Empirical CVSS Parameterization and Enrichment for Data-Driven Software Vulnerability Risk Assessment

Extracting Threat Intelligence from Public and Blackhat Forums for Temporal CVSS Estimation

MATHEUS MARTINS and LEANDRO P. AGUIAR, Siemens Corporate Research
MATEUS NOGUEIRA, BRUNO HRYNIEWICZ, MIGUEL BICUDO, DANIEL JIMENEZ, LEONARDO VENTURA, FABRICIO FIRMINO, LUCAS SENOS, GABRIEL RIBAS, and DANIEL MENASCHÉ, Federal University of Rio de Janeiro
TREYTON KRUPP, ASHTON WOIWOOD, and ZUBAIR SHAFIQ*, University of Iowa
LUCA ALLODI, Eindhoven University of Technology

Patch management is a critical aspect of securing IoT and industrial control systems (ICS) against attacks. Nonetheless, the patch management problem is more challenging in the realm of IoT and ICS as compared to their traditional desktop counterparts, as the update of such devices usually needs to be scheduled in advance and may have cyber-physical implications. In this paper, we present measurements and models for dynamic risk assessment, to assist in patch management decisions. Building on top of CVSS scores, we propose a methodology to combine historical data on vulnerability weaponization into CVSS scores, to provide both explanatory as well as predictive insight into how risk evolves over time. The proposed methodology combines a classifier with statistical inference, and is flexible to account for different aspects of the vulnerability lifecycle.

CCS Concepts: • **Computer systems organization → Embedded systems**; *Redundancy*; Robotics; • **Networks →** Network reliability;

Additional Key Words and Phrases: datasets, neural networks, gaze detection, text tagging

ACM Reference format:
Matheus Martins, Leandro P. Aguiar, Mateus Nogueira, Bruno Hryniewicz, Miguel Bicudo, Daniel Jimenez, Leonardo Ventura, Fabricio Firmino, Lucas Senos, Gabriel Ribas, Daniel Menasché, Treyton Krupp, Ashton Woiwood, Zubair Shafiq, and Luca Allodi. 2018. Empirical CVSS Parameterization and Enrichment for Data-Driven Software Vulnerability Risk Assessment. *J. ACM* 37, 4, Article 111 (August 2018), 17 pages.
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Background and motivation. Patch management is a critical aspect of securing IoT and industrial control systems against attacks [9, 15]. Due to the the intrinsic physical implications and critical nature of such systems, traditional IT approaches do not apply in most cases, resulting in a number of unpatched systems for which updates are available. Additional information about vulnerabilities and risks need to be considered to support patch decisions that minimize risks while maximizing availability of the production processes [8].

0004-5411/2018/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

Prior art. Risk assessment for patch management usually relies on the common vulnerability scoring system (CVSS). The CVSS is a well established *de facto* solution for risk assessment and the National Vulnerability Database (NVD) associates a CVSS score to each of the known vulnerabilities. Although the CVSS score contemplates adjustments in the risk score to account for temporal variations, there is no well established procedure to estimate how the CVSS score varies over time. Such an assessment can be useful for predictive purposes and for historical understanding of risk evolution. To the best of our knowledge, there is no public data on the evolution of CVSS scores.

Goals. In this paper, we propose measurements, models and methods for dynamic risk assessment, to assist in the process of patch management. Our models, parameterized through measurements, account for the vulnerability lifecycle, and capture how risk evolves over time. By assessing the risk as a function of historical data about vulnerabilities, we envision that current patching strategies can be revised and fine tuned towards a true closed loop between vulnerability rating and patch installation metrics. Knowing when risk will eventually surpass a given threshold can assist in decisions about patch installation or deferral.

Contributions. We summarize our key contributions as follows.

Risk assessment model. We propose a model which empirically enriches the CVSS score, for predictive purposes. The model leverages the fact that the CVSS score already accounts for a temporal subscore, and we use data on the time instants at which vulnerability weaponizations occurred to tune the temporal evolution of CVSS.

Longitudinal analysis of weaponization times. We report statistics on the lifecycle of vulnerabilities, based on matchings between CVEs and exploits. In particular, we study how weaponization times depend on different parameters, such as the base CVSS score associated to a vulnerability.

Vulnerability classifier. We assess different classifiers with the aim of estimating the time it takes for a vulnerability to be weaponized for the first time. The proposed classifiers divide the vulnerabilities into four classes, based on the time at which the first weaponization occurred.

Paper structure. The remainder of this paper is organized as follows. First, we introduce basic background and terminology, followed by the measurement framework used for empirical risk assessment. Then, Section 4 reports a longitudinal analysis of findings based on the data. In Section 5 we present our risk model, followed by vulnerability classification results in Section 6. Section 8 contains related work and broader discussions, and Section 10 concludes.

2 RISK ASSESSMENT AND CVSS PRIMER

The CVSS score is the *de facto* solution for risk assessment of software vulnerabilities. The CVSS score is available under two versions, version 2 and 3. In both versions, it encompasses a base, a temporal and an environmental score. To each vulnerability that is granted a CVE identifier, its corresponding base score is publicly made available at the National Vulnerability Database (NVD).

The fundamental idea behind the base CVSS score is the notion that risk is given by the product of impact and exposure. Whereas *impact* quantifies negative consequences due to an incident, *exposure* measures the probability that an incident will occur. The base CVSS score comprises subscores which account, for instance, for the impact caused by an incident and the access vectors associated to each vulnerability.

2.1 Temporal CVSS score

The CVSS temporal and environmental modifiers are used to adapt the base CVSS score according to the evolution of risk and specifics of the environment where the asset being considered is placed.

For the purposes of this work, the most important feature of the temporal CVSS score consists of the fact that it modifies the base CVSS score, in such a way that the temporal CVSS score varies between 66% and 100% of the base score. In the most favorable scenario, the value of the CVSS temporal score can be up to 66% of the value of the base CVSS score. In the less favorable scenario, the temporal CVSS score equals the base CVSS score.

In this work, we look for a compromise between leveraging the CVSS score as our reference risk metric, while at the same time enriching its usefulness with empirical data about weaponization events. Note that the temporal CVSS score itself also comprises subscores, related to the availability and maturity of patches and exploits (weapons). Nonetheless, there are no common guidelines to quantify maturity. For this reason, in this paper we do not consider the temporal CVSS subscores for tracking risk evolution. *Instead, we take a simpler empirical approach, and assume that the risk uniformly increases every time a weapon is made available, in such a way that the risk is minimal when there are no weapons available, and reaches its maximum level when all the known weapons become available.*

2.2 Terminology

In what follows, we briefly introduce the terminology adopted in the remainder of this paper.

CVE disclosure day: the day at which a vulnerability, identified by its Common Vulnerability and Exposures identifier (CVE id), is publicly disclosed.

CVE reservation day: the day at which a vulnerability is confirmed by a vendor, and the corresponding CVE id is reserved. The reservation day always occurs before or at the same day that as the disclosure day.

Weaponization: the event of publicly making available an exploit (weapon) for a vulnerability.

Base score: the basic CVSS risk score, provided by the National Vulnerability Database.

Temporal score: built on top of the base CVSS score, the temporal score accounts for the risk evolution over time.

Temporal score range: the temporal score varies, according to the CVSS standards, between 66% and 100% of the base CVSS score.

3 MEASUREMENT FRAMEWORK

Next, we describe the measurement framework used to collect the data that is fed into the proposed classification and prediction algorithms. We extract data from two main sources: ExploitDB and NVD. ExploitDB and NVD provide data on exploits and vulnerabilities, respectively. In particular, ExploitDB already contains, for many of the exploits in its catalog, the corresponding CVE identifiers which allow us to relate exploits to vulnerabilities. Figure 1 illustrates the considered measurement framework.

We feed an Elastic Search database with data on matchings between exploits and CVEs. For each exploit we know its publication date, and for each CVE we know its discovery and disclosure dates. In addition, we also have textual information about the CVEs, as well as their corresponding CVSS scores and subscores. Using these data, we construct a classifier which aims at inferring the dates at which the first weaponization of each vulnerability will occur. We also gather statistical information about each class of vulnerabilities. Combining the results of the classifier with statistical information about each class, we track the evolution of the CVSS score over time. More details about each of such steps will be provided in the forthcoming sections.

4 LONGITUDINAL DATA ANALYSIS

Next, we report a longitudinal data analysis on the matchings between CVEs and exploits. Each matching corresponds to an exploit that leverages a given vulnerability, identified by its CVE

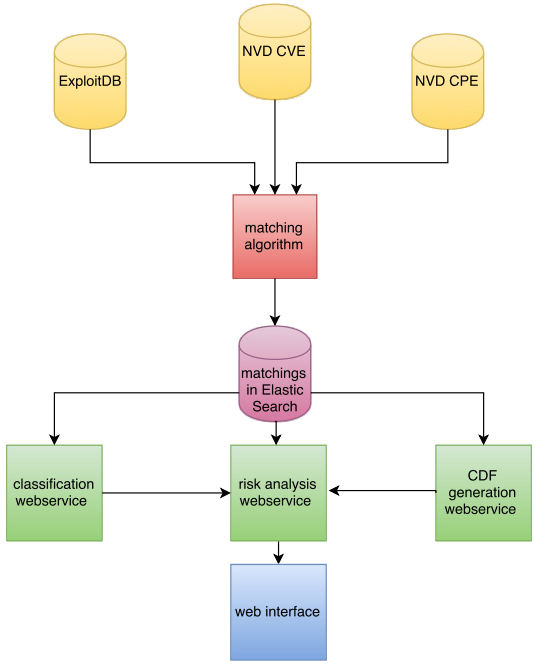


Fig. 1. Measurement framework

identifier. Note that a single CVE may match with multiple exploits, as there might be multiple weapons, with different levels of maturity, that leverage the same vulnerability. Similarly, a single exploit may match to multiple CVEs, if it leverage multiple vulnerabilities.

Matchings. The matchings between CVEs and exploits were obtained from two sources: 1) directly from the ExploitDB website and 2) indirectly, by inspecting the source code of the exploits in search for regular expressions that indicated their connection to a certain set of CVEs. We obtained a total of 25,894 matches which constitute our matching database.

Figure 2 shows the histogram of the number of vulnerabilities per product, for the top 20 products. The Linux kernel has the largest number of vulnerabilities (200 at total). Product with index 7 is a placeholder substituting all unidentified products. We also counted the number of exploits that matched each CVE. For the vast majority of CVEs (roughly 18,000) we found only one associated exploit. Nonetheless, for some CVEs we found up to 10 associated exploits (see Figure 3).

4.1 Time to weaponization

The reference day used in the plots that follow is the CVE reservation day. Figure 4 shows the CDF of the difference between the first weaponization day and the CVE reservation day, for all the considered matches. A significant fraction, 81%, of the exploits are zero-days (i.e., the exploit is released at the same day or before the CVE disclosure).

4.2 Impact of IoT-related products

Figure 5 compares the time to weaponization for IoT and non-IoT products. To simplify the distinction, we assume that IoT products are those that have actuators, sensors or motors. A product is classified as being IoT if it contains suggestive tags in its name, such as “camera”, or if it is from

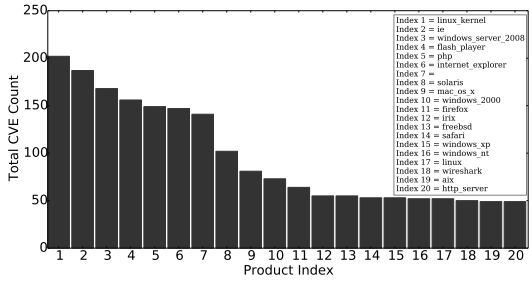


Fig. 2. Histogram of vulnerabilities per product

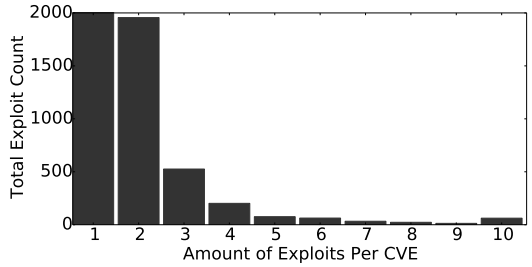


Fig. 3. Histogram of exploits per vulnerability. Most of the vulnerabilities (17,900 CVEs) have one associated exploit (for presentation purposes, the y-axis is restricted to vary up to 2,000).

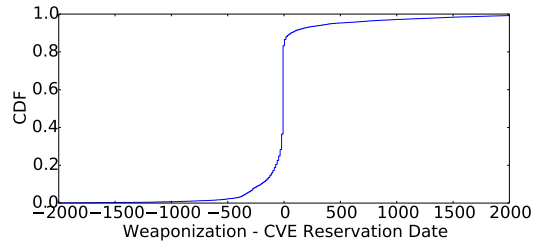


Fig. 4. CDF of first weaponization day since CVE reservation, for the whole population

one of the following vendors: Siemens, Schneider-Electric, Allegro, Rockwell Automation, Brother, Honeywell, Webcamxp or Axis. As shown in Figure 5, the vendors of IoT products tend to be more reactive to exploits, announcing CVEs more promptly after an exploit is made available. In addition, after CVEs are disclosed, exploits are made available slightly faster for IoT products as compared to their non-IoT counterparts. The figure also indicates that roughly 40% of the exploits are zero-day and 40% (resp., 20%) are made publicly available before (resp., after) the CVE reservation.

Note that because a significant fraction of the exploits are devised either before or at the day of CVE reservation, a CVSS-centered risk model must necessarily be able to track historical risk evolution. This is because CVSS scores are released together with CVEs. By the time the CVSS score is released, with high probability there are already exploits circulating in the network.

Some of the discussion that follows provides insights on the root causes for the distinct vulnerability lifecycles observed for IoT and non-IoT products, and a more in-depth analysis is left as subject for future work.

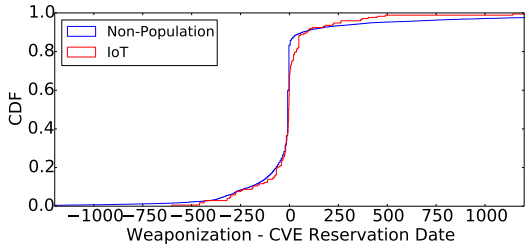


Fig. 5. Days to first weaponization (IoT and non-IoT products)

4.3 Impact of criticality

Figure 6 shows the CDF of the time to weaponization, conditioned on the criticality of the CVE as measured by its CVSS score. We have also investigated the constituents of the CVSS score, such as the confidentiality, integrity and availability impact subscores (not shown in this paper). Interestingly, the CDFs associated to such subscores are very similar to those presented in Figure 6 for the CVSS score as a whole.

We note that vendors react more promptly after finding exploits for critical CVEs as opposed to non-critical ones, i.e., critical exploits have CVEs disclosed quicker than non-critical ones. Noting that vulnerabilities in IoT products tend to have higher criticality, this behavior may partially explain the distinction between the lifecycle of IoT and non-IoT products. After CVE disclosure, in contrast, the time it takes for hackers and researchers to publicly announce exploits for critical vulnerabilities is usually larger than that for non-critical ones. We conjecture that this may be due to the value of such exploits in the black market, and we are currently undergoing a measurement campaign in such markets to check that hypothesis. Note also that roughly 40% of critical exploits are released after CVE disclosure. Whether the disclosure of the CVE has prompted more weaponizations, or whether such exploits were already available in black markets, is also subject of ongoing research.

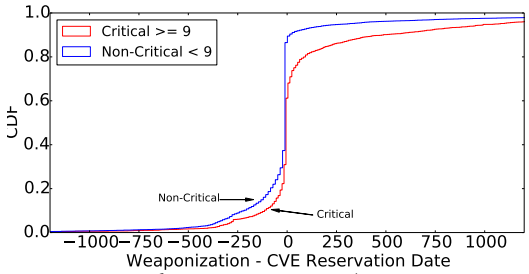


Fig. 6. Days to first weaponization (given CVSS scores)

4.4 Impact of CVSS subscores

Figures 7, 8 and 9 show the CDF of days to first weaponization, conditioned on CVSS availability impact, confidentiality impact and integrity impact subscores, respectively. As mentioned above, it is interesting to see that there is a common pattern among all those CDFs: in all cases, for the most critical vulnerabilities (that have complete impact) the corresponding CDF lower bounds the less critical ones (that have partial or no impact). This means that the comments made in the previous section are applicable not only at the general CVSS level but also at the level of subscores. In particular, vendors tend to react more promptly to the creation of exploits to impactful vulnerabilities, and hackers tend to be more cautions or challenged to release those exploits once the vulnerabilities are made publicly available.

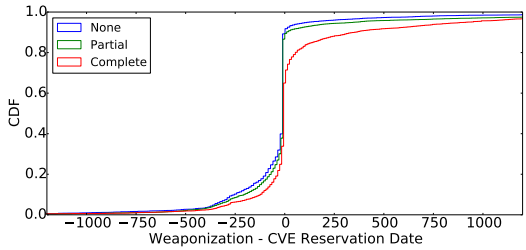


Fig. 7. Days to first weaponization (given CVSS availability impact)

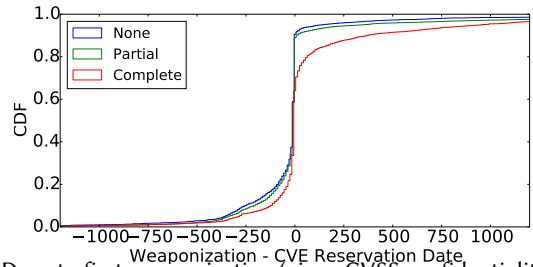


Fig. 8. Days to first weaponization (given CVSS confidentiality impact)

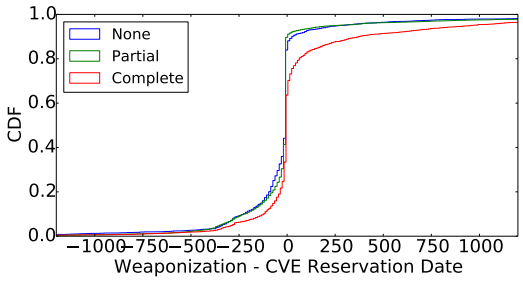


Fig. 9. Days to first weaponization (given CVSS integrity impact)

4.5 Impact of product and vendor

Impact of product. Next, we consider the top 5 products with largest number of associated exploits. Figure 10 shows the CDF of the time to weaponization for such products. Adobe flash player is the quickest to disclose a CVE after an exploit is made available, with Internet Explorer being the slowest. In what follows, we analyze reaction times conditioned on vendors and observe that in general Adobe tends to be quick to react after its products are weaponized.

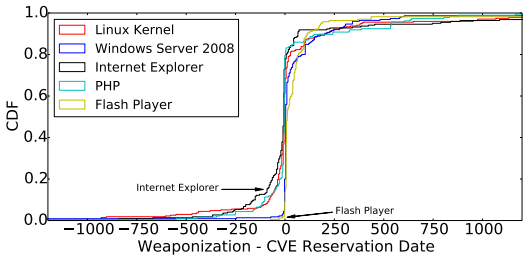


Fig. 10. Days to first weaponization, for top 5 products

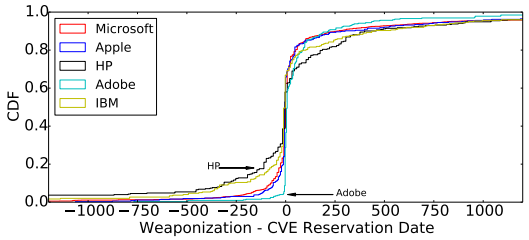


Fig. 11. Days to first weaponization, for top 5 vendors

Impact of vendor. Next, we consider the CDFs of weaponization times for different vendors, as shown in Figure 11. We find that Adobe and HP are at two extremes in terms of responsiveness. Whereas Adobe is quick to disclose CVEs after exploits are announced, HP takes the longest to respond. In contrast, for the vulnerabilities that are disclosed before exploit availability, it takes longer for hackers and researchers to publicly disclose the exploits for HP as compared to Adobe. This might be the case because HP exploits are more valuable in the black market, or because they are harder to construct. Ongoing work involves identifying the driving forces behind vendors response times. To this aim, we are crawling black markets to gain further insights on the evolution of the price of exploits and their complexities.

In general, we observe that the reaction time corresponding to the top 5 vendors and products (Figures 10 and 11) is similar to the one exhibited for the population of IoT and non-IoT products (Figure 5). Notable exceptions are Adobe and flash player, which have much faster reaction times, with less than 10% of their exploits being available strictly before CVE disclosure. Regarding exploits made available after CVE disclosures, we observe that the fraction of those for the top 5 vendors and products tends to be higher than the ones observed in the population as whole (roughly 40% of the points in Figures 10 and 11 have an abscissa greater than 0, as opposed to approximate 20% in Figure 5).

4.6 Take away message

The key take away message from our longitudinal analysis refers to the fact that different vendors and products are associated to distinct lifecycles which can be estimated using historical data. In addition, for a significant fraction of vulnerabilities we have exploits available before or after CVE disclosure, which indicates that exploits foster the reservation of CVE identifiers.[1]

5 RISK MODEL

Next, we describe the proposed risk assessment model. First, we introduce our basic working assumptions.

5.1 Working assumptions

The proposed model is based on the following working assumptions:

(A1) Risk is measured using CVSS scores. As CVSS scores are widely used as a metric for risk assessment, the proposed risk assessment model extends CVSS in a natural way. By leveraging the temporal aspect of CVSS, we propose the use of weaponization information to track risk evolution.

(A2) Risk varies between 0 and CVSS base score. To handle common scenarios wherein exploits are available before CVE disclosure, we allow risk scores to vary between 0 and CVSS base

[1]In this paper, we do not distinguish between exploits and proof-of-concept weapons. We envision that the weapons that appear before CVE disclosure are PoC, and afterwards full-fledged exploits.

score. Note that the standard CVSS score does not handle such cases, as CVSS scores are released at CVE disclosure and the tracking of risk before CVE disclosure is out of the scope of CVSS.

(A3) For CVEs for which exploits are not available at the time of CVE disclosure, the risk level at CVE disclosure equals 66% of the base score. According to the CVSS standards, the temporal CVSS score varies between 66% and 100% of the base score. As soon as a vulnerability is disclosed, in the absence of exploit information we assume that the risk score is at its most favorable level. Note, however, that if zero-day exploits are likely, the model will capture that risk will immediately rise, possibly at day zero.

(A4) At every weaponization, risk increases uniformly. We assume that the increments in risk level at every weaponization are equal. Such assumption can be relaxed in future work, leveraging additional information about exploit data.

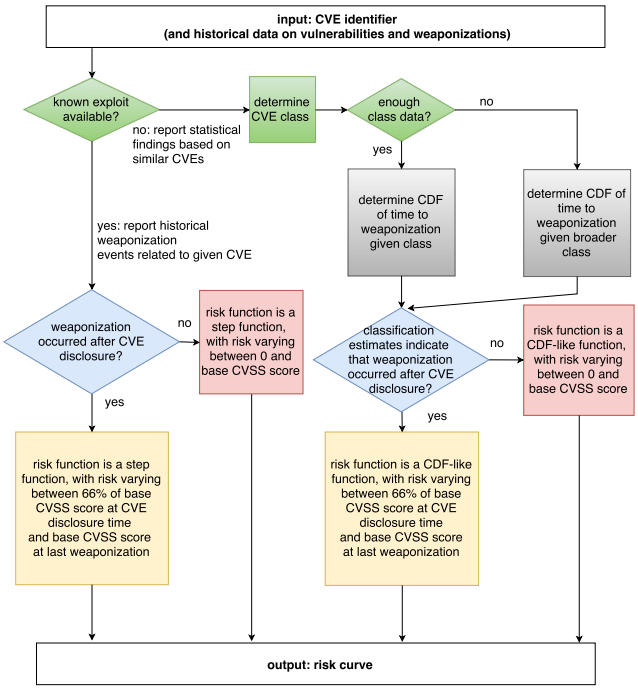


Fig. 12. Risk model

5.2 Model overview

Given a vulnerability, characterized by its CVE identifier, we consider two possible cases. First, if exploit information is available for that CVE, the model is used for explanatory purposes, to indicate how risk has evolved over time. The knowledge of historical risk evolution can help, for instance, to determine for how long risk has surpassed a given threshold. The longer the risk sustained high levels, the more likely it is that an incident will occur. Otherwise, if exploit information is not available for the considered CVE, we leverage data about the whole population to infer or predict how risk will evolve over time. An overview of the proposed model is summarized in Figure 12.

5.3 Risk assessment in the absence of exploit information

Next, we consider a CVE for which there is no public information available on weaponizations. In this case, we divide the problem of risk assessment into two problems, a) classification and b) statistical inference.

5.3.1 Vulnerability classification. Next, our goal is to classify vulnerabilities to assist in the process of determining when an exploit will be made available. To this aim, we use CVE features such as textual description, CVSS scores and subscores, vendor and product.

Let Δ be the time difference between CVE disclosure and weaponization, measured in days. In the simplest setting, we set four classes: (a) $\Delta \leq -60$, (b) $-59 \leq \Delta < 0$, (c) $0 \leq \Delta \leq 60$ and (d) $\Delta > 60$. The division into four classes corresponded to a good compromise between a reasonable classification performance and the generation of classes with enough samples so as to be able to obtain a CDF of time to weaponization per class.

5.3.2 CDF of time to weaponization. Given the CVE class, we construct the corresponding CDF of time to weaponization. Such CDF can be conditioned on different features of the CVE, including its product or vendor. The level of filtering must be set so as to balance between specialization of the results and sample size.

If CVE disclosure is estimated to have occurred before first weaponization (bottom right diamond in Figure 12), risk level starts at 66% of base CVE level (Assumption A3). Otherwise, it starts at zero (Assumption A2). Then, the CDF of time to weaponization is used to interpolate between the minimum risk level and the CVSS base score. This way, at every weaponization that occurred at the considered class, risk increases uniformly (in agreement with Assumption A4).

5.4 Illustrative examples

5.4.1 Abstract examples. Figure 13 illustrates, through four examples, how risk evolves over time under the four settings considered in Figure 12. When exploit data is available, the model serves as a visualization tool to quickly track how risk evolved over time (left hand side of Figure 13). When exploit data is not available, the model combines class data about time to exploitation with CVSS scores for the given vulnerability to produce estimates of risk evolution (right hand side of Figure 13).

5.4.2 Concrete examples. Figure 14 illustrates through two concrete examples (Heart Bleed and Wanna Cry) the output of our model. In the x-axis we have days (where day 0 corresponds to vulnerability disclosure) and in the y-axis we have CVSS scores. Blue, orange and green lines correspond to ground-truth, risk as predicted by our classifier-based model and risk as predicted by the vendor-based model, respectively. The small numbers that appear close to the steps in the blue lines are the identifiers of the corresponding exploits at ExploitDB. Each exploit disclosure causes the risk to increase (see Section 5.1). Note that whereas the classifier-based model is conservative for the Heart Bleed vulnerability, it is lenient for Wanna Cry.

6 CVE CLASSIFICATION DETAILS

Next, we report our preliminary results on vulnerability classification. As discussed in Section 5.3.1, our goal is to classify the vulnerabilities based on the estimated time to their first weaponization. In this section, our aims are to 1) introduce the adopted classification methodology and 2) compare and report the performance of five different classifiers.

The first step in our methodology consists in filtering those CVEs for which we find at least one corresponding exploit. Such pairs of CVEs and exploits constitute the matchings which are used to train the classifiers. Our database contains 25,894 such matchings (see Section 4).

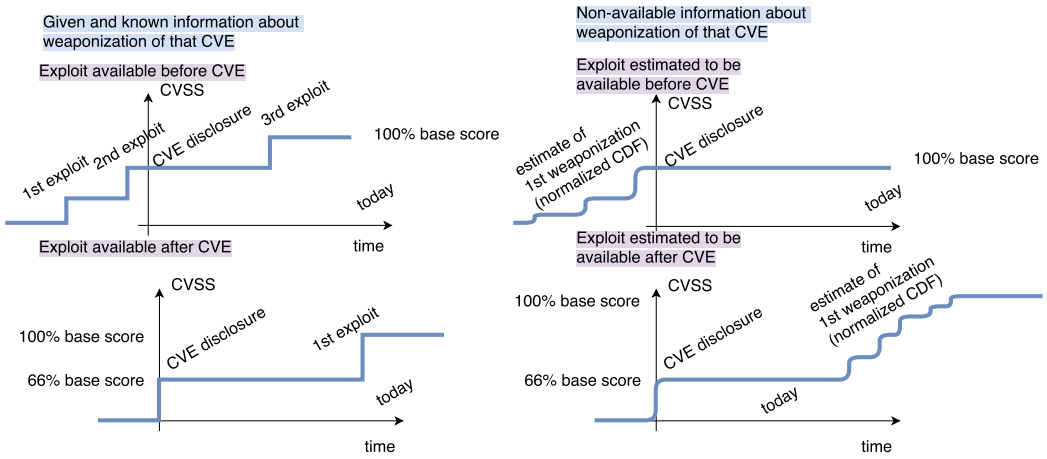


Fig. 13. Illustrative examples of CVSS risk score evolution

6.1 Feature vector

To each CVE we associate a feature vector comprising 18,776 features. The considered features include the CVE vendor name, product name, CVSS subscores (availability impact, integrity impact, confidentiality impact, access vector, access complexity and authentication requirements), CVSS score and features obtained from the CVE description. The importance of the latter has been highlighted in [2]. In addition, we also included features accounting for side information on the vendor and the products impacted by the vulnerability. In our tests, we have observed a significant improvement on the accuracy of the classifier after including such two last sets of features.

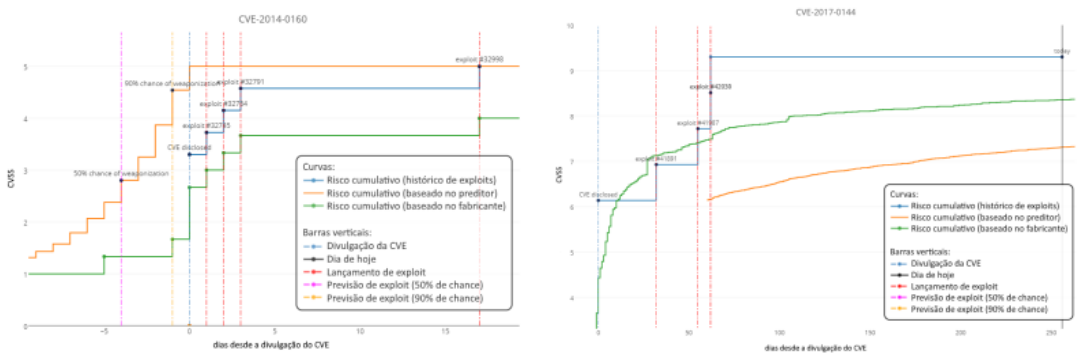
Note that the CVSS scores and subscores are numerical features, as determined by the NIST and NVD standards. To obtain features out of the product and vendor names, we applied the *hash trick*, which consists of associating to each name a distinct dimension in the feature vector, and allowing the values of those features to vary between 0 and 1.

6.2 Classifiers and accuracies

Our first approach to build a predictive model of weaponization time consisted in framing the problem as a regression problem. However, the regression results were not satisfactory, in part due to the irregular distribution of the samples (see Section 4). To cope with such challenge, the problem was re-framed as a classification problem. We selected four windows of time as the target classes, so as to roughly have a uniform number of samples at each window. The four classes are described in Section 5.3.1.

Five different classifiers have been considered: (1) Naive Bayes, (2) linear SVM, (3) SVM with a radial basis function kernel (rbf), (4) Gradient Boosting Tree (GBT) and (5) Neural Network (NN) with three hidden layers. Hyperparameters have been optimized using a grid search approach. The dataset was split into training, validation and test sets with proportions of 60%, 20% and 20%, respectively. We repeat each experiment 30 times, and report the accuracy and standard deviation associated to each classifier in Table 1. The neural network reached an accuracy of 67%, and corresponding standard deviation of 0.03.

Figure 15 shows the CDF of the CVSS error, accounting for our model predictions against multiple baselines. In Figure 15(a), 15(b), 15(c) and 15(d) the baselines are (*a*) the CVSS score at the day at which the exploit was released, under the assumptions of Section 5.1, (*b*) the CVSS scores at



(a) Heart Bleed (b) Wanna Cry

Fig. 14. Heart Bleed and Wanna Cry. Examples of risk evolution over time: blue, orange and green correspond to ground-truth, risk as predicted by our classifier-based model and risk as predicted by the vendor-based model. Note that whereas the classifier-based model is conservative for the Heart Bleed vulnerability, it is lenient for Wanna Cry.

the days at which there was at least one exploit available, (*c*) the mean CVSS score of the class of the vulnerability and (*d*) the median CVSS score of the class of the vulnerability. As we move from Figure 15(a) to 15(d), we consider increasingly less information and/or longer periods of time, which result in increased levels of error. Note that in Figures 15(c) and 15(d) we consider only the CVSS mean and median of the class, and still find that for 20% of the vulnerabilities the CVSS error is below 2 units. As shown in Figure 15(a), for 85% of the vulnerabilities our classifier was able to correctly assess the CVSS at the day at which the first exploit was released. We are currently investigating alternative ways to assess the classification error, e.g., directly accounting for the error in the prediction of the day at which the first exploit is released.

Table 1. Accuracy test for different classifiers

Classifier	Average Accuracy	STD Accuracy
Naive Bayes	0.24	0.01
SVM (linear)	0.56	0.02
SVM (rbf)	0.57	0.01
GBT	0.62	0.13
Neural network	**0.67**	**0.03**

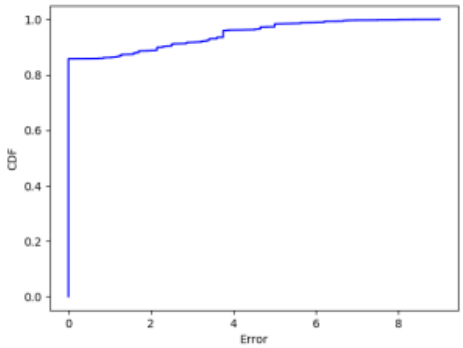
6.3 Take away message

Given a CVE for which weaponization information is not available, a neural network can be used to produce a vector of class *responsibilities*. Such responsibilities indicate the likelihood that the CVE pertains to each class. Each class, in turn, is associated to a different set of statistical measures, whose applicability to the considered CVE should be ranked based on the responsibility estimates.

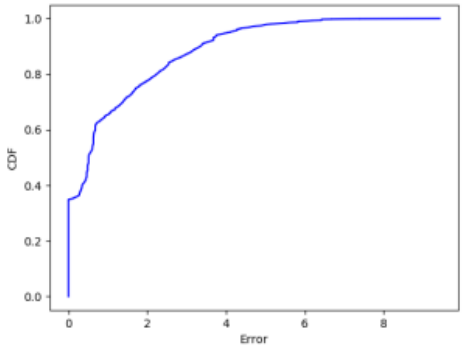
7 CVSS ENRICHMENT THROUGH BLACKHAT FORUM MINING

7.1 Why resorting to blackhat forums?

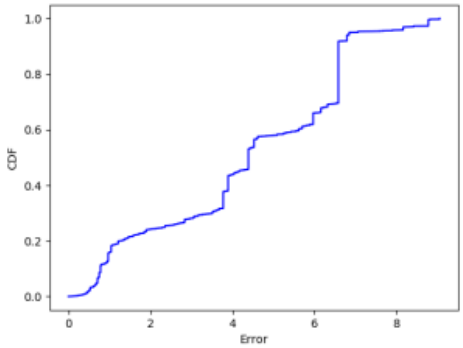
In figure17 we show how many exploits appeared in ExploitDB associated with the vulnerabilities found in crimeBB. When the number of matches between the vulnerabilities in crimeBB and the



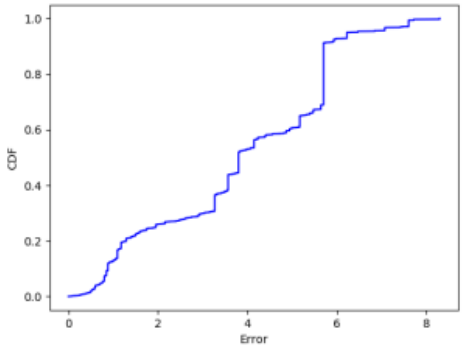
(a) CVSS error at 1st exploit announcement



(b) mean CVSS error at days in which at least 1 exploit was available



(c) CVSS error taking mean CVSS of class as baseline



(d) CVSS error taking median CVSS of class as baseline

Fig. 15. Assessing classification accuracy: CDF of CVSS error under different baselines. As shown in part (*a*), for 85% of the vulnerabilities our classifier was able to correctly assess the CVSS at the day at which the first exploit was released.

exploits from ExploitDB is zero, we see that there are many vulnerabilities with a high degree of exploitability (i.e. Functional or High) that ExploitDB doesn't have records about them. Thus, this indicates that many vulnerabilities are being exploited in the wild and the most reliable and accessible sources of information about exploits like ExploitDB, many times, won't even have records of these exploits.

7.2 Benefiting from blackhat forums and IBM X-Force: comparative analysis of CVSS parametererization from multiple sources

include here tree from Senos + Gabriel
indicate that it may be helpful to improve matching

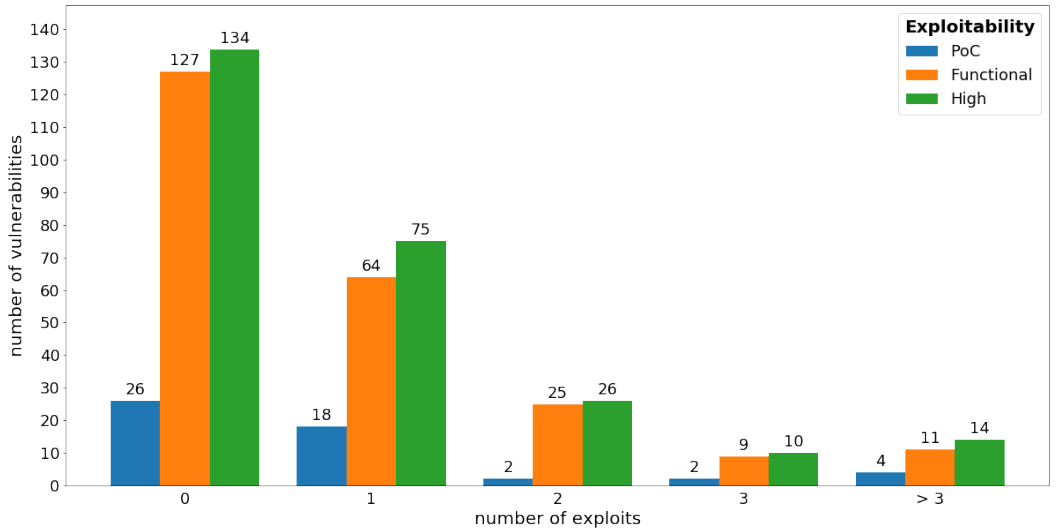


Fig. 16. crimeBB vulnerabilities and how many exploits from ExploitDB were associated with them

7.3 Exploitation in the wild: a feasiblity analsyis of augmenting CVSS from blackhat forums

Next, our goal is to show the feasibility of using data from blackhat forums to augment CVSS accounting for exploitation of vulnerabilities in the wild. Figures 18 and 19 show the decision trees to classify the exploitability of vulnerabilities based on CrimeBB data.

8 RELATED WORK

Patching practices and patch management. There is a vast literature on patching practices and patch management [3], ranging from optimal patching strategies [6] to ICS patch behavior characterization [15]. In this work, we propose a risk model which we envision to be coupled with existing patch management solutions. The ultimate goal is to devise risk-aware patch management strategies, which make use of a risk model for predictive purposes as well as to better document current practices and incrementally refine them [1].

Risk assessment and vulnerability lifecycle. The lifecycle of vulnerabilities has been studied for more than a decade [5, 7, 10, 11]. Predicting whether a given vulnerability was already exploited, based on features such as its textual description, has been considered in [4]. Recent works have also considered the interplay between the lifecycle of vulnerabilities and patching practices [14]. In this paper, in contrast, we couple the lifecycle of vulnerabilities with CVSS scores to produce a predictive model for risk.

9 DISCUSSION

CVE-centric versus exploit-centric risk assessment. In this paper we have considered a CVE-centric perspective on risk assessment. In particular, we consider the retrospect tracking of risk when weaponization occurs before CVE disclosure. Nonetheless, as most of the exploits are released before or at the same day as CVE disclosures, an exploit-centric risk assessment would also be valuable. Ultimately, a combined perspective should capture risk associated to exploits for which CVEs have not been released yet, together with risk dynamics associated to disclosed CVEs.

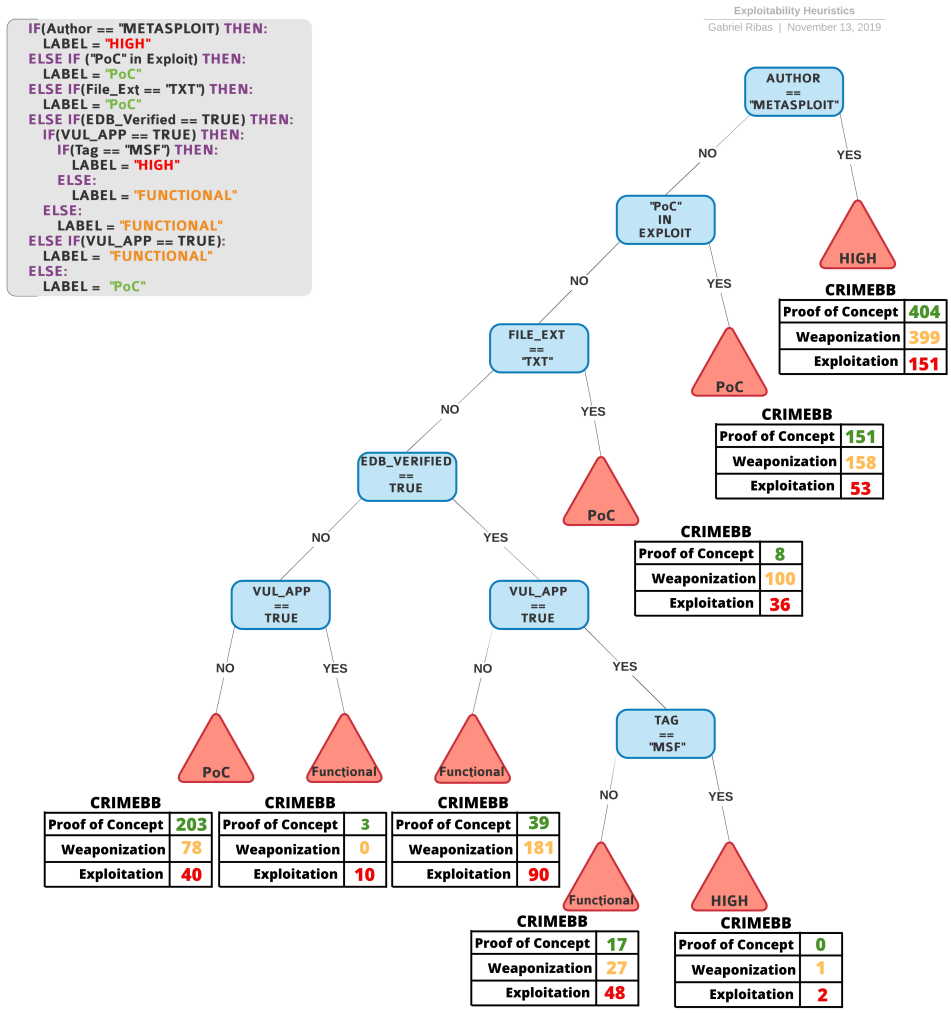


Fig. 17. Decision tree to classify vulnerabilities according to their exploitability level and compare NVD and ExploitDB against CrimeBB.

Vulnerability classification criteria. The methodology introduced in this paper for risk assessment involves the classification of CVEs as a function of their time to weaponization. Alternatively, clustering techniques could be used to search for similar CVEs. Once a new CVE is released, its corresponding cluster could be used to assess a typical time to weaponization.

Additional aspects involved in risk assessment. In this work we considered weaponization as the major drive for risk increase. Nonetheless, risk assessment involves a number of other factors [12], such as the number of vulnerable hosts in the network and the maturity of patches. The larger the number of vulnerable hosts, the greater the incentives for hackers to develop exploits for a vulnerability. At the same time, for old vulnerabilities the few hosts that remain vulnerable after long periods of time may be easy targets of *script kiddies*. The risk associated to a given vulnerability,

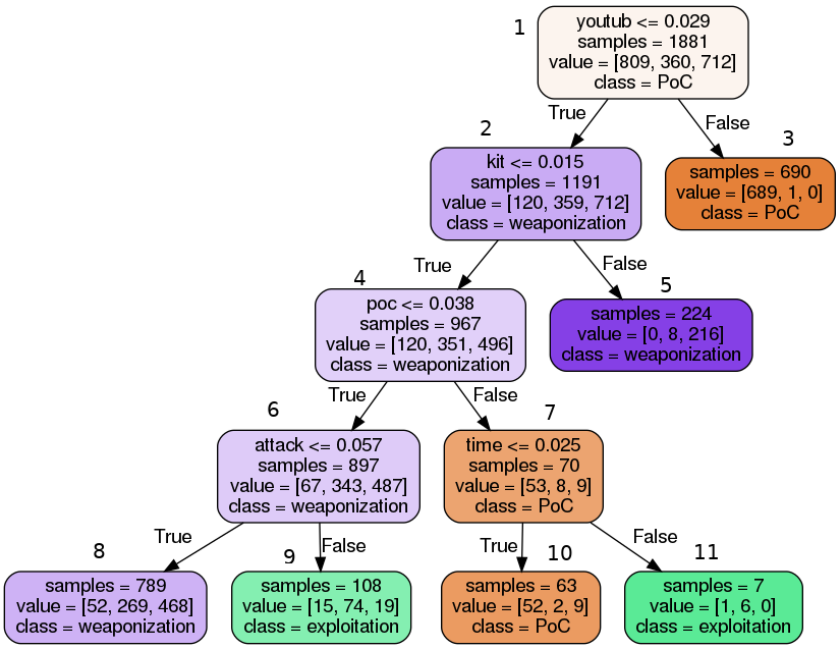


Fig. 18. Tree with accuracy 77% separating classes: PoC, weaponization and exploitation

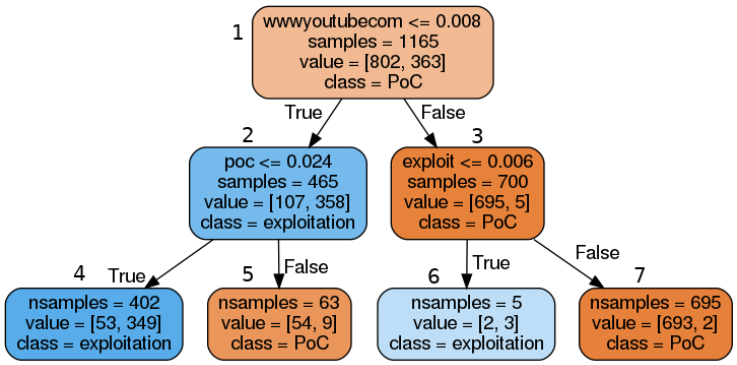


Fig. 19. Tree with accuracy 89% separating classes: PoC and exploitation

therefore, depends not only on the state of the host, but also on the state of the whole population, giving rise to strategic aspects which are subject for future work.

Stages in vulnerability lifecycle. We have focused on the time to weaponization as the major stage in the vulnerability lifecycle. Additional stages include, for instance, the time towards the first effective exploitation (incident). Nonetheless, the amount of public data available about such incidents including timing at the granularity required for our purposes is very scarce [13]. If such information is provided, one can easily extend the model presented here, e.g., using a semi-Markov model, to account for multiple risk level stages. Then, the first stage corresponds to the first weaponization and the second stage corresponds to a successful exploitation.

10 CONCLUSION AND FUTURE WORK

The Common Vulnerabilities and Exposure (CVE) dictionary is the *de facto* database of software vulnerabilities. Coupled with the CVSS, it provides extensive information about risk levels associated to different threats, and is designed to account for temporal and environmental aspects. Nonetheless, public databases fall short on providing temporal information about how risks evolved over time. Such information, if available, could help not only for explanatory but also for predictive purposes. In this paper, we presented measurements and models to empirically assist in the tracking of risk scores. We believe that such effort is an important first step towards risk-aware patch management. This, in turn, is particularly critical for IoT and ICS devices whose cyber-physical nature precludes too frequent fixes.

As future work, we plan to (i) use indicators of compromise (IoC) for exploitation prediction and (ii) report ground-truth of CVSS score timelines based on events related to vulnerabilities reported at NVD, using heuristics to match those events to CVSS subscores.

REFERENCES

[1] Elisa Bertino, Kim-Kwang Raymond Choo, Dimitrios Georgakopolous, and Surya Nepal. 2016. Internet of Things (IoT): Smart and secure service delivery. *ACM Transactions on Internet Technology (TOIT)* 16, 4 (2016), 22.

[2] Benjamin L Bullough, Anna K Yanchenko, Christopher L Smith, and Joseph R Zipkin. 2017. Predicting Exploitation of Disclosed Software Vulnerabilities Using Open-source Data. In *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*. ACM, 45–53.

[3] Hasan Cavusoglu, Huseyin Cavusoglu, and Jun Zhang. 2008. Security patch management: Share the burden or share the damage? *Management Science* 54, 4 (2008), 657–670.

[4] Michel Edkrantz, Staffan Truvé, and Alan Said. 2015. Predicting Vulnerability Exploits in the Wild. In *Cyber Security and Cloud Computing (CSCloud), 2015 IEEE 2nd International Conference on*. IEEE, 513–514.

[5] Stefan Frei, Martin May, Ulrich Fiedler, and Bernhard Plattner. 2006. Large-scale vulnerability analysis. In *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*. ACM, 131–138.

[6] Christos Ioannidis, David Pym, and Julian Williams. 2012. Information security trade-offs and optimal patching policies. *European Journal of Operational Research* 216, 2 (2012), 434–444.

[7] Pontus Johnson, Dan Gorton, Robert Lagerström, and Mathias Ekstedt. 2016. Time between vulnerability disclosures: A measure of software product vulnerability. *Computers & Security* 62 (2016), 278–295.

[8] Dale Paterson. 2019. ICS Security Patching: Never, Next, Now. (2019). https://www.linkedin.com/pulse/ics-security-patching-never-next-now-dale-peterson.

[9] D.A.L. Pfleger and A. Avritzer. 2017. Patch management for industrial control systems. (March 8 2017). https://www.google.com/patents/EP3139318A1?cl=en EP Patent App. EP20,160,185,736.

[10] Jukka Ruohonen, Sami Hyrynsalmi, and Ville Leppänen. 2015. The sigmoidal growth of operating system security vulnerabilities: an empirical revisit. *Computers & Security* 55 (2015), 1–20.

[11] Jukka Ruohonen, Sami Hyrynsalmi, and Ville Leppänen. 2016. Software Vulnerability Life Cycles and the Age of Software Products: An Empirical Assertion with Operating System Products. In *International Conference on Advanced Information Systems Engineering*. Springer, 207–218.

[12] Jukka Ruohonen, Sami Hyrynsalmi, and Ville Leppänen. 2016. Trading exploits online: A preliminary case study. In *Research Challenges in Information Science (RCIS), 2016 IEEE Tenth International Conference on*. IEEE, 1–12.

[13] Verizon RISK Team. 2015. 2015 Data Breach Investigations Report. (2015).

[14] Neal Wagner, Cem Şafak Şahin, Jaime Pena, James Riordan, and Sebastian Neumayer. 2017. Capturing the security effects of network segmentation via a continuous-time markov chain model. In *Proceedings of the 50th Annual Simulation Symposium*. Society for Computer Simulation International, 17.

[15] Brandon Wang, Xiaoye Li, Leandro P de Aguiar, Daniel S Menasche, and Zubair Shafiq. 2017. Characterizing and Modeling Patching Practices of Industrial Control Systems. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 1 (2017), 18.