Vulnerability Forecasting: Theory and Practice

Article *in* Digital Threats Research and Practice · November 2021

DOI: 10.1145/3492328

CITATIONS
3

READS
224

3 authors, including:

Eireann Leverett
Concinnity Risks
26 PUBLICATIONS **98** CITATIONS
SEE PROFILE

Matilda Rhode
Cardiff University
11 PUBLICATIONS **430** CITATIONS
SEE PROFILE

Vulnerability Forecasting: theory and practice.

ÉIREANN LEVERETT*, MATILDA RHODE*, and ADAM WEDGBURY, Airbus, U.K.

It is possible to forecast the volume of CVEs released within a time frame with a given prediction interval. For example, the number of CVEs published between now and a year from now can be forecast within 8% of the actual value. Different predictive algorithms perform well at different lookahead values other than 365 days, such as monthly, quarterly, and half year. It is also possible to estimate the proportions of that total volume belonging to specific vendors, software, CVSS scores, or vulnerability types. Some vendors and products can be predicted with accuracy, others with too much uncertainty to be practically useful. This paper documents which vendors are amenable to being forecasted. Strategic patch management should become much easier with these tools, and further uncertainty reductions can be built from the methodologies in this paper.

Additional Key Words and Phrases: cyberrisk, forecasting, prediction, CVE, vulnerabilities, vulnerability management

1 MOTIVATION

Most computer security vulnerability management problems are driven by reactivity to the disclosure of a Common Vulnerability Enumeration (CVE) or the release of a patch. The cadence of patch release is usually correlated with the release of CVEs, while the patch installation is dictated by business risk decisions and often follows a different cadence. The release of CVEs is commonly viewed as stochastic, unpredictable, and difficult to foresee before they emerge, and we seek to challenge that view throughout this paper.

It is possible to derive an expectation of exploit existence as a probabilistic posterior from CVE disclosure: recent analysis has found that 80% of exploits are written before the CVE is published, by 23 days on average [Chen et al. 2019][Suciu et al. 2021]. This resonates with the authors' experience that a basic exploit often has to be written to convince the company of the need for a CVE disclosure in the first place. The only question is whether such exploits are released more publicly in frameworks such as Metasploit, semi-publicly on platforms such as exploitDB; or semi-privately in exploitkits, or completely private as a company that discovers it internally. This exploit prediction work motivated us that CVEs too, might be quantitatively predicted.

We posit that vulnerability management could be much improved by predictive tools such as prediction of the volume of CVEs being produced within discrete time spans (ideally, this would be user-specified time spans). If those forecasts are accurate (plus or minus some reasonably small confidence or prediction interval), then three discrete things are accomplished:

(1) *Security Engineering Logistics Provisioning* The size of a patching team and resources can be better estimated and budgeted for **in advance**. Essentially answering; 'Is our team growing in pace with the volume of vulnerabilities they must handle next year?' This is not only a people management issue, but also one of bandwidth, uptime availability, and change management. Many large companies spend millions on these

*All authors contributed equally to this research.

Authors' address: Éireann Leverett, eireann.leverett@airbus.com; Matilda Rhode, matilda.rhode@airbus.com; Adam Wedgbury, adam.wedgbury@airbus.com, Airbus, Quadrant House, Celtic Springs Business Park, Coedkernew, Duffryn, Newport, U.K., NP10 8FZ.

2576-5337/2022/1-ART1 $15.00

problems; the vulnerability management industry itself was estimated at 12.5 billion USD in 2020 with a compound annual growth rate (CAGR) of 4.5%[1].

(2) *Board-level cyber risk communication* Our inability to forecast earthquakes leaves us exposed to risks we cannot predict, whereas a 48-hour advance warning of hurricanes allows evacuation orders. Similarly here, telling the Chief Finance Officer that resources are required for the increased forecasted CVEs this year is significantly different than finding out each day how many you must patch. What distinguishes this from engineering and team management is the ability to amortize the risk over longer time periods by producing projective cost-benefit scenarios, which can only be done with significant prediction window increases. A probabilistic model of CVE forecasting can capture a way to quantitatively communicate how likely we are to get 5 vulnerabilities above CVSS 7 in a single month. If you know the cost of patching and the cost of exploitation, you can explain cyber risk to the board in language used for other types of quantitative risk.

(3) *Software supply chain dependency* Imagine you are deploying a medical device that will not be rexamined by a doctor for many years. It needs to operate for 8 years and will only be patch-able twice within that operational lifespan. This scenario may be extended to any estate with operationally defined maintenance windows (I.E. many operational technology environments). You are confident that your own software engineering teams have software engineering quality to meet your specifications, but what about the operating system or libraries outside your control? The ability to forecast or even estimate the number of patches per year would help very much in your operational planning, especially with bandwidth and availability at such a premium forecasting expected vulnerabilities can help to predict the human resources likely to be required for a given maintenance window. This is not to suggest you would prioritise the system that has less security vulnerabilities, but rather the one that releases them in timely and consistent manner. Especially if their patch release cycle synchronises to your diagnostic windows every 3 years. In other words, the system that has less variability or uncertainty around how many patches would be installed is preferred, not necessarily the one with less vulnerabilities.

(4) *Cyber insurance* The number of vulnerabilities released each year has a significant impact on cyber insurers portfolios. In particular those that are network facing, in common CPEs, and have weaponised exploits. The ability to forecast the volume of vulnerabilities is the first step to putting a price on the impact of those vulnerabilities across millions of businesses worldwide. Another possibility may emerge with using predictions as a parametric trigger for insurance linked securities[2].

(5) *Equities processes* The ability to decide if a vulnerability has value for offensive or defensive purposes rests in some sense of how scarce it is, how widespread it is, and how impactful. When evaluating a particular vulnerability, one can use the quantitative background for forecasting to rate or score a vulnerability according to these factors as a measure of scarcity. The other aspects of how often the vulnerability can be found in the wild or how damaging it can be are not covered by the work in this paper, but the element of scarcity is captured through a quantitative measure of the historical record.

Additionally, better forecasting begets better forecasting. Once we could predict the raw volume of CVEs in a discrete time frame we found other uncertainty reductions for related forecasts began to follow more easily. In other words, forecasting Firefox as a subset of all CVEs this year is easier than forecasting just from Firefox data alone. In fact, the more sub-categories you can forecast the less of the total is left uncertain.

Many other cyber risk management problems become significantly easier because we have more time to plan ahead and be less reactive; essentially shifting risk-based decision making to an earlier point in the timeline, without impacting the quality of the decision itself. A projection of increased vulnerabilities over the next year may help risk managers make the case for an increase in resources using quantitative data.

[1] https://www.marketsandmarkets.com/Market-Reports/security-vulnerability-management-market-204180861.html
[2] https://content.naic.org/sites/default/files/capital-markets-primer-linked-securities.pdf

A future application of CVE forecasting will also be found in the conditional probabilities; such as 'If the next CVE in this software is a memory corruption, what is the probability that exploit is developed within 3 months?' or 'If this software becomes deployed on 25% of internet-facing servers, what chance is there that an exploitable vulnerability will be found in it?'. To achieve any of that, forecasting and risk analysis of the CVE data, public disclosures, exploits, and operating systems, is necessary. Thus we believe that the work we have done reducing the uncertainty of how many CVEs we will see in the near future is both well-motivated, and foundational to future cyber risk questions.

This paper is organised as follows: the next section provides an overview of the contributions of this paper followed by an overview of background literature in Section 3. Section 4 outlines the methodology used, section 5 introduces the dataset underlying the models and the results are presented in Section 7. Section 8 discusses some limitations of the proposed approach and finally, Section 9 concludes the findings of the paper and discusses potential applications.

2 KEY CONTRIBUTIONS

The key contributions of this work are summarised below:

(1) A publicly available **leading indicator** that has not been identified in previous work, which improves machine learning model accuracy.
(2) **Two statistical models** are applied to this problem which have not been documented in previous work and produce accurate long-term predictions (Section **??**), which outperform all other predictive models tested for long-term (1 year) predictions. Each of these models relies on metadata, therefore having two builds some security through redundancy in case one of the metadata models is changed.
(3) A set of models for predicting the number of vulnerabilities in both the **short- and long-term**, forecasting within 8% of the true values for 1-year lookahead predictions on average. Previous work has focused on short (1 to 3 months) or long term (1 or 2 years) time periods only, when patch managers are likely to want to analyse both next month's and next year's predictions (Section **??**). Our improvement of 8% percentage error for yearly predictions improved on previous work which achieved 15% percentage error.
(4) A flexible framework to predict the number of **total vulnerabilities or a specific subset**, using an appropriate model for the task. Previous work typically looks at the number of vulnerabilities for *either* specific products *or* all CVEs, our model is appropriate for predicting all CVEs and some sub-types (Section 7.5).
(5) This work acknowledges that we may want to predict CVEs in products that have never had a CVE before. We document some of the challenges of predicting less commonly seen sub-types of vulnerabilities accordingly (Section 7.5).

There are various different types of forecasting for security vulnerabilities. In this paper, the focus is primarily on forecasting *publicly disclosed vulnerabilities*, that might impact a user of any software or firmware. Alternatively, one can think of it as the rate of vulnerability publication (with respect to the NVD). The aim is to translate that prediction and forecasting capability to optimise minimum resources for maximum risk reduction in vulnerability management. However, we take a broader view in the literature below to overview other aspects of the research because it also offers insights.

3 BACKGROUND AND RELATED WORK

As an organising principle, there are three broad streams of vulnerability forecasting research.

(1) **Existing network vulnerabilities** Given previous network or single host vulnerability scans, forecast how many previously known CVEs will be next found, and the density of vulnerabilities in different targets, or more simply: the rate of change between host or network scans.

(2) **Future software vulnerabilities** Given source code or a compiled binary, forecast how many vulnerabilities there might be, where they are, or how much effort it might take to find them.
(3) **Future vulnerabilities: all types** Given data on historical Common Vulnerability Enumerations (CVEs), use machine learning to estimate the volume of published vulnerabilities in a given time frame, and/or resource requirements to patch them. Given further information such as Common Vulnerability Scoring System (CVSS) or knowing an exploit already exists, estimate difficulty to patch and/or risk reduction from having done so.

In the literature review that follows the papers are grouped into these three categories.

3.1 Stream One: Forecasting the Vulnerability of Specific Computers or Networks

This branch of academic literature concerns itself with the prediction of the vulnerability of a given machine or network. Sometimes it is even more specific, such as the presence of a particular Common Vulnerability Enumeration (CVE) on an Internet Protocol (IP) address or range of addresses e.g. [Venter and Eloff 2004]. Or, closely related to the CVE forecasting work in this paper, estimating the number of unpatched vulnerabilities in an organisation based on the products in use and Bayesian inference [Chatzipoulidis et al. 2015] or fuzzy sets [Aleksić et al. 2014]. This seems more applicable to offensive security teams than defensive security teams, though perhaps they could use it as a metric to optimise patching strategies. This may have a stronger application where scanning is likely to miss results due to computers joining and leaving the network (university WiFi or conferences) or being down for maintenance (OT Networks). In short, where the network's graph of connections could be considered dynamic in a temporal sense, this type of prediction may be very useful.

3.2 Stream Two: Pre-disclosure Vulnerability Volume Estimation

There is plenty of literature on predicting the number of undiscovered/undisclosed vulnerabilities in software. Some of that literature focuses simply on the numbers [Petersson et al. 2004; Scott and Wohlin 2008; Vander Wiel and Votta 1993], and other papers on how much effort it takes to find them and what improvement may or may not be achieved [Brady et al. 1999; Ozment and Schechter 2006; Rescorla 2005]. Some even make forays and efforts into the value of predictability and forecasting [Alhazmi and Malaiya 2006]. The primary aim and application of this type of research is the prevention of vulnerabilities during the software development lifecycle. In other words, preventing the vulnerabilities from reaching a production system and getting assigned a CVE at all.

Another distinctive sub-field of this research advocates becoming active; one can deliberately inject vulnerabilities to be able to figure out the rate at which your security testers find them [Schuckert 2016]. Alternatively, one could use a security code review team to find vulnerabilities, then not fix them, and see if the quality assurance team finds them again to produce an estimation of how many have not been observed (Lincoln Index) [Geer 2015]. Such an index allows you to estimate the vulnerabilities remaining in a static codebase, which may or may not have value depending on the velocity of your development team; if your development team is changing the code too quickly, your fixes may not have much value as they inject more vulnerabilities at the same time.

3.3 Stream Three: Post-Disclosure Vulnerability and Exploit Forecasting

This is the stream of greatest relevance to this paper and explores the predictive power of estimating the number of vulnerabilities that are publicly disclosed as CVEs.

The primary application of such research would be in managing patching teams, tools, and networks, exploring patch strategies, and thinking about third party software interactions.

Yasasin et al present an excellent systematisation of knowledge paper, that also demonstrates some promising and practical empirical results for up to three-month lookaheads [Yasasin et al. 2020][3]. The paper surveys the literature much more broadly than we do here and provides many other points of departure for the interested reader. It compares multiple time series forecasting models to predict the number of vulnerabilities for specific vendors. In particular, if the model under construction is meant to be a vendor or product-specific one (I.E. all Microsoft product disclosures, or Mozilla:Firefox disclosures), for which there may be many time periods in which there are zero disclosed vulnerabilities. The authors caution against the use of relative metrics such as percentages, and we didn't use them in our own work below. The authors detail how to test the predictive power of various time series models and report that Croston's and ARIMA forecasting models tend to perform best but do not select an overarching methodology for predicting the number of vulnerabilities attributed to *any* product. Further, the work is limited to 1, 2 and 3-month forecasts "Since forecasting research has shown that long-term forecasts are generally limited in their predictive accuracy", whereas here we chose to predict up to one year in advance with no accuracy reduction to that of their 3-month predictions. Our work builds on this paper by using a more robust testing methodology where strict date separation is enforced between the training and test set to better-mimic real-world use (more details of this in Subsection 4.1).

Another inspiring paper revisits predicting product-specific disclosures for Operating Systems [Pokhrel et al. 2017]. This one details both ARIMA and an approach using Artificial Neural Networks (ANNs). The authors predict numbers of CVEs for 12 single months, sums them, and compares to the yearly total for the test set, but this is not the same as predicting the next year's total CVEs since the model has information about what has happened so far in a year (e.g in July) to predict the total number appearing between January and December. These increasing forecast projections are decreasingly accurate, but might still be accurate enough for resource planning. Here the authors observe strict date splitting, only testing on data from 2016 and training on data up to the end of 2015. The confidence intervals are defined for the predictions, which also assist the astute risk manager. However, we note that prediction intervals would have been a better choice to represent uncertainty boundaries than confidence intervals in this respect as the former capture both the uncertainty around the mean prediction *and* around the variance (thus the prediction interval is always wider than the confidence interval).

Last [Last 2016] predicts broader categories: all vulnerabilities, Browser CVEs, Operating System CVEs, and Video Codec CVEs. This approach uses root mean squared percentage error (RMSPE) as a validation metric (which the deep reader remembers was discouraged by [Yasasin et al. 2020] in the cases of zero-inflated time series); and notes the occasional limited success of using software release dates alongside other data to aid in the forecasting accuracy. This work predicts 12, 18 and 24-month windows, using strict date separation, with a testing window from 2012 to 2015, achieving a median 15% RMSPE for global vulnerability predictions over 24 months, the work presented herein is able to significantly improve this metric for long-term global predictions. Last's work reveals an increasing accuracy as the forecast window increases from 12 to 24 months, this is a result we have found to extend to short term predictions: the total number of vulnerabilities next year is easier to predict than next month's. However, this approach predicts the cumulative total CVEs for all time, such that the percentage error metrics cannot be considered consistent over time, furthermore, the cumulative totals i.e. how many CVEs there have been since 2001, are unlikely to be useful to patch managers.

Wang et al. [Wang et al. 2008] provide a method to optimise response to demand volumes. This provides a solid foundation for optimising our response to the numbers of vulnerabilities a potential model may predict but, more discussion is needed about the variability of response to differing numbers of patches. Useful as this paper is, it is not specific to our domain of publicly disclosed vulnerabilities, though it does detail an interesting

[3]The paper finishes: "To conclude, we hope that our study encourages academics to develop suitable forecasting methods for IT security vulnerabilities which subsequently improve prediction accuracy. ", and we wish to acknowledge that their paper did exactly that in inspiring the test methodologies and approaches we developed here.

method for making unsupervised hybrid models by scaling horizontally with more processing power. In our own research, we have not found processing power to be a limiting factor.

One paper notes that vulnerabilities follow market share [O'Donnell 2008], while another warns us to 'mind the denominator' [Jardine 2018]. For our purposes that might mean using the full Common Platform Enumeration (CPE) dictionary to look at probabilities for vulnerabilities, instead of only vulnerability databases. Indeed we discuss later in the paper the challenge of predicting CVEs for products that have never appeared in the CVE List before, and that is where using the CPE might help in some cases.

It is important to scour amongst hacker literature as well as academic papers. Many academics hide their light under a bushel and present very practical results at hacker conferences such as BlackHat or Chaos Communication Congress. Allodi, for example, has captured a very important piece of information security risk reduction measurement in this presentation of the perils of CVSS as a risk quantification[4]. The gist of it is that most vulnerabilities don't have exploits, and don't get exploited. Improving the *specificity* (false positive for exploited vulnerabilities) of our work would save us a lot of patching effort. Allodi has also published a related paper on the distribution of exploitation per CVE[Allodi 2015].

An intellectual offspring of Allodi's work was performed by Cyentia Institute and Kenna Security[5]. Here they look at a wider dataset of exploited machines in IDS traffic, trying to learn from what gets exploited and extrapolating towards better patching. That work has now become a special interest group at FIRST.org for Exploit Prediction and Scoring Systems (EPSS)[6].

Previous work into vulnerability forecasting has demonstrated that it is possible to predict the numbers of all expected CVEs or subsets of CVE categories for a few months or a couple of years. However previous literature has failed to unify a framework that is capable of spanning both. In this paper, we present a model to predict both short- and long-term numbers of vulnerabilities using publicly available data from MITRE that has not previously been written about for this purpose.

This stream also has applications to supply chain management and passive vulnerability scanning. It may not be intuitive that forecasting is useful to risk management even when you have *no intention* of patching as a risk treatment. To illustrate that, consider that CVE forecasting could answer questions of the form: A submarine is launched with vendor X's product; knowing it can not be patched for a year due to cost/battery/bandwidth reasons, how many vulnerabilities can be expected to emerge in Vendor X's product within that year of service? Another way of looking at the same problem might be: Estimate how often I need maintenance downtime to keep Vendor X's products patched above CVSS 7 this year. With this in mind, what data can we use for prediction and forecasting, to move towards answering such questions?

4 METHODOLOGY

This paper seeks to predict the number of CVEs as a whole as well as sub-types of CVEs (e.g. particular products, CVSS severity, attack vector) across different time-spans (1 month, 3 months, 6 months, 9 months, and one year). It may be that different models are better suited than others to predict certain sub-types over certain periods of time-frames. Predictive modelling may perform well in laboratory tests but over time, underlying changes in the distribution of data known as concept drift may make forecasts less accurate and reliable. Moreover, the best model for predicting may change over time.

This paper proposes a methodology, see Figure 1 to solve both of these problems simultaneously in a robust way, where robustness refers to the ability to handle changes in the underlying distribution of data.

[4]https://media.blackhat.com/us-13/US-13-Allodi-HOW-CVSS-is-DOSsing-Your-Patching-Policy-Slides.pdf
[5]https://i.blackhat.com/USA-19/Thursday/us-19-Roytman-Predictive-Vulnerability-Scoring-System-wp.pdf
[6]https://www.first.org/epss/

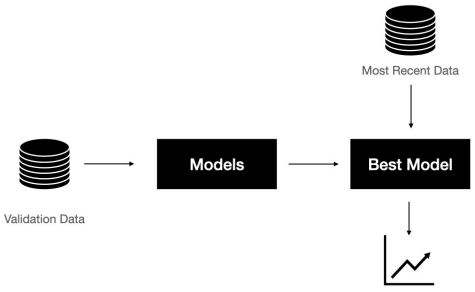


Fig. 1. High level overview of prediction methodology

The first problem is how to handle many different distributions of data: subtype vs subtype, 1-month predictions vs 1-year, 1-year prediction in 2015 vs 1-year prediction today. This issue is tackled by allowing multiple different modelling strategies to compete and selecting the best one. The 'best one' is the model which scores the lowest prediction interval on a recent validation set. Prediction intervals, like confidence intervals, assert likelihood bounds around a prediction being correct. Unlike confidence intervals, prediction intervals must also account for the variance over time in a forecast. Let σ be an unbiased estimate of of the standard deviation calculated by $\sqrt{\frac{\sum (Y-\hat{Y})^2}{N-2}}$, and z a confidence interval z-score e.g. 1.96 for 95% confidence interval, then the prediction interval is $\hat{Y} \pm z * \sigma$ for a validation set. In essence, this means that the best model is the one that has seen the lowest average deviation from the validation data. It is possible to swap this metric for another, as is indeed tested below in experiments, this is to allow flexibility to the user and because it is perfectly possible to configure the model as such.

The second problem is robustness over time: what if the underlying structure becomes effectively unpredictable, how will the user know not to rely on even the 'best model'. This is also tackled by using prediction intervals. Prediction intervals act as a guide to the user as to how well the model is performing. If the model predicts 2,000 vulnerabilities with a 90% prediction interval of ±1800, the user should be able to judge whether or not the model is providing useful insights (if there is only capacity to patch or manage 200 vulnerabilities then it may in fact be useful!).

To critique our own approach the inherent weakness in this methodology is the reliance on recent past performance i.e. performance on a validation set. If there is a significant change in periodicity, volume, or variance starting today, the model is unlikely to be able to capture it unless it has been through a similar change during the validation period. Despite this drawback, we still believe that this methodology improves upon previous research by creating a framework to handle a variety of input data and changing input data.

4.1 Model Evaluation

This section presents the evaluation methodology used in this paper: retrocasting. Retrocasting is used for evaluating model accuracy and underlies the goal of this paper which is forecasting. Forecasting is used for resource management and planning, i.e. real-world use. **Retrocasting** iteratively pretends we are somewhere in the past trying to predict the number of CVEs in the future, this rolling window approach allows us to approximate the performance of our model when it is deployed. Retrocasting predicts the number of CVEs seen in a fixed period of time (e.g. one month) for multiple "end" dates (end of the lookahead window) in the test set, here the

end dates are staggered at 10-day intervals between Sept. 2018 and Sept. 2020. Retrocasting fixes the lookahead (the window of time we are trying to predict) but changes the end date and use the average accuracy metrics as an indicator of future performance. Empirically we verified that it is possible to be more accurate on some longer time windows, than on some shorter ones. The retrocasting window used throughout this paper are 24 measurements taken between January 2019 and January 2021. **Forecasting**, by contrast, uses a fixed window. This is fixed from date from which we are trying to predict and changes the lookahead to give estimates for different time periods into the future. The purpose of forecasting is that for use in resource management, patch maintenance teams may want a short-to-long-term view of the number of vulnerabilities being seen. Previous work has indicated that a range of machine learning and statistical paradigms, some capable of time series analysis are superior to others in their forecasting capabilities. For comparison with prior work and to ensure that the most accurate model is being used for each forecast context (lookahead and CVE sub-group) a number of evaluation metrics are collected to determine the best model.

5 DATA USED

As with previous work in vulnerability forecasting [Last 2016; Pokhrel et al. 2017; Yasasin et al. 2020], this research uses the US government's National Vulnerabilities Database (NVD) with a goal to predict the number of vulnerabilities (matching certain criteria) that will be published to this database within a given future time span (lookahead window). The NVD data itself can be used as a signal for this prediction but is more effective when augmented by additional data from MITRE; the organisation that maintains and publishes the CVE-list, which feeds the NVD.

During data cleanup and preparation, we performed the standard times series and forecasting analysis (Fig 2, first-order differencing (Fig 3), and detrending (Fig 4), to make a stationary time series. On examination, it appeared it was unlikely prediction could succeed with these methods beyond 3 months, where the partial auto-correlation falls below a noise floor of the variance (Fig 5). However, once we found the MVUE (Section 6.2) and Little's Law (Section 6.3) approaches it became possible to extend practical forecasting (+/- 5 percent uncertainty) beyond this natural limit to a year ahead.

This alone is a significant finding, but we have also shown that the data support strong prediction intervals at different timescales, and shown that some products are ripe for forecasting, while others have too little data or don't decompose well under traditional time series analysis.

5.1 NVD

The NVD maintains detailed information about CVEs including but not limited to:

(1) date published
(2) the organisation reporting the CVE (known as a CVE Numbering Authority or CNA)
(3) a text description
(4) Strings indicating specific entities impacted by the CVE (Known as Common Platform Enumerations or CPE strings)

CVEs are identified by a CVE-ID string in the format: CVE-YEAR-XXXX. During this research, it became evident that the YEAR in the CVE-ID did not match the publication year in all instances. This is because the CVE-ID reflects the year in which the vulnerability was *assigned* rather than published. Figure 6 illustrates the disparity between publication year and CVE-ID year, some early CVEs were added retrospectively even assigned a CVE year later than they are published but this is an anomalous attribute of the early entries in the database. Every CVE is assigned prior to being published, sometimes years in advance, sometimes as little as an hour. The XXXX part of the ID corresponds to a serial number thus allowing us to infer that on publication of CVE-2020-0100 there are unpublished CVE IDs reserved for CVE-2020-0001 to CVE-2020-0099 even if they have

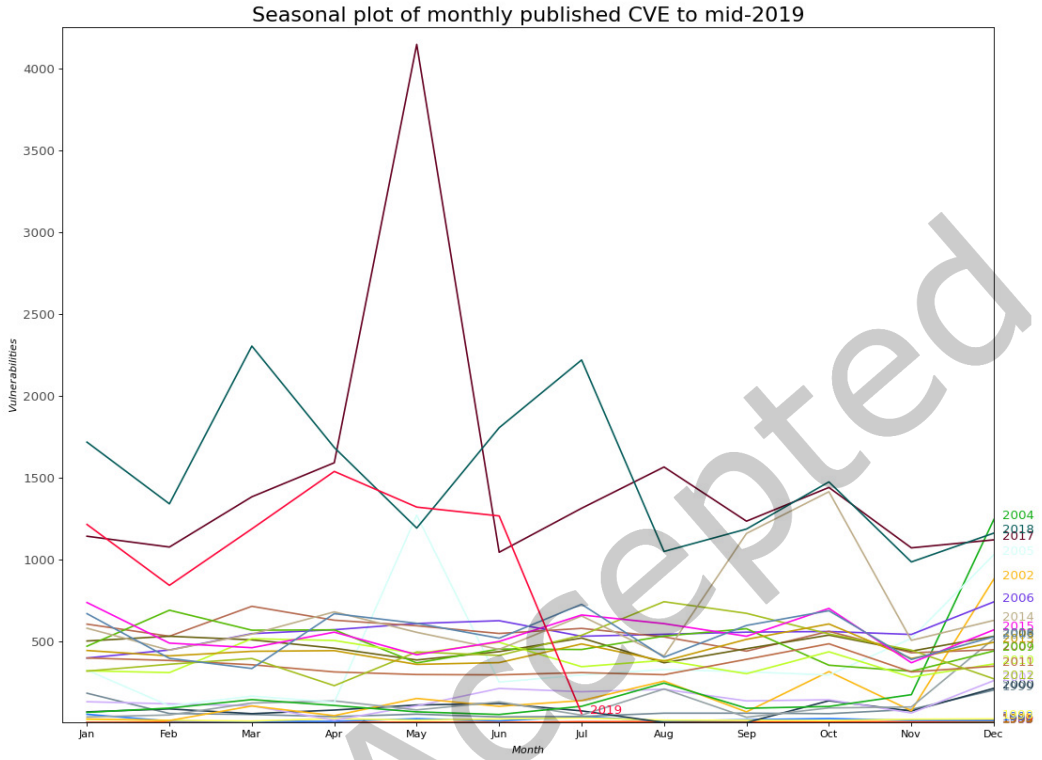


Fig. 2. Seasonality analysis of vulnerability publication volumes

not been published[7]. Thus we are able to find that N CVEs are in the pipeline, which improves the accuracy of predictive models. This latent data is the foundation of one of the models presented in this paper in 6.2. In short, CVEs are ordered on arrival, but released stochastically according to progress internal to MITRE. That ordering allowed us an initial insight into how long they take to process, later supplanted by the CVE list we discuss below.

5.2 MITRE

Every CVE-ID is assigned by MITRE or one of their approved CNAs[8]. The specific assigned dates are published by MITRE [9], which gives much better granularity of the assigned date than using the CVE-IDs. MITRE provides the assigned date whilst the NVD catalogues the date published. Research from Chen (Palo Alto Networks) [Chen 2020] asserts that 80% of CVEs have a publicly available exploit on average 23 days before the CVE is published. Combining NVD and MITRE data reveals that the median lag between the assigned date and publication date

[7] In practice these can be more digits such as CVE-2020-XXXXX or CVE-2020-XXXXXX
[8] 144 organisations as of 5th November 2020: https://cve.mitre.org/cve/cna.html
[9] Available to download: https://cve.mitre.org/data/downloads/index.html

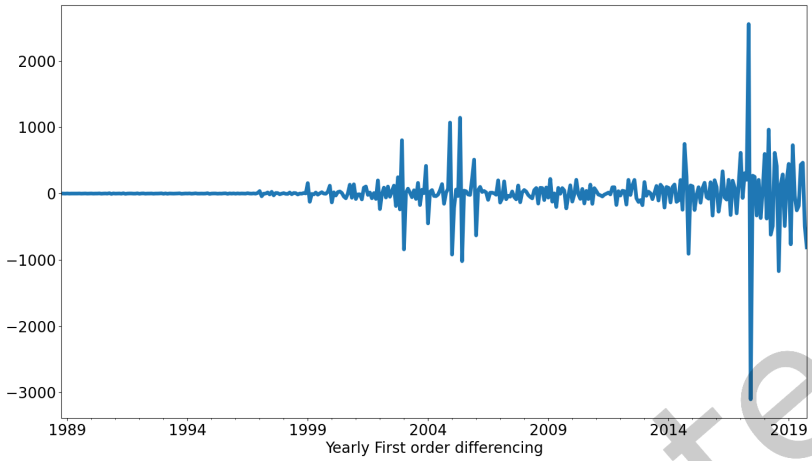


Fig. 3. 12 rolling month first order differencing.

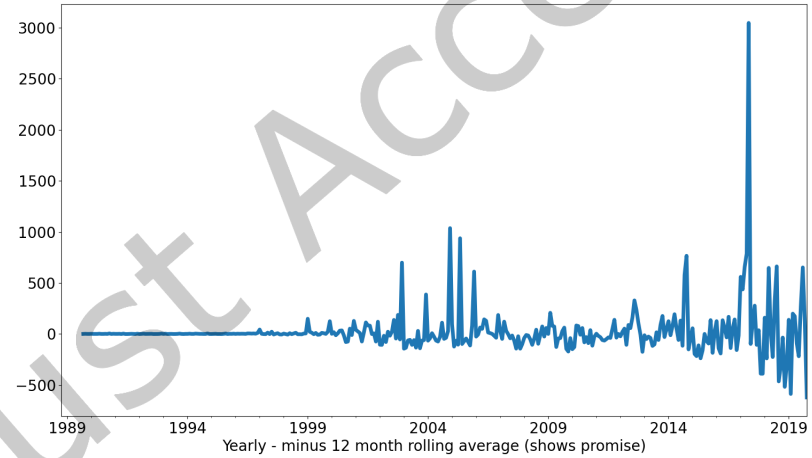


Fig. 4. 12 rolling month first order differencing minus trend.

is 51 days (see Table 1). These statistics underscore the possibility and usefulness of being able to predict the publication of CVEs at least one month in advance.

5.3 Challenges

Examining the raw data highlights some of the challenges in predicting CVE numbers. Four key challenges are briefly described below:

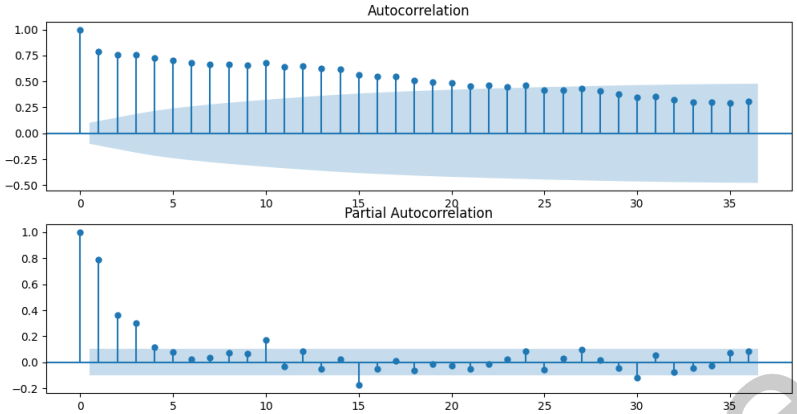


Fig. 5. ACF-PACF

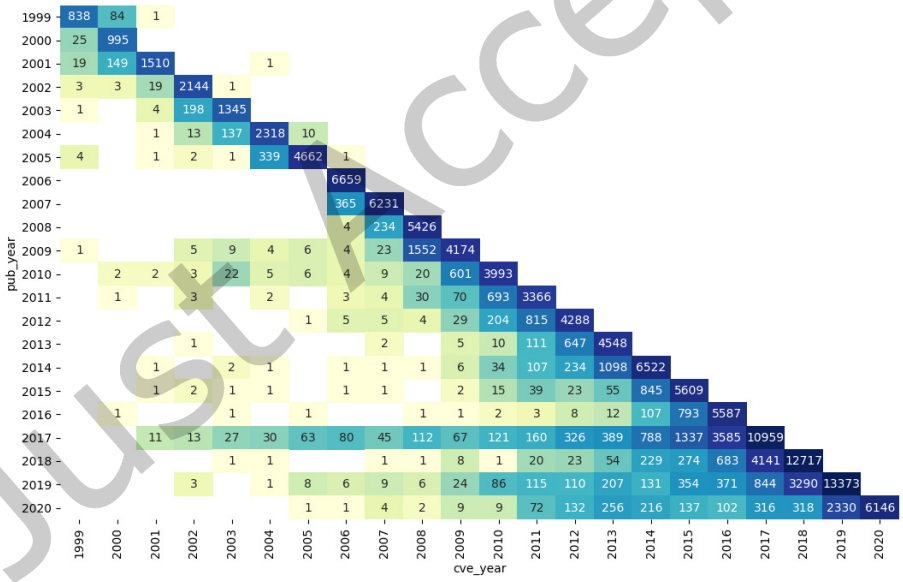


Fig. 6. Heatmap of CVE Year against Publication Year over the historical record.

Short-term variance Due to the idiosyncratic policies and practices of both CNAs and MITRE, some of which are common to all non-automated processes, sometimes an abnormally large number of CVEs are published in a single day. Figure 7) shows the normalised variance of weekly, monthly, quarterly and yearly predictions.

mean	162 days 11:50:28
standard deviation	390 days 06:14:18
min	0 days 01:02:00
1st quartile	1 days 05:17:00
median	51 days 15:57:00
3rd quartile	160 days 20:29:00
max	6263 days 15:15:00

Table 1. Summary of lags between database entry and publication of CVEs

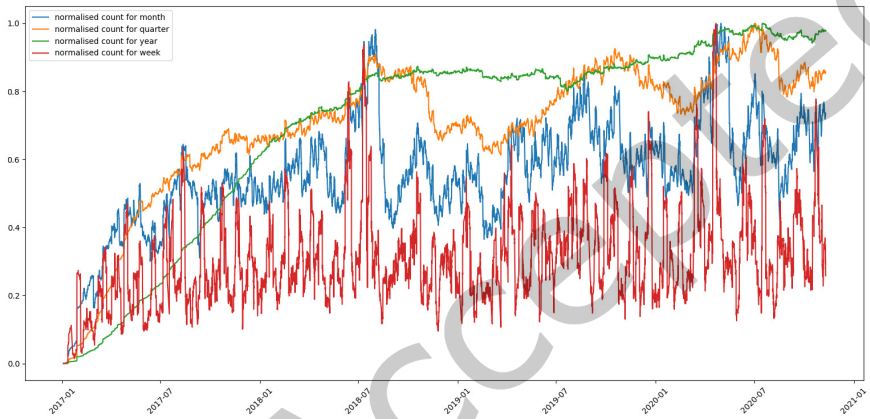


Fig. 7. Weekly, monthly, quarterly and annual total vulnerabilities January 2017 to October 2020 normalised between 0 and 1 shows greater variance as for shorter time windows.

Although there is a dramatic increase in the yearly values, in accordance with the central limit theorem (e.g. [Heyde 2006]), the variance decreases with the number of observations, which can make it easier to predict with reasonable accuracy, which is what we have found in our empirical results.

New products Some products may see a sudden, unprecedented increase in the number of CVEs. This may be because the product is new, because its market share has increased [O'Donnell 2008], or due to some other factor that is exogenous to NVD or MITRE data (Such as the automation of vulnerability finding). It is challenging for the model to predict something entirely new which has either not previously existed or has been at zero for a long time. This is a particular challenge for long-term CVE predictions.

Structural changes in CVE publication patterns Though the variance is lower for annual total CVE counts, there are clear step-changes in the numbers of CVEs being published (see Figure 8) either side of 2005 and of 2017 and to a lesser degree in 2014. This is a challenge for models using historic data to predict new vulnerability totals, one which is explored using data filtering in the experiments reported in this paper.

Assigned date is not a sufficient leading indicator for long term prediction. Whilst the assigned date will indicate how many CVEs are in the pipeline, this data is not sufficient to predict the number of vulnerabilities that will be published within a given forecast window (lookahead). This is because (1) some CVEs are rejected by

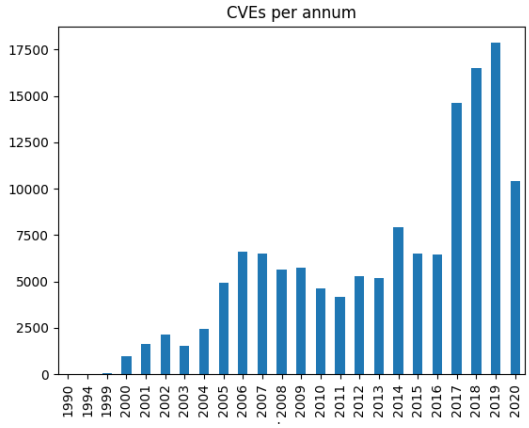


Fig. 8. Annual total CVEs published 2002 to 2020

MITRE (2) some CVE-IDs are reserved by CNAs speculatively but never populated (3) even if we know how many are in the pipeline, we do not know how many more will be assigned *and* published within the forecast window.

6 PREDICTIVE MODELS

As described above, a suite of models is trialled for forecasting. These model include both statistical models and machine learning models. The line between a statistical and machine learning model is not sharply defined. Here, we distinguish machine learning and non-machine learning models as those models capable of learning from multiple input features and those which only take the historical data of the target values as inputs.

There are several types of model tested:

6.1 Baseline models

A forecasting model is of no use if it cannot beat the most simple non-time series forecasting methods. The two baselines used here are:

(1) Previous value - take the previous year/month/week total to predict this year/month/week
(2) Moving average - take the mean over the last few years/months/weeks to predict next year/month/week

Since this is a forecasting problem, time series and forecasting models are expected to perform the best. The forecasting models used here are:

(3) Exponential smoothing - this statistical method is similar to moving average but previous values are not weighted equally, instead an exponentially decreasing weight is assigned to previous predictions [Holt 2004]. Exponential smoothing is only capable of handling univariate data.
(4) ARIMA - Auto-Regressive Integrated Moving Average (ARIMA) model (formalised in by [Box et al. 2015] Box, Jenkins, Reinsel and Ljung), ARIMA is capable of handling some types of non-stationarity in time series data. ARIMA has previously been used for vulnerability forecasting with success by [Yasasin et al. 2020] and [Pokhrel et al. 2017] for 1, 2 and 3-month predictions.
(5) Croston's method [Croston 1972], as mentioned above, attempts to rectify poor forecasts for data, which has zero values for some measurements. Croston's method has also been used by [Yasasin et al. 2020].
(6) Long-Short-Term Memory Recurrent Neural Network is a time-series machine learning model which adjusts weights through a series of layers of interconnected neurons and acts as a general function

approximator. The recurrent neural network is a time-series neural network, the LSTM variant [Hochreiter and Schmidhuber 1997] was developed to handle exponentially increasing and decreasing weights during training over long time sequences.

Machine learning models can be used to learn from data, in this instance, historic vulnerability data may be useful to predict the upcoming number of vulnerabilities, but we do not know the mathematical relationship between the data. Machine learning may uncover underlying relationships through exposure to data. The LSTM-RNN is a time series machine learning model (listed above) but there are some non-time series models which can be made to work with time-series data by feeding in historical data as individual inputs e.g. this year's total together with last year's total vulnerability count. The disadvantage here is that the model does not know the temporal relationship between the data but we test this method, in any case, considering the top-3 most commonly cited algorithms discussed by the winners on the data science competition platform Kaggle [Goldbloom 2015]:

(7) Random Forests [Breiman 2001] are an ensemble method which, once trained, groups together multiple decision trees and takes a majority vote.
(8) Gradient Boosted Machines [Friedman 2001] also rely on decision trees but trees are added during training in order to make up for the shortcomings of those trees already in the ensemble i.e. trees are combined during training rather than at the end.
(9) **Discussed above** Neural Networks have been mentioned above and a number of flavours can be used including LSTM-RNNs, which is the type that will be used here.

Further to these models, we use two additional statistical models which exploit metadata from the CVE database, namely the CVE-ID number and the reservation of CVEs prior to publishing. We refer to these models as: 'A yearly MVUE' and 'Little's law exit rate'.

(10) 'A yearly MVUE' - this approach uses a minimum variance unbiased estimator which has been used for serial-number analysis to estimate the expected maximum serial number given a set of observations. The CVE-ID numbers appear to be assigned serially each year, hence a *yearly* MVUE
(11) 'Little's Law exit rate' - is rooted in Little's law [Little and Graves 2008] which is a queuing theory equation for estimating the number of items in a queue from the average waiting time and average arrival rate to the queue. These two data are available thanks to the recording of CVE reservations by MITRE, but in this case, we are interested in how many items will be dequeued (published) rather than the number of items in the queue; hence Little's Law exit rate.

Since these two features are new, we will explain them in further detail.

6.2 A Yearly MVUE for CVEs

There is a long history of serial number prediction in academic literature[Ruggles and Brodie 1947]. The Minimum Variance Unbiased Estimator (MVUE) used in tank production prediction problems of the Second World War is detailed within [Johnson 1994]. We reproduce the original equation here:

$$\hat{y} = M + \frac{M}{Obs} - 1 \tag{1}$$

Where $\hat{y}$ is the predicted number of CVEs associated with a particular year (CVE-2020-XXXX for example), M is the maximum CVE published, and Obs is the number of CVEs published (observed) that conform to the string CVE-2020-XXXX. The CVE-ID numbers are tied to a CVE-YEAR.

The year portion is not used to indicate when the vulnerability was discovered, but only when it was made public or assigned [MITRE 2020] but we are interested in when a CVE gets published. Therefore the MVUE gives the estimated number of CVEs for a given CVE-YEAR and this is then combined with the likelihood that a CVE will be published in the time period that we are interested in where P

is a vector of probabilities $p_0, p_1, ...p_T$ that a CVE will be published for each previous CVE-YEAR and q is the fraction of a year the time period being predicted e.g. 6 months gives q=0.5. The number of observed CVEs is subtracted from the estimate since each CVE can only be published once. There are some anomalous CVE-IDs that have a huge number and do not appear to be sequential with the other numbers, any CVE-IDs above 1,000,000 were removed.
The yearly MVUE for CVEs is therefore:

$$\hat{y_T} = q \sum_{t=0}^{T-1} p_t (M_t + \frac{M_t}{Obs_t} - 1) \tag{2}$$

The probability vector P is calculated from cumulative historic data and can be seen as the likelihood that a CVE published today uses a year-ID equal to this calendar year, equal to the previous calendar year etc.

6.3 Little's Law Exit Rate

John Little's theorem, known as Little's Law [Little and Graves 2008][Little 2011], is used in queuing theory to estimate the size of a servicing queue, L from just two statistical metrics: the arrival rate into the queue, λ, and the mean service time, W such that:

$$L = \lambda W \tag{3}$$

Before a CVE is published, it is entered into the MITRE database, thus the difference between this timestamp and the publication timestamp of NVD can be seen as the service time[10]. However, unlike the general use of Little's law, we are not interested in how many vulnerabilities are being serviced but instead how many are going to *have been serviced* in a given window. For this, we adopt a variation of Little's Law to estimate the exit rate from the queue $\hat{y}$ using the probability p that the service time is within the window of interest $p(W \leq window)$. λ gives the arrival rate to the queue during the window, which we use as the baseline for the exit rate, added to this is the number of CVEs in the queue from the last period, multiplied by the inverse of the exit rate to capture those which were not published within the previous window. For example, the monthly arrival rate at the start of May was 1000 and was 800 in April and the probability of being published within 1 month of being assigned is 0.25, then the predicted number to be published by the end of May is 1000 + 800 * (1 - 0.25) = 1600 since 25% of the 800 were probably published in April.

$$\hat{y} = \lambda + N * (1 - p(W \leq window)) \tag{4}$$

The summary statistics for the entire NVD database (see Table 1 above) shows that at the time of writing the median service time is 51 days and 16 hours but the mean is 162 days implying that the data is skewed by some large outlier values, as shown by the max of 6,632 days. Some CVEs created shortly after the CVE programme was instigated in 2001 are still being published today and these CVEs can skew both the estimation of service time and the N constant since these CVEs are not necessarily being actively processed. For this reason, we set a "lookback" limit on the Little's Law exit rate model such that the $p(W \leq window)$ is calculated within a pre-specified window and N only represents items added to the queue within the previous window period e.g. if we are predicting how many CVEs will come out next year, we just look at how many are still left in the queue from

[10]In fact, our initial coding of this did not use MITRE's assigned date. Instead, it estimated the exit rate from historical data on a yearly scale. Mitre's assigned date merely adds the ability to scale this down to monthly or daily granularity.

last year and do not include the forgotten souls from the early 2000s. Succinctly, by reducing the memory of Little's Law we are less sensitive to history, and more sensitive to changes in the rates. Little's law was recently applied to CVE production times by other researchers [Miranda et al. 2021], but they did not make use of it as a predictor or measure its predictive power against other forecasting models as we have done here. Our work can be seen as independent confirmation of the applicability of Little's Law to the problem space.

6.4 Model configuration

Of the 11 models tested, some require configuration which can have a significant impact on performance, moreover, the data used may influence the accuracy of the model. To test all variations for one lookahead period took around 20 minutes, which could be run nightly. These experiments used a desktop PC with 64GB RAM, a quad-core 3GHz CPU and an Nvidia GTX 1080 GPU to training the neural network (LSTM) model. We elaborate the variations below:

6.5 Hyperparameters

For each of these models, there are a number of hyperparameters to be selected. The smallest hyperparameter search space is for the ARIMA model, which has 507 possible hyperparameter configurations. A random search of 10 configurations is used instead of a grid search since previous work has found the former method can produce better results [Bergstra and Bengio 2012]. The best hyperparameter configuration is the one with the smallest prediction interval for the validation set. Future work could increase the number of hyperparameter configurations tested, but this requires additional computational resources.

6.6 Input Features

Some models, such as exponential smoothing, can only take a single feature as input. Other models can take multiple inputs; for these models, the historic counts for the time period of interest are provided as well as the counts per week, month, quarter and year. The data required to calculate the yearly MVUE and Little's Law exit rate are also provided to the models: max CVE-ID, number of CVEs observed this year, the proportion of the year observed so far, CVE arrival rate, number in queue and probability that service time is less than lookahead. Models taking multiple inputs, therefore, use 12 features.

6.7 Lookback

For the time-series models and the machine learning models using 'flattened' data, the models may perform better or worse with different lookback windows e.g. the rolling mean model could take the mean over the last 2 or last 5 periods. The lookback length can be considered an additional hyperparameter, here we test a lookback of 2 times, 4 times and 8 times the lookahead window i.e. 2 years, 4 years or 8 years for a 1-year prediction; 6 months, 12 months, 24 months for a 3-month prediction.

6.8 Historic data exposure

Some models do not use lookbacks but instead, use the entire historic dataset e.g. ARIMA. There are large shifts in annual figures around 2005, 2014, and 2017 (see Fig. 8 above), the models are also tested using data from only after 2017 dates in case it aids performance by omitting obsolete data

Model	# features	# hyperparamters	hyperparameter configuration	lookback
Exponential smoothing	1	1 (smoothing factor)	optimise on training set	forever
Croston's method	1	2 (scalarisation factor and smoothing factor)	optimise on training set	forever
ARIMA	1	3 (autoregressive, differencing, and moving average parameters)	random search	forever / limited by autogregressive parameter
LSTM	12*T	4 (hidden neurons, depth, bidirectional, epochs)	random search	limited (time series)
RF	12*T	5 (number of estimators, criterion, min. samples split, min. samples leaf, bootstrap)	random search	limited (flattened)
GBM	12*T	7 (number of estimators, criterion, loss, min. samples split, min. samples leaf, subsample, learning rate)	random search	limited (flattened)
Yearly MVUE	3	-	-	forever
Little's Law Exit Rate	2	-	-	limited

Table 2. Features used, hyperparameter choices and data lookback window

trends. Those machine learning models using lookbacks are also impacted by this historic data as it will either be included or omitted from the training set.

6.9 Model configuration details

In summary, 11 models are trained, 10 of these have configurable hyperparameters of which 10 subsets are tested across 3 different lookbacks.

Table 2 gives an overview of the algorithms used, the number of features, the possible hyperparameters and the lookback data used to fit (or train) the models. Where the number of features is 1, the only feature is the historic data for the lookahead period e.g. if lookahead is 30 days, it's the historic 30-day totals. Where the number of features is 12*T, this means the model is using the 12 features described above but for a certain lookback period, T. For non-time series models, this is flattened into a 1-dimensional vector, for time series model a 2-dimensional matrix of size 12*T is used.

Exponential smoothing: The only hyperparameter for this model is the smoothing parameter, the SimpleExpSmoothing module from the StatsModels python library [Seabold and Perktold 2010] was used, which has in-built automatic parameter optimisation on the training set.

Croston's method: Mohammadi's python implementation [Mohammadi 2020] which also optimises parameters on the training set

ARIMA: for the ARIMA model, the autoregressive (AR), differencing (I) and moving average (MA) parameters may be estimated by a number of different methods. Due to the multiple datasets that this model is trying to fit (number of months, sub-types of CVE) we adopt a random search approach allowing values between 0 and 12 for the AR and MA parameters and 0, 1 or 2 for the differencing parameters since we do not typically observe time series of degree 3 or more. ARIMA can take exogenous variables so could take additional features, however, this requires the forecast value of said exogenous variables which introduces additional instability to the model so was omitted. Sometimes the ARIMA model cannot converge with the random parameters provided in which case another set of parameters were selected until 50 models and/or all possible configurations have been attempted. This was also implemented using the StatsModels python library [Seabold and Perktold 2010].
LSTM the LSTM was implemented using Keras [Chollet et al. 2015] python library. For neural networks, the hyperparameters can have a significant impact on the success of the model and a random search was used to find good hyperparameters.
RF and GBM for the random forests and gradient boosting machines, the time series data used by the LSTM is flattened. This allows these models to have some temporal perspective but they do not know which parameters are the same from a different time period nor the order of these variables and a random search was used to find good hyperparameters. These models were implemented using the Scikit-learn python library [Pedregosa et al. 2011] and further details on the hyperparameters can be found in the Scikit-learn documentation.

7 EXPERIMENTAL RESULTS

This section reports the results of forecasting different groups of CVEs over various time spans. The 95% prediction interval is used to select the best model, calculated over the 25 steps prior to the forecast date. We also present retrocasts whereby multiple historic forecasts are plotted on a single graph against the true number of CVEs that were observed.

7.1 Historic data, Lookback and Features

Time-series forecasting models such as ARIMA have the historical lookback build into their parameters, the number of steps to look into the past and how much history to consider can be specified when building the model. For the machine learning models and for the Yearly MVUE and Little's Law Exit Rate features the amount of history considered impacts the training data for the models and the number of samples in the 'queue' respectively.
Therefore we explored using different amounts of historic data: since 2001 when NVD records begin, and since 2017, the most recent step-change to in annual CVE volume level (see Figure 8 above). The lookback determines the data history used *per prediction*. Here we tested 1 x lookahead, 2 x lookahead and 4 x lookahead i.e. 12, 24 or 48 months of history to predict 12 months ahead.
Yearly MVUE and Little's law provided good estimations, particularly of yearly CVE volumes. For this reason, the features generated for the Yearly MVUE and Little's Law Exit Rate models are also supplied to the machine learning models. Here we show how these two developed models performed against machine learning models with and without the derived features.
Table 3 shows the comparative performance on yearly predictions of CVE numbers when varying the amount of historic data used, the lookback window and the features seen by the models. The dummy model benchmark is the lowest MAE achieved by a dummy model (previous value or rolling mean) over the prediction period - this is 1618, achieved by the rolling mean using a 24-month lookback, the average number of CVEs across the retrocast (Jan 2019 to Dec 2020) is 17,271. Only two models

Historic data since	Lookback window (months)	Features	Lookahead (months)	Algorithm	MAE
-	-	count	12	previous	1809
-	12	count	12	rolling	1936
-	24	count	12	rolling	1847
-	48	count	12	rolling	2734
2001	-	count	12	Exponential	1809
2017	-	count	12	Exponential	1790*
2001	-	count	12	Croston's	1812
2017	-	count	12	Croston's	1798*
2001	-	count	12	ARIMA	1772*
2017	-	count	12	ARIMA	5002
2001	12	count + MVUE + Little's Law	12	LSTM	1402*
2001	24	count + MVUE + Little's Law	12	LSTM	1430*
2001	48	count + MVUE + Little's Law	12	LSTM	1515*
2017	12	count + MVUE + Little's Law	12	LSTM	5084
2017	24	count	12	LSTM	5913
2017	24	count + MVUE + Little's Law	12	LSTM	4518
2017	48	count + MVUE + Little's Law	12	LSTM	5374
2001	12	count + MVUE + Little's Law	12	RF	4136
2001	24	count + MVUE + Little's Law	12	RF	5845
2001	48	count + MVUE + Little's Law	12	RF	4667
2001	24	count + MVUE + Little's Law	12	RF	5845
2017	12	count + MVUE + Little's Law	12	RF	2377
2017	24	count	12	RF	1792*
2017	24	count + MVUE + Little's Law	12	RF	2313
2017	48	count + MVUE + Little's Law	12	RF	2486
2001	12	count + MVUE + Little's Law	12	GBM	2850
2001	24	count + MVUE + Little's Law	12	GBM	3352
2001	48	count + MVUE + Little's Law	12	GBM	3104
2001	24	count + MVUE + Little's Law	12	GBM	3325
2017	12	count + MVUE + Little's Law	12	GBM	1985
2017	24	count	12	GBM	1736*
2017	24	count + MVUE + Little's Law	12	GBM	1651*
2017	48	count + MVUE + Little's Law	12	GBM	1959
-	-	MVUE	12	Yearly MVUE	1490*
-	-	Little's Law	12	Little's Law Exit Rate	**1267***

Table 3. 12 month predictions for Jan 2019 to Dec 2020 mean absolute error (MAE) different cutoff dates, lookback periods and feature sets. **best dummy model MAE, *models which improve on this benchmark.

beat the dummy model: LSTM using all historic data, all features and a 24-month lookback window (MAE=1430) and the Yearly MVUE model (MAE=1286).

Interestingly, the RF and GBM, which provide competitive performance with the dummy model do not perform as well with more historic data, instead, they are more accurate with data since 2017, the opposite is true for the LSTM. This may be due to the fact that the LSTM can process time-series data whereas the other two cannot.
The Little's Law model is the best performing model followed by the LSTM then the Yearly MVUE model, but the Little's Law model and Yearly MVUE cannot be used for subsets of CVEs since we do not know the details of these CVEs in advance therefore cannot determine which metrics to extract for subsets of forecasted general CVEs. This is the reason we integrate the features from these models to machine learning models, which in some cases are improved by the inclusion of these features (see LSTM MAE above).

7.2 Model selection methodology

Table 3 reports the accuracy metrics on the retrocast test set and would enable us to pick the best model with the benefit of hindsight. Looking at predictions for individual points in time, however, it becomes clear that the 'best model' would change if the retrocast window changed. Since we do not know which model will be best for a given prediction, can we devise a methodology in order to automatically select this model? The dual benefit of this approach is that if it works for different time periods, it could also work for different data feeds such as other lookaheads or subgroups of CVEs e.g. individual products.
In order to choose a model, each prediction uses past performance on a held-out validation set to select the algorithm with the highest performance, we refer to this henceforth as the **combined model**. The training, validation and test sets are first split strictly by date such that the validation set data occurs after the training set data. If the validation set were randomly extracted from dates within the training set (even if excluded from the training set) this could give a less-accurate representation of model performance on the test set due to information leakage. The validation set comprised the window up to most recent 6th, 5th, 4th, 3rd, and 2nd date for which the true value is available, whereas the test set comprises the window leading to the most recent date (the test date). For each new prediction the training, validation and test data shifts, with new models trained (or calculated in the case of the statistical models).
The performance is determined by a metric; a patch manager could configure the performance metric on this validation set to use mean absolute error, prediction intervals or some other metric. Here we use the number of times that the model gives a 'good' prediction, this is so that one terrible prediction on the validation set does not rule the model out. 'good' here means within 10% of the validation predictions. If there is a tie, this is broken using the 95% confidence prediction interval.
Figure 9 illustrates the model selection process. The top graph shows the competing models predictions on the 25 previous steps; only 5 of the hyperparameter random searches are shown here for each algorithm to make the graph more readable. The middle graph shows the best-performing model for each algorithm i.e. each algorithm which required hyperparameter searching is represented by the one with the lowest 95% prediction interval. Finally, the bottom graph shows the historic predictions intervals (70%, 80% and 95%) for the best overall model and the future prediction relative to the true number observed; since this was in the past we can plot the true value.
The model selected for this example (to predict the total number of CVEs that will appear between 01 January 2018 and 01 January 2019) is an ARIMA model which predicts 15,809 and the actual number observed was 16,347.
The model selection methodology is outlined graphically in Figure 9.

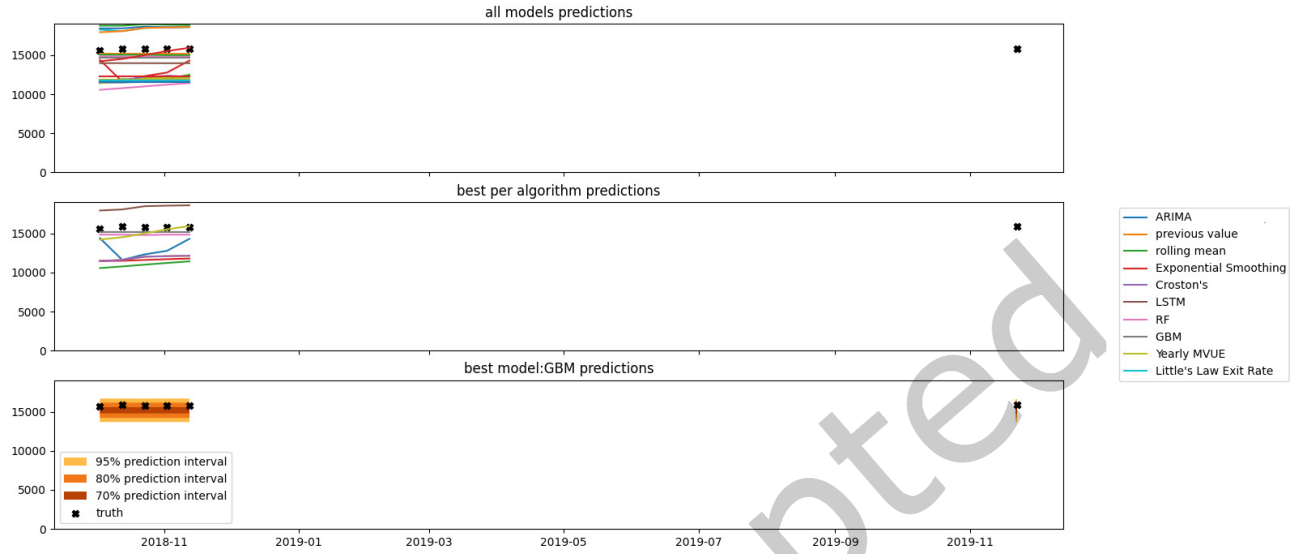


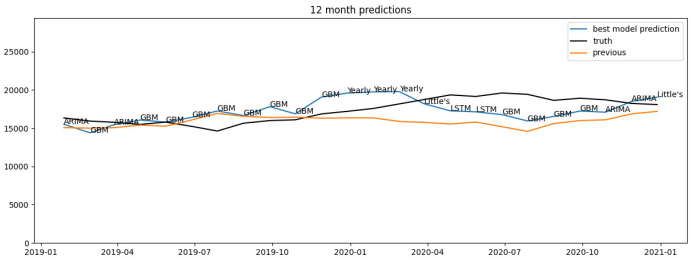
Fig. 9. Model selection process to predict the total number of CVEs that will appear between 01 January 2018 and 01 January 2019. Black crosses indicate the true value observed with a validation set (most recent 5 predictions) on the left hand side and the future prediction on the right hand side (12 months from January 2018 i.e. January 2019). The top graph shows all predictions, the second graph shows the best of all models, bottom graph shows best model predictions.

7.3 Predicting different time periods

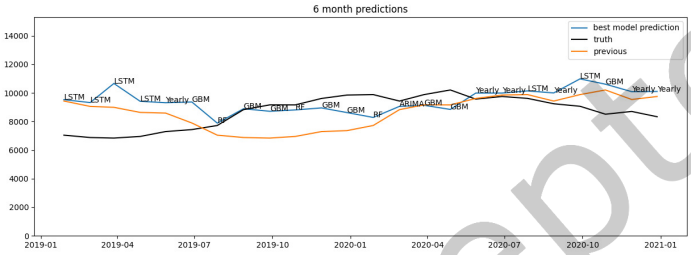
Previous work has tended to look at predictions several months ahead. Since budgets and resources are often allocated annually rather than quarterly we have also looked at longer-term predictions up to 12 months in advance.

The following table reports the predicted and actual values for each month in 2020 predicting monthly, quarterly, 6-monthly and yearly number of CVEs, i.e. the prediction for 01 January 2020 was made from the data available on 01 January 2019. Only the best algorithm (known with the benefit of hindsight) is reported, together with the previous value, rolling mean and combined model approach. The combined model looks at model scores within 10% of the true value. If there is a tie, a prediction interval at the 95% confidence level is used as a tie-breaker. Although we can look at the best model across the retrocast in the previous section - how could we choose this model over the others? Yearly MVUE scores best at predicting annual across the testing period but it does not perform well in 2018. The combined model outperforms the dummy models (previous value and rolling mean) for 12 and 6 month predictions but not for 1 or 3-month predictions when we consider the MAE score. The following plots show the combined model vs. the dummy model vs. the true observed values for the retro cast period for 12, 6, 3 and 1 month predictions:

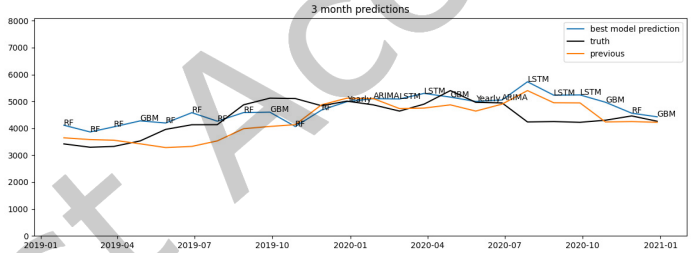
Figure 10 displays retrocasts for 1, 3, 6 and 12-month predictions. Retrocasts are multiple historic predictions made within a time period and shows the true value against these predicted values. From these graphs, we can look at past performance over time for the forecast models selected by iteratively



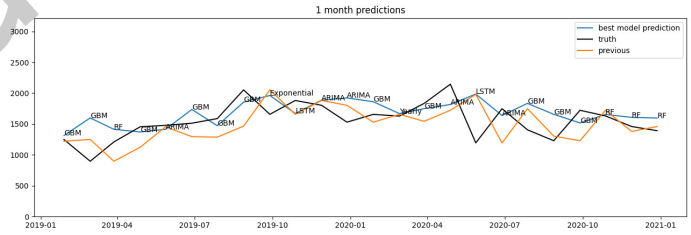
(a) 12 month retrocast



(b) 6 month retrocast



(c) 3 month retrocast



(d) 1 month retrocast

Fig. 10. Retrocasts for 1, 3, 6 and 12 month

Historic data since	Lookback window (months)	Features	Lookahead (months)	Algorithm	MAE
-	-	count	12	previous	1809
-	24	count	12	rolling	1847
2001	-	Little's Law	12	Little's Law Exit Rate	**1267***
-	-	-	12	Combined model	1572*
-	-	count	6	previous	1447
-	12	count	6	rolling	1466
2001	12	count + MVUE + Little's Law	6	GBM	**854***
-	-	-	6	combined model	1303*
-	-	count	3	previous	440
-	12	count	3	rolling	575
2001	2	count	3	Exponential Smoothing	**438***
-	-	-	3	combined model	478
2017	1	count	1	previous	307
2017	1	count	1	rolling	**210**
2017	4	count + MVUE + Little's Law	1	RF	231
-	-	-	1	combined model	235

Table 4. 12, 6, 3 and 1 month mean absolute error (MAE) for 24 predictions between January 2019 to January 2021 comparing different cutoff dates, lookback periods and feature sets. **=best dummy model MAE, *=models which improve on this benchmark

pretending we are somewhere in the past and trying to predict the number of CVEs. This retrocast is useful to see if this approach tends to work over time and to generate accuracy metrics with greater statistical significance.

The figures show that the combined model tends to predict above the previous value score. When looking at the shorter-term predictions in particular, when there is a sudden increase in the number of CVEs, the combined model tends to be closer to the true value than the dummy model - for example in August 2019 (1-month and 3-month), and July 2020 (1 month). We speculate that this is probably preferable to patch managers who would prefer to be over-resourced than under-resourced.

The combined model requires training tens of models (here we trained 269 to check all lookbacks and feature sets) per lookahead window, this is not computationally cheap and requires some cost to implement. A single model still produces confidence intervals which could be used to gauge the accuracy of a prediction. If a practitioner were only to implement one model to predict 12 months in advance we would suggest using Little's Law given it's low error rate. However, if multiple time windows and sub-categories of vulnerabilities will also be predicted and only one type of algorithm used, the Gradient Boosted Machine gave the best overall performance across all data types.

7.4 Product-specific predictions

Though the total number of CVEs can be used as a weather vane for global vulnerability trends, but typically individuals will be responsible for a (set of) products. Other subsets of CVEs that may be interesting are the number of CVEs above a certain CVSS score (since patching goals are often tied to these scores) or perhaps to compare two vendors or products when considering which to take on. By definition, these subsets of CVEs have fewer data points than the global number of CVEs. Moreover, some products have never had a CVE then suddenly 25 appear because a researcher has turned their attention to it. This data is not captured by the NVD database, perhaps it could be predicted using exogenous data streams e.g. market share [O'Donnell 2008] but this is outside the scope of this paper. There are therefore some products that we exclude outright as being forecastable: those which have never had a CVE, those with very little data (less than 5 CVEs ever) and those with very volatile and irregular histories are more difficult to predict too.

Our goal is to create a flexible model selection framework, therefore we do not make any changes to the code or methodology for product selection. The only difference is that now the features are the global variables + count features for the specific product. The MVUE and Little's law features do not help for these specific predictions.

Table 5 reports the mean absolute error for the combined model against the two basic models: previous value and rolling mean across 24 predictions between January 2019 and January 2021 for 12, 6, 3 and 1-month predictions. The products chosen for comparison are three major operating systems and the most widely used PDF reader globally. It is interesting to note that at the one-month level, all products see a large variance in the number of CVEs published with the minimum being at least 7 times the maximum in all cases.

Despite the diversity in volume and variance of volume, the combined model gives the lowest MAE for most products and lookaheads. One notable case in which it performs poorly is for the Linux Kernel 12 month prediction for which the MAE is 40% higher than the MAE just using the previous year's value. The model performs surprisingly well for some products such as Adobe Acrobat Reader 1 month predictions, which range between 0 and 175. Though the model copes well with the dramatic rise and fall in CVE volume, it cannot predict a sudden increase in CVEs when there has not been sufficient historic data, see Figure 11.

As a quick first-pass to distinguish CVE subsets that may be predictable using the combined model, we used the Augmented Dickey-Fuller test [Dickey 1997], which tests the null hypothesis that a time series has differential stationarity. This revealed that around 50 products could benefit from uncertainty reduction around the number of expected CVEs using the combined model. Future work could refine and improve on this method further.

7.5 CVSS scores and type predictions

It may also be of interest to further drill into the spread of CVSS scores. For all CVEs, we found that the number of CVEs with a CVSS score above e.g. 9 or 7 was incredibly stable over time, thus negating the need for forecasting, though the combined model can still be used here and should adapt were the volume to change dramatically. The total number of CVSS scores above a certain threshold may be of interest to insurance and risk analysts, but for individual product owners it may be more valuable to predict the CVSS above a certain level for a particular product.

In taking a smaller subset, it can be even more challenging as the fewer samples, the higher the variance. Here we have used Linux vulnerabilities due to the model's poorer performance above and have tried to estimate the number of vulnerabilities with a CVSS score above 7. This methodology

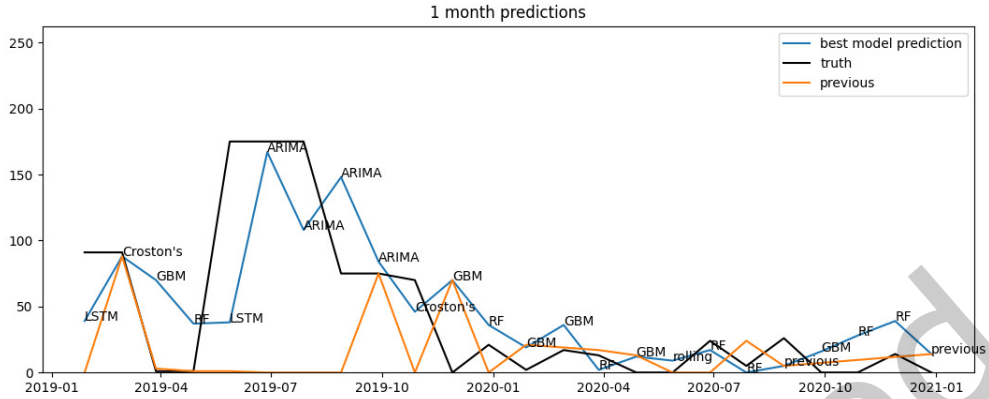
Product	Lookahead	Combined model MAE	Previous value MAE	Rolling mean MAE	Min Observed	Max Observed
Adobe Acrobat Reader	12	**163**	175	172	101	519
Adobe Acrobat Reader	6	91	**75**	80	45	320
Adobe Acrobat Reader	3	63	49	**48**	14	250
Adobe Acrobat Reader	1	**27**	42	35	0	175
Microsoft Windows 10	12	**156**	173	174	492	570
Microsoft Windows 10	6	**84**	108	95	232	570
Microsoft Windows 10	3	**65**	**65**	71	102	305
Microsoft Windows 10	1	31	**19**	21	21	200
Mac OSX	12	**156**	177	161	470	865
Mac OSX	6	**83**	116	100	143	457
Mac OSX	3	73	84	**72**	46	321
Mac OSX	1	53	79	**52**	4	200
Linux Kernel	12	191	**129**	138	200	389
Linux Kernel	6	**68**	83	87	84	266
Linux Kernel	3	38	**29**	36	39	155
Linux Kernel	1	**11**	13	12	6	87

Table 5. Individual product predictions 1, 3, 6 and 12 month. MAE of combined model, previous value and rolling mean are compared for 24 predictions between January 2019 to January 2021. Minimum and maximum number of CVEs observed during a given period are reported as an indication of variance.

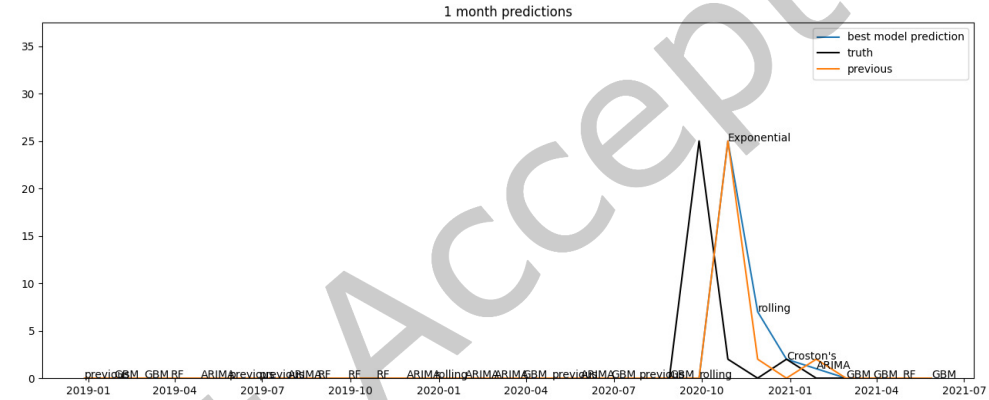
is the same as for product forecasting, the data used is simply selected using a different filtering method.
Table **??** shows that in this case, the combined model actually outperforms all others, despite the wide range of values seen (see Min Observered and Max Observed columns).

8 LIMITATIONS

As mentioned earlier the choice of predictor in the combined model is based on the validation set. There may in fact be exogenous factors outside the NVD data which would explain significant changes in the vulnerability volume such as structural changes in reporting entities, vendors and/or the market for vulnerability hunting, which in turn may be due to the increased market share held by a product [O'Donnell 2008].



(a) Acrobat



(b) Tensorflow

Fig. 11. Predicting 1 month of Adobe Acrobat PDF Reader CVEs vs. Tensorflow deep learning library CVEs

Lookahead	Combined Model MAE	Previous value MAE	Rolling mean MAE	Min Observed	Max Observed
12	**34**	61	54	89	191
6	**22**	29	33	41	113
3	**16**	22	25	18	66
1	**4**	7	9	0	30

Table 6. Linux Kernel CVEs with a CVSS above 7 volume prediction

The generalisation of this model could be tested with different vulnerability feeds and/or databases, we hope this paper has given inspiration for future work to do so.

9 CONCLUSION AND FUTURE WORK

Initial correlation analysis of the vulnerabilities indicated that there might be a predictive process memory limit at the 3-month mark. The MVUE and Little's Law features both gain value from data far further back in the process than this three month lookback. This is intuitive as Little's Law is an average of arrival and exit rates, and as such even benefits from outliers that take years to process. Equally, a serial number from 6 months ago could still be the highest published. This extension of the memory through statistical features enables accurate predictions as far as 12 months ahead. In other words they have more predictive power than the correlation of previous values.

By adding these diligently crafted statistical features to machine learning models, we have achieved better accuracy in forecasting the volume of CVEs than previous work. By forecasting the total volume first, you can use this number (or prediction interval) as an input to vendor/product CVE volume forecasts (composition models), and traditional time series forecasting approaches will potentially yield strong results for as many as 50 products at the time of writing. We have also produced useful visualisation techniques to verify the accuracy of predictions over time, both in retrocasting to build the confidence of the forecasts, and in communicating the forecasts to end-users themselves.

It is also worth making a relevant policy statement while concluding this paper. This research must be done simply because NVD does not publish a forecast themselves. Nor does MITRE, or Microsoft, or Linux Foundation, or Mozilla, or Google. Perhaps there are reasons these organisations choose not to publish their upcoming CVE volumes. Though if they did it would significantly ease the efforts of patch management and cyber risk quantification, but to our knowledge any conscious decision **not** to publish such things is undocumented. Disclosure debate would be improved by such discussions, regardless of the outcome.

Future work could look to build on this approach by validating it with other vulnerability streams, such as the China NVD [CERT 2021] instead of the US NVD, and there is no reason they could not publish their own forecast as well.

We have built on the foundations of others who researched in this field before us, and we thank them for documenting their testing methodologies. We respected those works and tested our models with their standards at a minimum. However, we have also extended their time windows and accuracy significantly, and we believe there is much better work to come in the future.

We predict the future of vulnerability forecasting research will continue to improve look ahead windows, uncertainty reduction, volume forecasts, feature forecasting, and accuracy. As it does so, it will become an integral part of vulnerability-centric cyber risk quantification, and allow us to measure our predictive power. Let the reactionary past of vulnerability management fade, the future of cyber risk belongs to the predictors and forecasters.

REFERENCES

Aleksandar Aleksić, Miladin Stefanović, Danijela Tadić, and Slavko Arsovski. 2014. A fuzzy model for assessment of organization vulnerability. *Measurement* 51 (2014), 214 – 223. https://doi.org/10.1016/j.measurement.2014.02.003

Omar H Alhazmi and Yashwant K Malaiya. 2006. Prediction capabilities of vulnerability discovery models. In *RAMS'06. Annual Reliability and Maintainability Symposium, 2006.* IEEE, 86–91.

Luca Allodi. 2015. The heavy tails of vulnerability exploitation. In *International Symposium on Engineering Secure Software and Systems*. Springer, 133–148.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research* 13, 2 (2012).

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

Robert M Brady, Ross J Anderson, and Robin C Ball. 1999. *Murphy's law, the fitness of evolving species, and the limits of software reliability*. Technical Report. University of Cambridge, Computer Laboratory.

Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

China CERT. 2021. China National Vunlenrability Database. https://www.cnvd.org.cn/.

Aristeidis Chatzipoulidis, Dimitrios Michalopoulos, and Ioannis Mavridis. 2015. Information infrastructure risk prediction through platform vulnerability analysis. *Journal of Systems and Software* 106 (2015), 28–41.

Haipeng Chen, Rui Liu, Noseong Park, and VS Subrahmanian. 2019. Using twitter to predict when vulnerabilities will be exploited. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3143–3152.

Jay Chen. 2020. The State of Exploit Development: 80% of Exploits Publish Faster than CVEs. (August 2020). Online [Accessed 13th Nov. 2020] https://unit42.paloaltonetworks.com/state-of-exploit-development/.

Francois Chollet et al. 2015. *Keras*. https://github.com/fchollet/keras

John D Croston. 1972. Forecasting and stock control for intermittent demands. *Journal of the Operational Research Society* 23, 3 (1972), 289–303.

David A Dickey. 1997. Introduction to Statistical Time Series.

Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

Dan Geer. 2015. For good measure: The undiscovered. *; login:: the magazine of USENIX & SAGE* 40, 2 (2015), 50–52.

Ant Goldbloom. 2015. What algorithms are most successful on Kaggle? Accessed online [20-04-21] https://www.kaggle.com/antgoldbloom/what-algorithms-are-most-successful-on-kaggle.

CC Heyde. 2006. Central limit theorem. *Encyclopedia of actuarial science* 1 (2006).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

Charles C Holt. 2004. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting* 20, 1 (2004), 5–10.

Eric Jardine. 2018. Mind the denominator: towards a more effective measurement system for cybersecurity. *Journal of Cyber Policy* 3, 1 (2018), 116–139. https://doi.org/10.1080/23738871.2018.1472288 arXiv:https://doi.org/10.1080/23738871.2018.1472288

Roger W Johnson. 1994. Estimating the size of a population. *Teaching Statistics* 16, 2 (1994), 50–52.

David Last. 2016. Forecasting zero-day vulnerabilities. In *Proceedings of the 11th Annual Cyber and Information Security Research Conference*. 1–4.

John DC Little. 2011. OR FORUM—Little's Law as viewed on its 50th anniversary. *Operations research* 59, 3 (2011), 536–549.

John DC Little and Stephen C Graves. 2008. Little's law. In *Building intuition*. Springer, 81–100.

Lucas Miranda, Daniel Vieira, Mateus Nogueira, Leonardo Ventura, Miguel Bicudo, Matheus Martins, Lucas Senos, Leandro P de Aguiar, Enrico Lovat, and Daniel Menasche. 2021. Measuring and modeling software vulnerability security advisory platforms. In *Risks and Security of Internet and Systems: 15th International Conference, CRiSIS 2020, Paris, France, November 4–6, 2020, Revised Selected Papers 15*. Springer International Publishing, 31–48.

MITRE. 2020. CVE - FAQs. Online [Accessed: 30 July 2020 https://cve.mitre.org/about/faqs.html#year_portion_of_cve_id].

Hamid Mohammadi. 2020. *Croston Python Library*. https://github.com/HamidM6/croston

A. J. O'Donnell. 2008. When Malware Attacks (Anything but Windows). *IEEE Security Privacy* 6, 3 (2008), 68–70.

Andy Ozment and Stuart E Schechter. 2006. Milk or wine: does software security improve with age?. In *USENIX Security Symposium*, Vol. 6.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

Håkan Petersson, Thomas Thelin, Per Runeson, and Claes Wohlin. 2004. Capture–recapture in software inspections after 10 years research—-theory, evaluation and application. *Journal of Systems and Software* 72, 2 (2004), 249–264.

Nawa Raj Pokhrel, Hansapani Rodrigo, Chris P Tsokos, et al. 2017. Cybersecurity: time series predictive modeling of vulnerabilities of desktop operating system using linear and non-linear approach. *Journal of Information Security* 8, 04 (2017), 362.

Eric Rescorla. 2005. Is finding security holes a good idea? *IEEE Security & Privacy* 3, 1 (2005), 14–19.

Richard Ruggles and Henry Brodie. 1947. An empirical approach to economic intelligence in World War II. *J. Amer. Statist. Assoc.* 42, 237 (1947), 72–91.

Felix Schuckert. 2016. PT: Generating security vulnerabilities in source code. In *Sicherheit 2016 - Sicherheit, Schutz und Zuverlässigkeit*, Michael Meier, Delphine Reinhardt, and Steffen Wendzel (Eds.). Gesellschaft für Informatik e.V., Bonn, 177–182.

Model	Hyperparameter	Possible Values ([min, max])
ARIMA	autoregressive	[0, 10]
ARIMA	differencing	[0, 2]
ARIMA	moving average	[0, 10]
LSTM	hidden neurons	[1, 1000]
LSTM	depth	[1, 3]
LSTM	bidirectional	True, False
LSTM	sequence length	[2, time series length]
RF	number of estimators	[1, 1000]
RF	criterion	mean squared error, mean absolute error
RF	minimum samples split	0.1%, 1%, 10%, 20%, 30%, 40%, 50%, 1 sample
RF	minimum samples leaf	0.1%, 1%, 10%, 20%, 30%, 40%, 50%, 1 sample
RF	bootstrap	True, False
GBM	number of estimators	[1, 1000]
GBM	criterion	mean squared error, mean absolute error, Friedman mean squared error
GBM	loss	least squares, least absolute deviation, Huber loss, quantile loss
GBM	minimum samples split	[0.1%, 1%, 10%, 20%, 30%, 40%, 50%, 1 sample]
GBM	minimum samples leaf	[0.1%, 1%, 10%, 20%, 30%, 40%, 50%, 1 sample]
GBM	subsample	50%, 60%, 70%, 80%, 90%, 100%
GBM	learning rate	[0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5]

Table 7. Hyperparameter values used for random search, see the following python libraries for details: ARIMA: the StatsModels [Seabold and Perktold 2010], LSTM: Keras [Chollet et al. 2015], RF and GBM: Scikit-learn [Pedregosa et al. 2011]

Hanna Scott and Claes Wohlin. 2008. Capture-recapture in software unit testing: a case study. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*. 32–40.

Skipper Seabold and Josef Perktold. 2010. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*.

Octavian Suciu, Connor Nelson, Zhuoer Lyu, Tiffany Bao, and Tudor Dumitras. 2021. Expected Exploitability: Predicting the Development of Functional Vulnerability Exploits. *arXiv e-prints* (2021), arXiv–2102.

Scott A Vander Wiel and Lawrence G. Votta. 1993. Assessing software designs using capture-recapture methods. *IEEE Transactions on Software Engineering* 19, 11 (1993), 1045–1054.

HS Venter and Jan HP Eloff. 2004. Vulnerability forecasting—a conceptual model. *Computers & Security* 23, 6 (2004), 489–497.

J. Wang, Z. He, J. Oh, and S. Park. 2008. Multi-Response Robust Optimization Using Desirability Function. In *2008 IEEE Symposium on Advanced Management of Information for Globalized Enterprises (AMIGE)*. 1–3.

Emrah Yasasin, Julian Prester, Gerit Wagner, and Guido Schryen. 2020. Forecasting IT security vulnerabilities–An empirical analysis. *Computers & Security* 88 (2020), 101610.