i}$	time of introduction of i-th generation

Table 1. Table of notation (Bass model). A *generation* refers to a software module version.

Let $\tilde{S}_i(t)$ be the number of potential adopters of the i-th generation by time t, without accounting for the upgrades of users that migrated from version i to more recent versions. Then,

$$\tilde{S}_i(t) = \sum_{k=1}^{i} m_k \prod_{j=k}^{i} F(t_j) \tag{6}$$

By time t, $\tilde{S}_i(t)$ out of the $\tilde{m}_i$ users will have adopted version i. The k-th term in the summation in (6) corresponds to the number of users that first adopted version k, and switched to version i by time t, which occurs with probability $\prod_{j=k}^{i} F(t_j)$. The latter product captures the probability that all intermediate upgrades between versions k and i occurred before time t.

The number of adopters of the i-th generation by time t, $S_i(t)$, accounting for the upgrades of users that migrated from version i to more recent versions is given by

$$S_i(t) = \begin{cases} \tilde{S}_i(t)(1 - F(t_{i+1})), & 1 \leq i < N \\ \tilde{S}_i(t), & i = N \end{cases} \tag{7}$$

According to (7), by time t a fraction $F(t - t_{0,i+1})$ of the $\tilde{S}_i(t)$ users that had already adopted version i will have switched to more recent versions, for $i = 1, \ldots, N-1$. For the latest version N, we have $\tilde{S}_N(t) = S_N(t)$.

5.2 Illustrative Examples

Next, we consider the special case where we have two versions of a software, i.e., $N = 2$. Setting $N = 2$ corresponds to the simplest possible setup, but already allows us to appreciate some of the key features of the Bass model, and compare it against alternatives such as autoregressive integrated moving average (ARIMA). One of our aims is to assess the predictive power of the models. When $N = 2$, the number of adopters of the i-th generation by time t, $S_i(t)$, for $i = 1, 2$, is given by

$$S_1(t) = F(t_1; p, q)m_1(1 - F(t_2; p, q)) \tag{8}$$

$$S_2(t) = F(t_2; p, q)(m_2 + F(t_1; p, q)m_1) \tag{9}$$

We consider two products, ProFTPD and and Allegro RomPager, whose population dynamics are depicted in Figures 7 and 8 (see also Figures 4(b) and 4(c)). For each product, we selected two representative versions and plotted the number of devices as a function of time. Time is divided into blocks of 13 days, which yields 40 measurement points for each version of each product. Note that the instant of time zero corresponds to the day at which we observed the first instance of the considered product version, i.e., versions 1.3.1 and 4.0.7 of ProFTPD and Allegro RomPager, respectively. The first occurrence of versions 1.3.4b and 4.5.2 of the corresponding products was observed at time 10 and 20, respectively. The first 30 samples were used for the tuning of parameters, and the latest 10 were used for validating prediction accuracy. Note that the Bass model correctly captures the trends for the two products. In particular, it captures the fact that old versions tend to be replaced by new ones. This hypothesis, which is built in into the model, allows us to fit the curves using only four parameters for each product (p, q, m_1 and m_2).

We contrast the results obtained with the Bass model against those generated by an ARIMA predictive model. Recall that the first 30 samples of each time series are used as our training set, and the last 10 samples constitute the test set. For this reason, in Figures 7 and 8 ARIMA predictions are shown on times from 31 up to 40. Bass model values are shown on the whole timeline, as the Bass model serves both predictive as well as explanatory purposes.

As indicated in Figures 7 and 8, ARIMA models also provide good approximations to the observed values. However, while the Bass model requires only one parameter per version to be tracked, the ARIMA model requires three parameters per version. Note, for instance, that the best parameterization of the ARIMA model for version

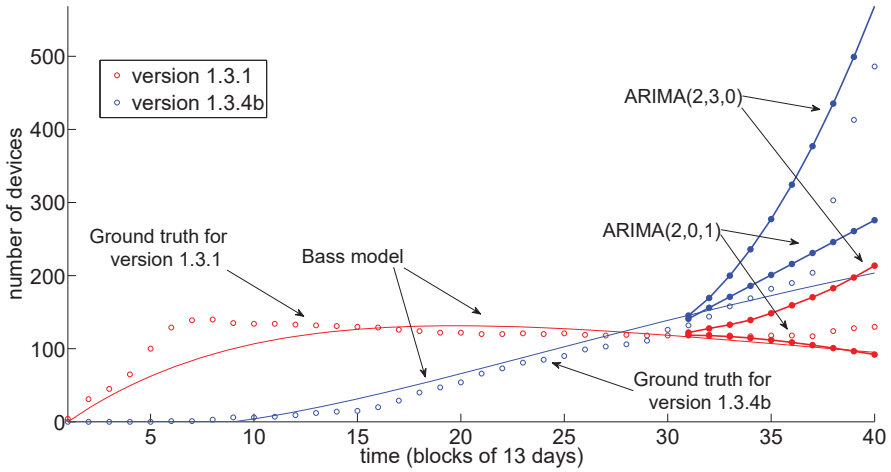


Fig. 7. Population dynamics associated to the ProFTPD software module. The predictive power of the ARIMA and Bass models are comparable, although the former requires a fine tuning of its parameters per software module version.

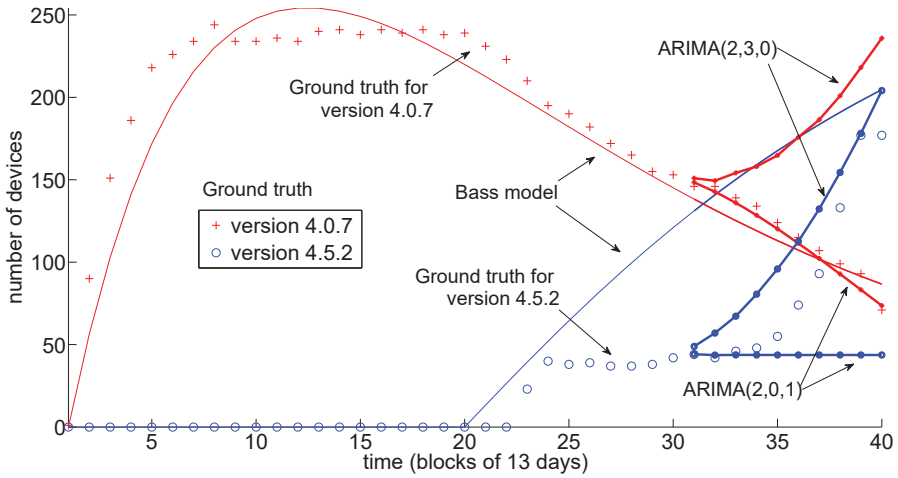


Fig. 8. Population dynamics associated to the Allegro RomPager software module. The predictive power of the ARIMA model is slightly better than the Bass model, although the former requires a fine tuning of its parameters per software module version.

4.5.2 of Allegro RomPager is given by (2,3,0). However, these parameters work very poorly for version 4.0.7 of the same software because while the curve capturing the population dynamics for the oldest version is concave, the latest is convex. The Bass model naturally captures this behavior, while providing a physical meaning to the different model parameters as described in the previous section.

5.3 Bass and ARIMA parameterization

Next, we describe the methodology to parameterize the Bass and ARIMA models. For each product, we divide the traces into forty points, and use the first thirty points for training and the last ten points for testing.

To search for the optimal sets of the Bass model parameters we used the nonlinear interior-reflective Newton method described in [7].[2] The solver must be supplied with an initial point from which a local minimum is searched for. Each initial point is an $N + 2$ dimensional vector, where the first N entries correspond to the incremental market potentials of the N versions, m_i, $i = 1, \ldots, N$, and the last two entries correspond to parameters p and q. For each product, we consider thirty initial conditions for the model parameters, chosen uniformly at random between 0 and 1000. For each initial condition, we use the solver to find a set of parameters which locally minimizes the difference between the Bass model estimates and the target values observed in our traces. We require the solution to satisfy non-negativity constraints. Then, the Bass model is evaluated using the obtained parameters, and the mean square error (MSE) between the estimates and the observed values is recorded.

To estimate the optimal sets of parameters of the ARIMA model, we use the maximum likelihood method described in [5].[3] We consider a nonseasonal linear ARIMA time series model, where the autoregressive degree P, differencing degree D, and moving average degree Q are fixed and given. The additional free parameters are estimated using the maximum likelihood method.

Note that, for each product, the parameterization of the Bass model is executed in an integrated fashion for all versions of that product. The parameterization of the ARIMA model, in contrast, is executed separately for each version of each product. This is because the Bass model naturally accounts for the interdependencies between different software versions, capturing the fact that part of the flow of users out of a version corresponds to the flow into a newer version. The ARIMA model does not capture such domain specific interdependencies.

5.4 Bass Model Fitting Results

Next, we report the results obtained by fitting the Bass model to our traces. Our goals are to (1) assess the predictive power of the model, (2) evaluate the sensitivity of the model with respect to its parameters, and (3) present and discuss the physical interpretation of the parameter values.

In Figures 9-12, we report the distributions of MSE and of selected model parameters. The dotted lines in the CDFs correspond to upper and lower bounds, considering the maximum and minimum values of the quantities of interest across the thirty runs.

Figure 9 shows the CDF of MSE of the Bass model predictions. For up to seventy percent of the products, the MSE remained below ten. It is interesting to note that the upper and lower bounds of the CDF, plotted with dotted lines, are very close to each other. This indicates that the results have a certain level of robustness against changes in the model parameters. In addition to its predictive power, the Bass model can also be used for explanatory purposes. For this reason, in Figure 9, we also report the MSE accounting for all the available data points (training and testing sets). The median of the MSE estimates equals 10 and 1 when assessing the model predictive and explanatory accuracy, respectively.

Figures 10(a) and 10(b) show the CDFs of the estimates of parameters p and q, respectively. Recall that these parameters control the rate and shape of the growth curves across generations of the same product. Note that although the MSE did not vary significantly among different runs of the model parameterization, the values of parameters p and q varied more significantly (compare the solid line with the dotted lines in Figures 10(a) and 10(b)). This observation indicates robustness of the model against changes in the parameter values. It is also worth mentioning that the values of p and q reported here are significantly larger than those previously observed for the diffusion of innovations of physical goods, where we typically have $p + q \leq 1$ [25].

[2] As implemented by the `lsqnonlin` function in MATLAB.
[3] As implemented by the `estimate` function in MATLAB.

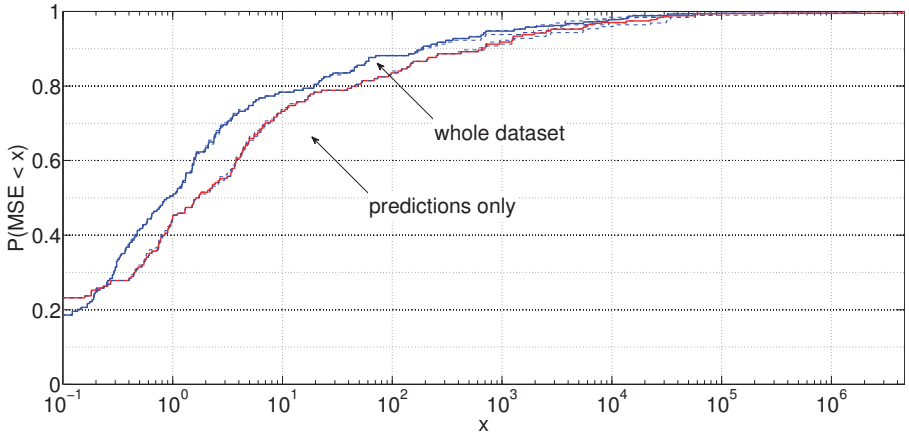


Fig. 9. Predictive accuracy of Bass model. The median of the MSE estimates equals 10 and 1 when assessing the model predictive and explanatory accuracy, respectively.

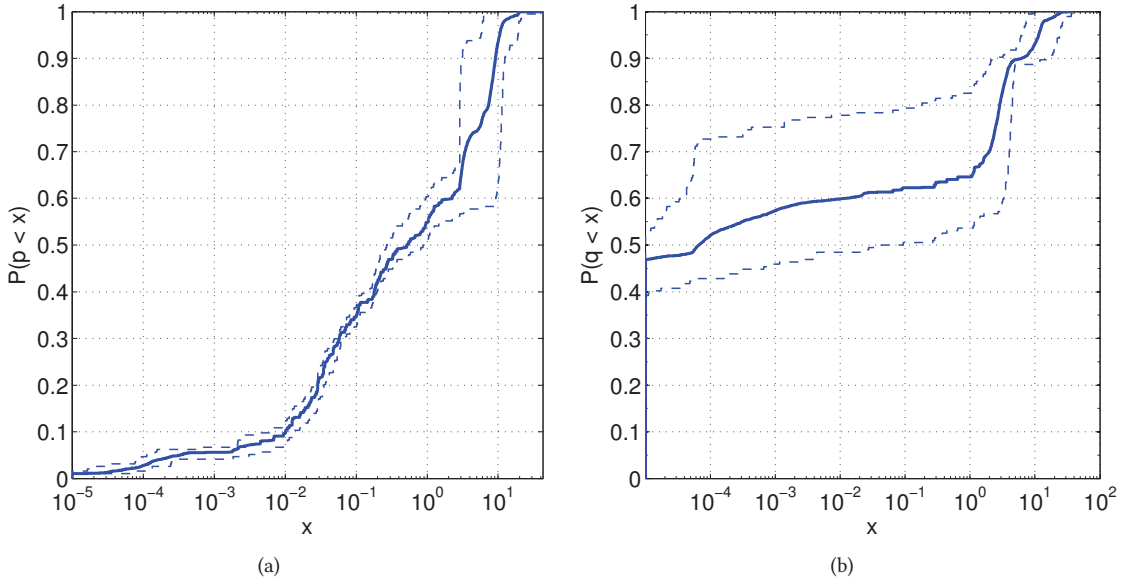


Fig. 10. CDF of Bass model parameters p and q. Note that the maximum and minimum values obtained for the optimal parameterization of p and q across 30 runs significantly varied, whereas the MSE remained roughly stable (see Figure 9), suggesting model robustness with respect to its parameters.

Whereas in Section 5.2 we presented results on specific products, in Figures 10(a) and 10(b) we report results on all observed products. Products can be clustered into groups, e.g., based on their vendors, and possibly using the parameters of the Bass model as features for the clustering algorithm. Future work consists in identifying product categories corresponding to distinct shapes of patching diffusion curves.

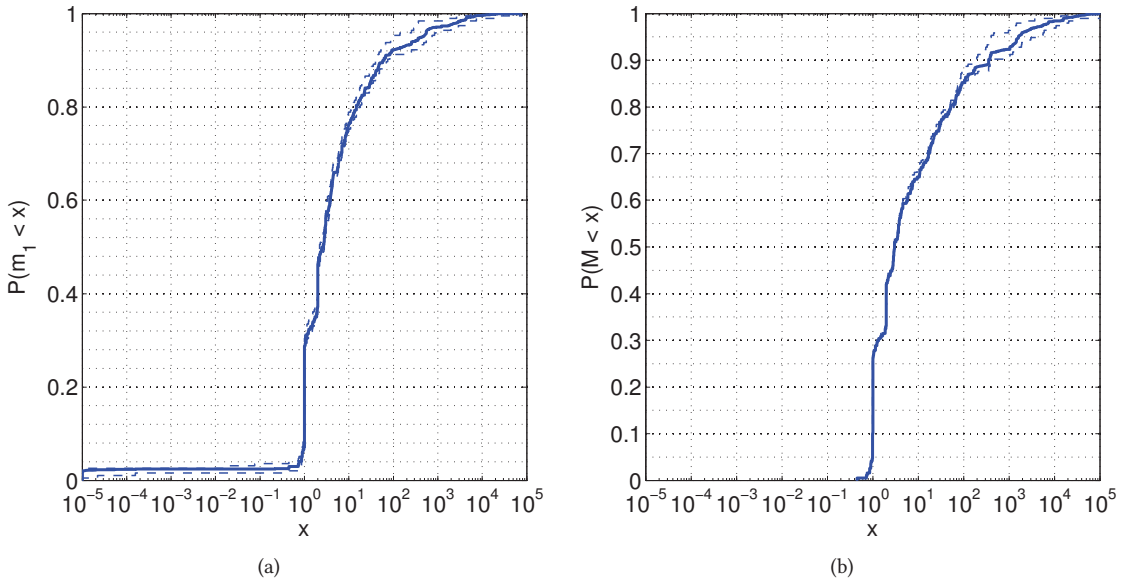


Fig. 11. CDF of Bass model parameters m_1 and M, which characterize the population potential for the first version and all versions of each product, respectively. The sampled user base was estimated to be less than a hundred devices for up to eighty percent of the modules observed in our traces (see also Figure 3).

Figures 11(a) and 11(b) show the CDFs of m_1 and M. Recall that m_1 (resp., M) is an upper limit to the number of users adopting the product at its first generation (resp., across all generations). For up to roughly eighty percent of the products we have $m_1 \leq 10$. This reflects the fact that in our traces a number of software modules were observed in tens of devices. Note that if we consider all versions of the software modules (Figure 11(b)), their corresponding sampled user base was estimated to be less than a hundred devices for up to eighty percent of the modules observed in our traces.

In summary, the Bass model predictions correspond to an MSE of less than 10 for up to 70% of the considered products. We observed that the model is robust with respect to changes in the initial conditions used to find its optimal set of parameters. The parameters of the model have a physical interpretation, and the equations are amenable to analytical treatment. In particular, the number of potential adopters of ICS products, as estimated by the model, did not surpass 1,000 users for up to 90% of the considered products. This indicates that the number of ICS products connected to the Internet is small, but non-negligible.

5.5 Contrasting the Bass Model Against ARIMA

Next, we assess the accuracy of the ARIMA time series model to capture the evolution of the number of users adopting different versions of the software modules. Figure 12 reports the MSE of the obtained results. We create a nonseasonal linear ARIMA time series model using autoregressive degree P, differencing degree D, and moving average degree Q. For each version of each software module, we vary each of the parameters P, D and Q between 0, 1, 2 and 3. As in the previous section, we divide the time series into forty intervals, and use the first thirty intervals for training and the last ten for testing.

Figure 12 reports the CDF of the average MSE across all parameterizations. It also shows the CDF of the MSE accounting for the best and worst parameterizations for each version of each software module. The average MSE follows similar trends as those obtained with the Bass model. If one is able to correctly set the ideal parameters of the ARIMA model, it produces excellent predictive power. However, the model is very sensitive to its parameters, and a wrong parameterization generates errors of the order of 10^5 for most of the ICS software modules.

We were not able to find a good parameterization of the ARIMA model that has good accuracy over all test cases. As illustrated in Section 5.2, the challenge stems from the fact that different versions of a product are associated to different trends with respect to the number of adopters. In Figures 8- 7, curves associated to older versions of a product are concave decreasing, while the ones related to newer versions are convex increasing. We envision that by leveraging such domain knowledge it may be possible to devise an algorithm to parameterize the ARIMA model with good accuracy over all test cases, similar in spirit to the algorithms discussed in [5].

It is worth noting that whereas the Bass model couples all the versions of a given software module through parameters p and q, the ARIMA model treats each version independently of all the others. For this reason, the ARIMA model requires more parameters than the Bass model, and the parameters do not have a direct physical interpretation. In addition, while the Bass model has both explanatory as well as predictive power, the ARIMA model is mostly applicable for predictive purposes. From an analytical viewpoint, to the best of our knowledge our work is the first application of the Bass model to capture the dynamics of patching behavior. Further tuning of the Bass model can increase its predictive power, and we consider first steps in that direction in the upcoming section.

5.6 Additional Diffusion Models

Next, we consider additional diffusion models that can be coupled with the Bass model [11, 26]. The two key properties of a diffusion model are its point of inflection and its symmetry. In the original diffusion model proposed by Bass (see (2)- (3)), the non-cumulative adopter distribution is maximum at a point $T^\star$, and decreases afterwards. This point corresponds to the inflection point $F^\star$ of the S-shaped curve of cumulative number of adopters given by (3). In the original model, the inflection point $F^\star$ is in the range between 0 and 0.5, and the non-cumulative

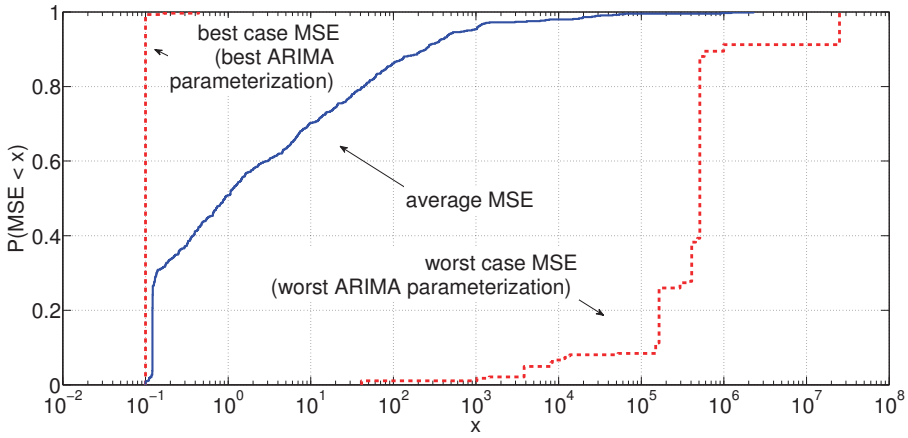


Fig. 12. Predictive accuracy of ARIMA. ARIMA requires three parameters per time series. Fine tuning the ARIMA parameters per software module version, and using the best parameterization, we obtain excellent results. However, taking the wrong parameterization yields poor accuracy. On average, the MSE is comparable to the one obtained using the Bass model.

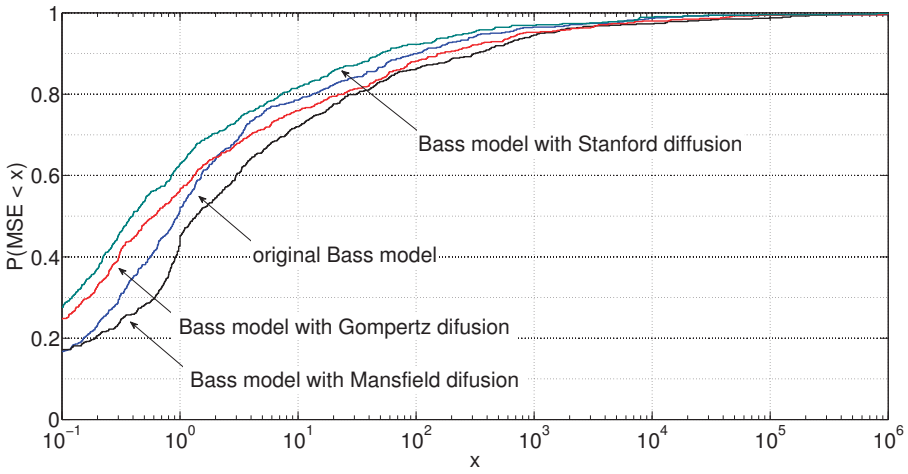


Fig. 13. Predictive accuracy of different variations of the Bass model. We note that the Stanford diffusion model produces slightly better results than the original model.

curve of adopters is symmetric with respect to time around $T^\star$, up to time $2T^\star$, and non-symmetric outside that range.

In general, the maximum diffusion and the inflection point of the cumulative number of adopters can occur at any time, and the non-cumulative curve of adopters can be symmetric or non-symmetric. For this reason, alternative diffusion models have been proposed to replace (2)- (3). We consider three alternative options, whose inflection point and symmetry properties are summarized in Table 2, obtained from [26].

As in the original Bass model, all the alternatives considered require two parameters per device. The parameters are (p, q), (c, q), (c, q) and $(T^\star, q)$ for the Bass, Gompertz, Mansfield and Stanford models, respectively. Note that whereas in previous work the constants c and $T^\star$ were assumed to be fixed and given, in our parameterization we allow them to be flexibly set in the search for the optimal set of parameters. The properties about the points of inflection and symmetry reported in Table 2, derived in previous works, refer to the constrained versions of the models.

Figure 13 illustrates how the alternative diffusion models perform in our dataset. The performance of the original Bass model was already presented in Figure 9. In Figure 13, the corresponding curve has subtle differences with respect to the one shown previously. This is because, for each device, in Figure 13 we used new sets of initial points to feed the optimizer that searches for a local minimum corresponding to the best fit of the model parameters (Section 5.4). We note that Mansfield diffusion model corresponds to the largest error, followed by the original Bass model which has competitive performance against the Gompertz diffusion. The Stanford diffusion model produced slightly better results than the original model. Interestingly, in previous works the Stanford model was used to capture energy-efficient innovations [26].

6 LIMITATIONS

Shodan provides an easy-to-access database of most Internet-connected devices, including those which are part of mission-critical infrastructure. That makes practical the task of evaluating and tracking vulnerability mitigation and patch management. We acknowledge, however, the existence of limitations while making assumptions and inferring results regarding patch management behavior. Some of these limitations of our data and analysis are presented below.

Model	Equation ($dF/dt =$)	Solution ($F =$)	Point of Inflection $F^\star$	Symmetry
Bass	$(p + qF)(1 - F)$	$\frac{1 - \exp(-(p+q)t)}{1 + \frac{q}{p}\exp(-(p+q)t)}$	0-0.5	NS
Gompertz	$qF \ln(1/F)$	$\exp(-\exp(-(c + qt)))$	0.37	NS
Mansfield	$qF(1 - F)$	$\frac{1}{1 + \exp(-(c+qt))}$	0.5	S
Stanford	$(q/t)F(1 - F)$	$1 + (T^\star/t)^q$	0-0.5	NS

Table 2. Diffusion models. NS (resp., S) refers to non-symmetric (resp., symmetric) curves of adopters.

(1) *Device tracking over time.* Shodan presents a database with devices that are difficult to individually track over a large timeframe. The IP addresses of devices might change over time due to dynamic address allocation. Furthermore, mapping devices to their corresponding IP addresses is non-trivial. For this reason, in this work we take a population dynamic perspective, accounting for the aggregate counts of the number of devices employing given software modules. The tracking of individual devices, for example using machine learning techniques, is left as subject for future work.

(2) *Unconventional services and ports.* Shodan provides mainly a search engine for service banners. It may miss uncommon services or custom protocols. Well-known protocols/services on non-standard UDP/TCP ports (as what might be typically found within this domain) might also not be discovered by Shodan.

(3) *Representative devices.* Different Internet-connected devices might not necessarily receive the same level of diligence from asset owners. The devices that receive less attention are likely connected to less critical parts of the production processes they support. For instance, a read only wireless sensor from a non-safety critical process might be less critical than an actuator in the cooling system of a thermoelectric. For this reason, generalizations of patch management efficiency based on sampled data need to be considered with caution, as most critical ICS devices are in fact connected to segregated or to "air-gapped" networks only ([41, p. 258], [2, p. 102], [29]).

(4) *Representative products.* We considered only products for which at least one update was observed, indicating that there is in fact a patch management process in place. All products that have never been patched were removed from the database. Such products may be either non-critical or security may not be a concern for their asset owners.

(5) *Product identification.* As mentioned in Section 3, we were able to identify product names and version numbers for more than 64,072 devices belonging to about 500 different ICS products. This leaves roughly a third of devices unknown. Currently, the identification of products is performed using simple regular expressions [40]. We envision that machine learning techniques can be leveraged to increase the coverage of analyzed devices [6].

(6) *High interaction honeypots.* It is a known fact that Shodan includes a number of Internet-connected honeypots. Low interaction devices were detected/removed using the Shodan provided API for honeypot detection (including e.g. known Siemens PLC honeypots). However, sophisticated high interaction honeypots might be undetectable with this mechanism. Since honeypots in a number of cases are left intentionally vulnerable (in order to attract attacker's attention), their presence might introduce some level of bias to our results.

As Shodan provides a public interface to sensitive data about devices connected to the Internet, it comes with no surprise that the collected data should be analyzed with scrutiny. Having in mind the caveats presented here, we believe that the advantages of working with a public dataset, the most notable being reproducible results, justify the endeavor. Future work combining the empirical results presented in this paper together with insights collected from private data (e.g., inside enterprise networks) might provide more insights and lead to increased levels of confidence about current patching practices of mission-critical ICS devices.

7 RELATED WORK

In this section, we discuss relevant prior work on vulnerabilities lifecycle and software patching, followed by recent efforts to characterize Internet-connected ICS devices, and related literature on diffusion models.

7.1 Vulnerability Lifecycle

The lifecycle of a software vulnerability encompasses its discovery, disclosure, exploit availability, and patch availability. Frei et al. [15] examined more than 80 thousand security advisories for more than 14 thousand vulnerabilities reported in OSVDB [38] and NVD [37] between 1996 and 2006. The authors identified three risk phases: (1) *black risk*, between discovery and disclosure; (2) *gray risk*, between disclosure and patch availability; and (3) *white risk*, between patch availability and patch installation by the user. The authors found that 20% of the vulnerabilities were discovered 20 or more days before their disclosure (black risk). The authors also found that 15-45% of vulnerabilities did not have patches available at disclosure (gray risk). They also modeled risk distributions (likelihood of exploit/patch availability before/after disclosure) using Pareto and Weibull distributions.

Jones [19] proposed the daily vulnerability exposure (DVE) metric and vendor time to fix (TTF) to quantify and compare unpatched publicly-disclosed software vulnerabilities. The author analyzed the patches released by Microsoft for Windows 2000 Server and reported that Microsoft released patches for 96% of the disclosed vulnerabilities within 163 days.

Shahzad et al. [44] analyzed the software vulnerability lifecycle for popular vendors and products. They examined more than 46 thousand vulnerabilities from NVD, OSVDB, and from the collection by Frei et al. [15], and studied the impact of CVSS scores on vulnerability exploitation and patching. The authors also used association rule mining to identify typical vulnerability exploitation and patching behavior. They found that even though vendors have become more agile in releasing patches, hackers are increasingly exploiting zero-day vulnerabilities and closed-source software vendors tend to release patches faster than their open-source counterparts.

In contrast to prior work on vulnerability lifecycle, our findings in this paper shed new insight into the vulnerability lifecycle of ICS devices. Most importantly, our findings shed light into the patching behavior of ICS devices after vulnerability disclosures. While prior work has focused on the statistical analysis of historical data, in this paper we also present predictive models of population dynamics.

7.2 Patch Management and Software Patching Behavior

In [9, 14], the authors analyzed user-agent strings from Google's HTTP request logs to examine web browser update dynamics for popular web browsers. They found that auto-update mechanisms employed by Firefox and Chrome are effective in upgrading to the latest browser version. For example, the authors reported that 97% and 85% Chrome and Firefox users respectively upgraded within three weeks of a new release. In contrast, due to the lack of seamless auto-upgrade capability, only 53% and 24% Safari and Opera users respectively upgraded within three weeks of a new release. While the authors advocated silent software update process for most Web application, they cautioned its use for high availability applications where silent updates may cause service disruption. In this paper, our focus is on mission-critical ICS devices which are not suited for silent updates.

Zhang et al. [48] proposed a game-theoretic model for risk-gain analysis of exploitation and patching behavior in cloud computing environments. Their game-theoretic model incorporated costs for both the attacker (e.g., victim and vulnerability identification and dealing with defense mechanisms) and defender (e.g., reputation loss and utility misuse). They found that earlier actions by attackers and defenders can be more rewarding; thus, they advocate a more responsive and proactive patching strategy in cloud environments. We envision that the game-theoretic insights provided by [48], when coupled to the measurements reported in the present study, may give rise to novel principled patch management strategies.

There are a number of works that study the patch management problem from an optimization perspective [8, 18, 32, 46]. In [46], the authors consider the problem of controlling patch management so as to maximize system availability, under risk level constraints. Dey et al. [8] analytically compare different patch management strategies with respect to system availability and risk. In such previous studies, vulnerabilities are assumed to arrive according to a Poisson process. In addition, such works do not consider how the application of a patch at a given site should impact the patch management at other sites. Our work is complementary, as it provides a first step to enable patching decisions accounting for the state of the entire population [48].

7.3 Industrial Control Systems

Project SHINE (SHodan INtelligence Extraction) [40] was the seminal effort to analyze the Internet exposure of industrial control systems by querying Shodan [28]. The objective of this work was to identify patching practices for industrial control systems, especially the ones directly or indirectly connected to the Internet. We believe that identifying the practices adopted for patch installation is a necessary first step towards mapping the root cause for the typical large white risk duration that is accepted by most organizations. The authors discovered approximately 587 thousand such devices among top 11 manufacturers such as EnergyICT, Siemens, and Moxa. While the report itemizes device counts for popular manufacturers, it does not report whether these devices are susceptible to publicly disclosed vulnerabilities.

The research literature on large scale exploratory analysis and measurements of industrial control systems is rapidly growing [12, 17, 21, 22, 31, 34]. In [12, 34], the authors characterize different properties of Internet-connected ICS devices such as their level of exposure. Mirian et al. [34] scanned the IPv4 space between December 2015 and March 2016 to identify more than 60 thousand SCADA devices. The authors also created high interaction honeypots to identify who is searching for vulnerable systems. In [12], Feng et al. scanned the IPv4 space between August 2015 and March 2016 and identified more than 141 thousand ICS devices. The geolocation distribution of ICS devices reported in their work is similar to our findings. For example, similar to our findings, the top two countries with Internet-exposed ICS devices are USA and China according to Feng et al. [12]. The differences in the number of identified ICS devices can be attributed to the differences in the scanning techniques (ports scanned and protocol detection methodology) and placement of scanners. Our work extends the findings in prior work on Internet-exposed ICS devices by (1) correlating ICS scan data with vulnerability information extracted from NVD and (2) providing models to capture the population dynamics of patching behavior.

7.4 Diffusion Models

Diffusion models have been studied for decades to capture processes of adoption and substitution of goods [13]. Such models were extensively used to study population dynamics with a number of different applications, for marketing purposes [25], product development decisions [23] and technology diffusion assessment [16]. The way old products are replaced by new ones has been considered in many industries, including electronics, pharmaceutics and consumer goods [36].

Our work relies on a generalization to the original Bass diffusion model [4] to account for multiple generations of products [26]. In this paper, we extended the interpretation and use of such a generalized diffusion model to capture the adoption of different versions of a given software. By establishing this bridge between the dynamics of software updates and diffusion models, we envision that future work can leverage information about population dynamics for different purposes, including better recommendation of patching decisions.

8 CONCLUSION

In this paper, we presented the first large scale study of ICS patching practices. We gathered data on 100,766 devices, and as our first contribution we correlated this data with NVD to assess the impact of vulnerabilities

on ICS patching. Our study demonstrates that ICS devices are currently patched at an alarmingly slow rate. For example, approximately 50% ICS devices do not exhibit version increases even 60 days after vulnerability disclosure. The findings allow consumers to compare and contrast patching behavior for asset owners with different control system vendors and products. As our second contribution, we proposed a variation of the Bass model to track the population dynamics of patching behavior. Our evaluation indicated that the Bass model has comparable prediction accuracy when contrasted against traditional ARIMA time series forecasting models, while requiring less parameters whose meaning is amenable to direct physical interpretation.

The results and insights presented in this paper open up a number of interesting avenues for future investigation. In essence, we believe that problems that up to this point have been analyzed using local and/or private data, such as patch deferral, triggering of security alarms to ICS customers and setting machines to quarantine, may leverage the measurements and insights presented in this paper. The model and the methodology introduced in this work may be taken as building blocks to derive and prioritize protection and mitigation measures autonomously, based on the estimation of the easiest exploitation vector from the attacker's perspective. To facilitate reproducibility and future work, our code and data is available to all interested researchers upon request.

ACKNOWLEDGEMENT

We would like to thank John Matherly for providing us free access to the Shodan database. We would also like to thank our shepherd, Gil Zussman, and the anonymous reviewers for their valuable feedback.

REFERENCES

[1] R. Anderson and T. Moore. The economics of information security. *Science*, 314(5799):610–613, 2006.
[2] ANSI/ISA-99.02.01-2009 standard. *Security for Industrial Automation and Control Systems Part 2: Establishing an Industrial Automation and Control Systems Security Program.* 2009. http://www.isa.org/MSTemplate.cfm?MicrositeID=988&CommitteeID=6821.
[3] A. Arora, R. Krishnan, R. Telang, and Y. Yang. An empirical analysis of software vendors' patching behavior: Impact of vulnerability disclosure. *ICIS 2006 Proceedings*, page 22, 2006.
[4] F. M. Bass. A new product growth for model consumer durables. *Management science*, 15(5):215–227, 1969.
[5] G. Box, G. Jenkins, and G. Reinsel. *Time series analysis: Forecasting and control.* Prentice Hall, 1994.
[6] H. Chen, S. Hariri, R. Breiger, and T. Holt. Identifying SCADA Devices and their Vulnerabilities on the IoT, 2017. SaTC PI Meeting: http://cps-vo.org/node/30557.
[7] T. F. Coleman and Y. Li. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on optimization*, 6(2):418–445, 1996.
[8] D. Dey, A. Lahiri, and G. Zhang. Optimal policies for security patch management. *INFORMS Journal on Computing*, 27(3):462–477, 2015.
[9] T. Duebendorfer and S. Frei. Web browser security update effectiveness. In *International Conference on Critical Information Infrastructures Security*, 2009.
[10] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer, et al. The matter of heartbleed. In *ACM IMC*, pages 475–488. ACM, 2014.
[11] C. J. Easingwood, V. Mahajan, and E. Muller. A nonuniform influence innovation diffusion model of new product acceptance. *Marketing Science*, 2(3):273–295, 1983.
[12] X. Feng, Q. Li, H. Wang, and L. Sun. Characterizing industrial control system devices on the internet. In *ICNP*, pages 1–10. IEEE, 2016.
[13] J. C. Fisher and R. H. Pry. A simple substitution model of technological change. *Technological forecasting and social change*, 3:75–88, 1971.
[14] S. Frei, T. Duebendorfer, and B. Plattner. Firefox (In)Security Update Dynamics Exposed. *ACM SIGCOMM CCR*, 39:16–22, 2009.
[15] S. Frei, M. May, U. Fiedler, and B. Plattner. Large-Scale Vulnerability Analysis. In *ACM SIGCOMM LSAD Workshop*, 2006.
[16] P. A. Geroski. Models of technology diffusion. *Research policy*, 29(4):603–625, 2000.
[17] H. R. Ghaeini and N. O. Tippenhauer. Hamids: Hierarchical monitoring intrusion detection system for industrial control systems. In *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, pages 103–111. ACM, 2016.
[18] C. Ioannidis, D. Pym, and J. Williams. Information security trade-offs and optimal patching policies. *European Journal of Operational Research*, 216(2):434–444, 2012.
[19] J. R. Jones. Estimating Software Vulnerabilities. *IEEE Security and Privacy*, 2007.
[20] W. Kandek. The laws of vulnerabilities 2.0. *BlackHat, Las Vegas, NV, USA*, 2009.

[21] F. Khorrami, P. Krishnamurthy, and R. Karri. Cybersecurity for control systems: A process-aware perspective. *IEEE Design & Test*, 33(5):75–83, 2016.
[22] C. Konstantinou, M. Sazos, and M. Maniatakos. Attacking the smart grid using public information. In *Test Symposium (LATS), 2016 17th Latin-American*, pages 105–110. IEEE, 2016.
[23] V. Krishnan and K. T. Ulrich. Product development decisions: A review of the literature. *Management science*, 47(1):1–21, 2001.
[24] M. Lelarge and J. Bolot. Network externalities and the deployment of security features and protocols in the internet. *ACM SIGMETRICS Performance Evaluation Review*, 36(1):37–48, 2008.
[25] V. Mahajan, E. Muller, and F. M. Bass. New product diffusion models in marketing: A review and directions for research. In *Diffusion of technologies and social behavior*, pages 125–177. Springer, 1991.
[26] V. Mahajan, E. Muller, and F. M. Bass. Diffusion of new products: Empirical generalizations and managerial uses. *Marketing Science*, 14(3), 1995.
[27] P. Maillé, P. Reichl, and B. Tuffin. Interplay between security providers, consumers, and attackers: a weighted congestion game approach. In *International Conference on Decision and Game Theory for Security*, pages 67–86. Springer, 2011.
[28] J. Matherly. Shodan. https://www.shodan.io.
[29] J. Matherly. Simple Security Metric: Internet Connected ICS, 2017. KIACS2017 keynote speak video: https://livestream.com/hdmediakw/events/7107294/videos/151813225.
[30] P. McDaniel and S. McLaughlin. Security and privacy challenges in the smart grid. *IEEE Security and Privacy*, 7(3):75–77, 2009.
[31] S. McLaughlin, C. Konstantinou, X. Wang, L. Davi, A.-R. Sadeghi, M. Maniatakos, and R. Karri. The cybersecurity landscape in industrial control systems. *Proceedings of the IEEE*, 104(5):1039–1057, 2016.
[32] P. Mell, T. Bergeron, and D. Henning. Creating a patch and vulnerability management program. *NIST Special Publication*, 800:40, 2005.
[33] P. Mell, K. Scarfone, and S. Romanosky. A complete guide to the CVSS 2.0, 2016. https://www.first.org/cvss/v2/guide.
[34] A. Mirian, Z. Ma, D. Adrian, M. Tischer, T. Chuenchujit, T. Yardley, R. Berthier, J. Mason, Z. Durumeric, J. A. Halderman, et al. An internet-wide view of ICS devices. In *IEEE PST*, 2016.
[35] J. A. Norton and F. M. Bass. A diffusion theory model of adoption and substitution for successive generations of high-technology products. *Management science*, 33(9):1069–1086, 1987.
[36] J. A. Norton and F. M. Bass. Evolution of technological generations: the law of capture. *Sloan Management Review*, 33(2):66, 1992.
[37] National Vulnerability Database (NVD). https://nvd.nist.gov.
[38] Open Source Vulnerability Database. http://osvdb.org.
[39] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow. IoTPOT: Analysing the Rise of IoT Compromises . In *USENIX WOOT*, 2015.
[40] B. Radvanovsky and J. Brodsky. Project SHINE (SHodan INtelligence Extraction). In *Tech. rep.*, 2014.
[41] R. Radvanovsky and J. Brodsky. *Handbook of SCADA/control systems security*. CRC Press, 2016.
[42] S. Ransbotham and S. Mitra. Choice and chance: A conceptual model of paths to information security compromise. *Information Systems Research*, 20(1):121–139, 2009.
[43] C. Scott and R. Carbone. Designing and Implementing a Honeypot for a SCADA Network. SANS Institute Reading Room, 2014.
[44] M. Shahzad, M. Z. Shafiq, and A. X. Liu. A Large Scale Exploratory Analysis of Software Vulnerability Life Cycles. In *International Conference on Software Engineering (ICSE)*, 2012.
[45] M. Souppaya and K. Scarfone. Guide to enterprise patch management technologies. *NIST*, 800:40, 2013.
[46] T. Uemura and T. Dohi. Optimal security patch management policies maximizing system availability. *Journal of Communications*, 5(1):71–80, 2010.
[47] Verizon. Verizon 2016 data breach investigations report. www.verizonenterprise.com/verizon-insights-lab/dbir/2016/, 2016.
[48] S. Zhang, X. Zhang, and X. Ou. After we knew it: empirical study and modeling of cost-effectiveness of exploiting prevalent known vulnerabilities across IaaS cloud. In *ASIA CCS*, 2014.