CHALMERS
UNIVERSITY OF TECHNOLOGY

Data Modelling for Predicting Exploits

Citation for the original published paper (version of record):

Reinthal, A., Filippakis, E., Almgren, M. (2018)
Data Modelling for Predicting Exploits
Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and

(article starts on next page)

Data Modelling for Predicting Exploits[⋆]

Alexander Reinthal[0000−0002−6071−8853], Eleftherios Lef Filippakis[0000−0002−2770−3326], and Magnus Almgren[0000−0002−3383−9617]✉

Chalmers University of Technology, Gothenburg, Sweden
`reinthal@student.chalmers.se`, `lefphilip@live.com`,
`magnus.almgren@chalmers.se`

Abstract. Modern society is becoming increasingly reliant on secure computer systems. Predicting which vulnerabilities are more likely to be exploited by malicious actors is therefore an important task to help prevent cyber attacks. Researchers have tried making such predictions using machine learning. However, recent research has shown that the evaluation of such models require special sampling of training and test sets, and that previous models would have had limited utility in real world settings. This study further develops the results of recent research through the use of their sampling technique for evaluation in combination with a novel data model. Moreover, contrary to recent research, we find that using open web data can help in making better predictions about exploits, and that zero-day exploits are detrimental to the predictive powers of the model. Finally, we discovered that the initial days of vulnerability information is sufficient to make the best possible model. Given our findings, we suggest that more research should be devoted to develop refined techniques for building predictive models for exploits. Gaining more knowledge in this domain would not only help preventing cyber attacks but could yield fruitful insights in the nature of exploit development.

Keywords: exploits · machine learning · concept drift · vulnerability management.

1 Introduction

Every year, thousands of vulnerabilities are published. Most of these vulnerabilities are benign and never exploited. As an example, in 2017, 12 561 vulnerabilities were published with Common Vulnerabilities and Exposures (CVE) identifiers, the industry standard of identifying vulnerabilities. Only 11% of these vulnerabilities had proof-of-concept exploits attached to them, and only a fraction of these exploits would ever actively be used in the wild. Since patching a vulnerability is time consuming and costly, security teams have to triage the vulnerabilities found in their system and patch the most critical vulnerabilities first. The CVSS

⋆ Preprint of https://doi.org/10.1007/978-3-030-03638-6_21 score has been a common tool for assessing the severity of a vulnerability. However, previous studies have shown that CVSS scores are not ideal indicators for which vulnerabilities are in need of patching [1].

Machine learning models could possibly be a good tool to use as a proxy of likelihood of exploitation. The topic has been studied in academia for many years but only recently gained traction in industry [11]. Having a functional machine learning model would not only alleviate the manual labour involved in sorting through the large volume of vulnerability information, but could potentially provide valuable insight in the nature of exploitation.

Although many researchers have built machine learning models for exploit prediction, recent research by Bullough et al. [3] has shown that the promising results obtained in previous research [2, 12, 5] were most likely an artifact of unrealistic treatment of data.

The goal of this study was to develop an understanding for how machine learning models are affected by different assumptions about the data. Using a novel method to aggregate data that accurately reflects knowledge about vulnerabilities prior to their exploitation, we make the following contributions:

- We confirm Bullough et al.'s result that the data undergoes concept drift for samples collected during the time period 2015 to early 2018.
- We find that using online web chatter about vulnerabilities has a positive impact on exploit prediction, and that zero-day vulnerabilities are detrimental to the model's performance.
- We discover that the early information about vulnerabilities is sufficient to make the best possible prediction.

In Section 2, we outline previous work on predicting exploits using machine learning. In Section 3, we describe our approach and give an overview of how we treat the data. In Section 4, we describe our four experiments and present their results. In Section 5, we briefly discuss our findings. Finally, in Section 6, we give some concluding remarks.

2 Challenges and Related Work

This section will cover some pitfalls with making a predictive model for exploits as well how researchers have tried to handle those problems. We limit this section to the main related work [3, 5, 12] and discuss challenges that they faced in detail.

2.1 Realistic Data Aggregation

An important task when predicting future exploitation events is to assemble data that excludes information after the vulnerability has been exploited. Since most vulnerability databases update their data continuously, previous research [3, 5] has tried to redact the data to, for example, exclude references to exploit databases. However, aggregating the correct data without knowing when something has been changed or what has been changed quickly becomes infeasable.

This practice comes with no guarantees that the vulnerability entries contain the same information as it did before it was published in an exploit database.

To combat this challenge, in this work we use NVD's change log to backtrack changes to approximately 8 days after publication in NVD. This way, our data is representative of the knowledge of vulnerabilities before they became exploited. For more details on how this was done, see Section 3.1.

2.2 Temporal Intermixing and Realistic Evaluation

Previous research has indicated that designing classifiers for exploit prediction requires careful sampling of the training and test sets that respects time [3]. This result was attributed to concept drift, a phenomenon which is often observed in predictive models where the relationship between predictor variables and labels, i.e. concepts, changes over time [13, 8]. Most of previous researchers has assumed that sampling of training and test sets do not have to respect time. Some have assumed this explicitly [12], some have assumed this implicitly [2, 5].

This study will try and recreate previous research [3] by conducting similar experiments. These experiments are outlined in Section 3.

3 Approach

To better understand what data is needed to make good predictions on exploits, we have designed two methods of aggregating data. These methods are outlined in Section 3.1. In Section 3.2, we give the reader an overview of our data and our feature engineering. Section 3.3 provides information on how we split the training and test data and how we labeled our data. Section 3.4 covers how we configured and tuned model parameters.

3.1 The Models

In the following section we describe the two models we designed that we call the *naive* and the *realistic* model.

The *naive* model was constructed from the data provided by NVD "as-is", which is how the data has been used in previous research [3, 5, 12]. This model collects NVD data and *web chatter* for an extended amount of time. The main issue with this model is that it keeps collecting data even after the exploitation event. Collecting information after that event will "leak" information about the future to the past. As such, the model cannot realistically be used for making predictions about the future.

As an alternative, we introduce our *realistic* model that limits aggregation by a cut-off time equal to the median number of days from NVD publication to Exploit DB publication. In the following sections, we use $t_{i,s}$ to denote the starting time of data aggregation of vulnerability i and $t_{i,f}$ to denote the finishing time of data aggregation of vulnerability i.

Naive Model To compare the realistic model to related work on predicting exploits, we designed the *naive* model which aggregates data as long as a vulnerability is active in the NVD database.

The *naive* model starts aggregating data from the day of the initial NVD publication for each vulnerability i. As we have stated before, we call this day $t_{i,s}$. The final day for aggregating data, $t_{i,f}$, is the date of last modification of vulnerability i. Notice that this model will keep aggregating data even if the vulnerability has been exploited. A schematic of this data aggregation can be seen in Figure 1.

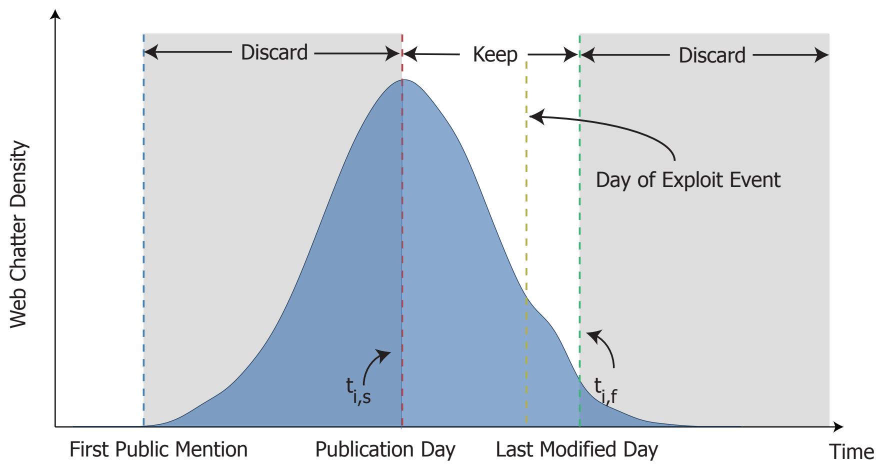


Fig. 1. This illustration shows during which period we aggregate data for vulnerabilities in the *naive* model.

Realistic Model In the realistic model, illustrated in Figure 2, we are more careful with how we aggregate data. For each vulnerability i, we set the first day of aggregation, $t_{i,s}$ (seen as the blue dashed line in Figure 2), to be the day of first recorded web mention.

If a vulnerability has been exploited the last day of aggregation $t_{i,f}$ is set to the day of exploitation (dark green dashed line in Figure 2). However, the majority of vulnerabilities do not get exploited. In that case, $t_{i,f}$ is set to $t_{i,s}+n'$ where n' is equal to the median number of days to exploitation from the first recorded web mention. The statistic n' is calculated by taking the median number of days to exploitation for all exploited vulnerabilities in the data. The median statistic is used as it is less influenced by outliers. In the exceptional case where NVD has not had time to publish the vulnerability n' days after the first recorded web mention, $t_{i,f}$ is set to the day of NVD publication.

Notice that the cut-off n' is in general the same across all vulnerabilities. This gives each vulnerability approximately the same amount of time to collect data, which is important to avoid creating biases in the data. For example, features such as number of days since published, will in the *naive* model be large for vulnerabilities that stay relevant for longer periods of time which is usually the case for exploited vulnerabilities.

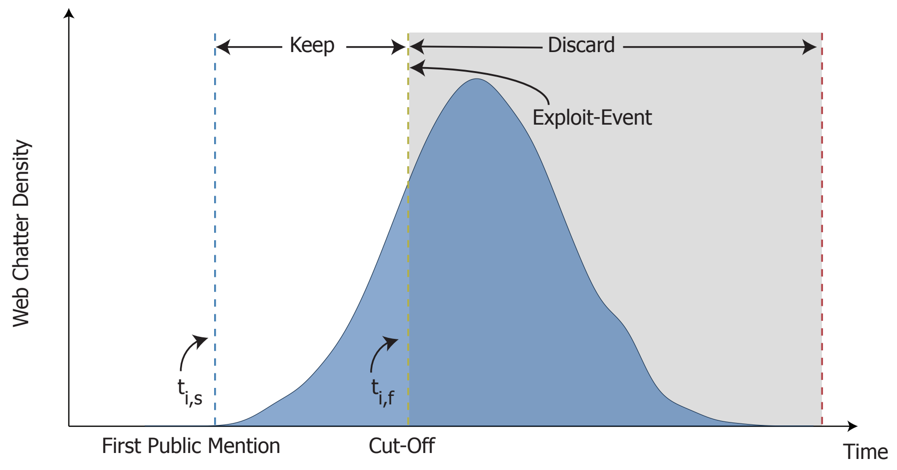


Fig. 2. This illustration shows how we set the cut-off day for collecting data for individual vulnerabilities in the *realistic* model.

3.2 Data Collections and Feature Engineering

To construct a machine learning algorithm, data was aggregated from three sources and then combined together using the Common Vulnerabilities and Exposures Identifier. In the following sections, we will describe each of the data sources along with their contributions of features for the machine learning algorithm. A complete summary of the features that were used in the algorithm can be seen in Table 1.

Vulnerabilities Each entry in the first collection is a vulnerability that has been assigned a unique CVE-ID and has been published in the NVD database [9]. We took the vulnerability descriptions and converted them into a term frequency inverse document frequency (TF-IDF) matrix. We also used the CVSS data by converting categorical features using a one-hot encoding and rescaling numerical features to standard normal distributions. Moreover, we constructed a set of features for the most common sources of references. These features encode the spread of references across web sites that report and document vulnerabilities. For the naive model, we also calculate the number of active days of the vulnerability.

Web Chatter The second collection, called the web chatter collection, consists of fragments of text that has been published online with at least one mention of a vulnerability present in our *vulnerabilities* collection. The data in this collection was provided by Recorded Future's cyber threat intelligence platform [10], which actively scrapes many relevant sources of vulnerability information. Twitter, GitHub and CERT announcements make up approximately 50% of the data in this collection. The other 50% was collected from roughly 9 500 miscellaneous sources such as paste bins, security forums and other cyber security information platforms. The text fragments where later aggregated on CVE-ID and converted into a TF-IDF matrix. Moreover, we also constructed a set of features that encode the spread between languages and sources of web chatter.

Exploits The data contained in the third collection are exploits published on Exploit Database's (Exploit DB) website [6]. Each entry in this collection is an exploit that mentions one or more vulnerabilities found in our *vulnerabilities* collection. Other exploit sources such as exploit kits, studied by for example Allodi et al. [1], were initially considered but ruled out as their acquisition is cumbersome and resulting models would not have been comparable with previous research.

Table 1. The features in our data set for both the naive and realistic model during a run in February 2018.

Source	Data		
	Category	Raw Data	Modelled as
NVD	References	List of URLs	Fraction of Common Sources
	Nr. References	Num.	Scaled to $N(0,1)$
	CVSS Data	Cat.	One-hot Encoding
		Num.	Scaled to $N(0,1)$
	Description	Text	TF-IDF
	CWE data	Multi. Cat.	Binary Vector
	Published Date	Date	Time difference
	Modified Date	Date	
Web Chatter	Source URLs	List of URLs	Fraction of Common Sources
	Nr. Source URLs	Num.	Scaled to $N(0,1)$
	Source Language	List of Languages	Fraction of Common Languages
	Captured Text	Text	TF-IDF
Exploit	Label	CVE-ID	$1 \Leftrightarrow$ CVE-ID $\in EDB$
	Publication Date	Date	*Not in data frame*

3.3 Training Sets, Test Sets and Labels

The training and test sets for both the naive and the realistic model have been split in such a way that the training set contains past events and the test set contains future events (relative to the training set). For example, using this model we could be training on last year's vulnerabilities to predict the next month's exploits.

To split the data in future and past events, we set a cut-off day d', which one can interpret as the "*present day*" of the model. This parameter is chosen to achieve an 80/20 split of training and test samples.

As shown in Figure 3, given a cut-off day d', a start of a vulnerability $t_{i,s}$ and an end of a vulnerability $t_{i,f}$, there are three cases which determine if a vulnerability ends up in the training or the test set. Any vulnerability that is a past event relative to the cut off day d' is put in the training set (case 1). Any vulnerability that is an ongoing vulnerability relative to the cut off day d' is pruned from the model (case 2). We do this in an effort to keep the model realistic as we do not want to train the model on an event that has not yet been concluded. Finally, the future events relative to the cut off d' are put in the test set (case 3). These future events are then used to evaluate the performance of our model.

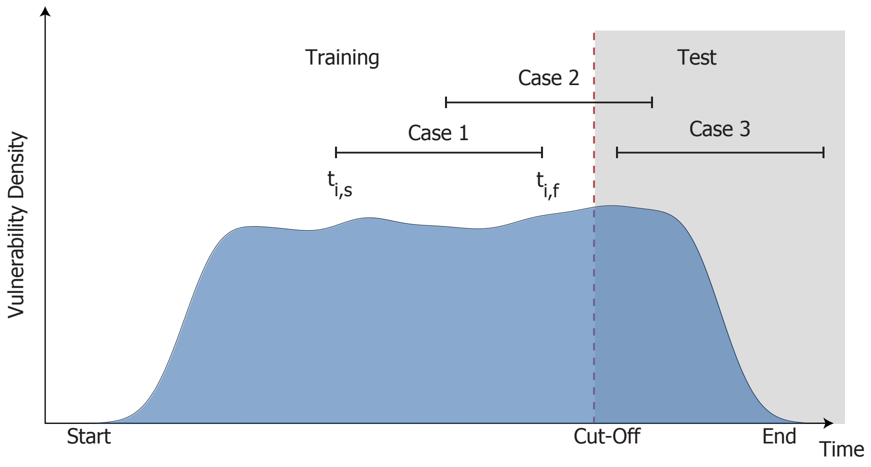


Fig. 3. Figure shows how training and test sets have been split for both the naive and the realistic model.

Labels for supervised learning are created in the following way:

$$y_i = \begin{cases} 1 \Leftrightarrow i \in \text{EDB} \\ 0 \qquad \text{otherwise} \end{cases}$$

where i is a vulnerability identified by its CVE-ID. This means that any vulnerability found in the exploit collection is labeled as exploited.

Table 2 displays the training and test split that we used for our data along with the number of samples and the positive and negative class percentage. For this realization of our data set, we used $d' = 2017\text{-}08\text{-}08$ as our training-test split parameter.

Table 2. Information about the training and test split with d' set to 2017-08-08.

Nr. Samples	Naive	Nr. Exploited	Realistic	Nr. Exploited
Total	24944	809 (3.2%)	24323	809 (3.3%)
Training (Case 1)	19644	719 (3.6%)	19146	719 (3.8%)
Dropped (Case 2)	179	3 (1.7%)	72	3 (4.2%)
Testing (Case 3)	5121	87 (1.7%)	5105	87 (1.7%)

3.4 Supervised Learning Algorithm and Optimization

To perform supervised learning, we choose the eXtreme Gradient Boosting package (xgboost) [4] as it is generally known to produce good results on imbalanced data sets. This implementation is based on the gradient boosting algorithm designed by Friedman et al. [7] which is an ensemble model that typically uses decision trees as predictors.

The choice of hyper-parameters were the result of a grid search, optimizing for highest F_1 across a 5-fold cross-validation on the training set. The resulting parameters and ranges for the grid search can be seen in Table 3.

For the training phase of our model we optimize for maximum F_1 score using validation hold out. The validation set is a 10% stratified random subset of the training set.

Table 3. Parameters used for our eXtreme gradient boosting algorithm, and the range for hyper-parameter tuning using grid search.

	η	γ	depth
Naive	0.1	0.8	7
Realistic	0.2	0	8
Grid Search Range for Tuning	$[0.1, 1]$ by 0.1	$[0, 3]$ by 0.2	$[4, 8]$ by 1

4 Results

In the following sections, we outline our experiments and study their respective impact on a baseline model described in Section 4.1. The goal of these experiments, which have been inspired by the work of Bullough et al. [3], is to develop an understanding for how different modelling practices affect the results of our classification algorithm. We apply these experiments to both the naive model and the realistic model to see whether the outcome is due to how the data is aggregated. The results of the classifiers are presented as precision and recall plots, which have been derived by obtaining precision and recall values from sweeping through the estimated likelihood values provided by the classifier.

4.1 Experiment 0: The Baseline Models

To have something to compare our experiments against, we made baseline models that were not subjected to any of the experiments outlined in the coming sections. The baseline models use the following assumptions.

- No zero-day exploits in the data.
- The training and test set are temporally separated with $d' = 2017\text{-}08\text{-}08$ as the separating day.
- Data include web-chatter features.

This is our best effort to modelling the problem as accurately as possible without any known form of faulty assumptions or unrealistic performance boosts. The baseline models are used for comparison in each experiment, and their respective precision and recall values can be seen as the orange lines in the precision and recall plots of each experiment, which are presented in sections 4.2, 4.3 and 4.4.

In Table 4, we present the maximum F_1 score for the naive and realistic baseline models, and in Figure 4 we compare their precision and recall curves. As seen in Table 4, the naive model has a lower maximum F_1 score than the realistic model, and in Figure 4, we observe that the naive model has worse over all performance since it has less precision for matching values of recall. This result suggests that the first 8 days is enough to make predictive models for exploits, realistic or not.

Table 4. The maximum F_1 score of the *naive* and *realistic* baseline models with their corresponding precision and recall values.

	Naive	Realistic
Precision	0.525	0.578
Recall	0.333	0.458
F_1 Score	0.407	0.511

4.2 Experiment I: Including Zero-Day Exploits

To make predictions about future events, we have to exclude zero-day vulnerabilities from our data set as these are announced after the exploit event has occurred and cannot be predicted in a meaningful way.

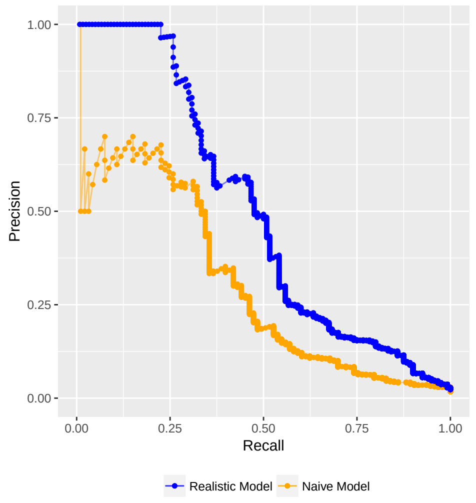


Fig. 4. Direct comparison of the classification performance for the *naive* and the *realistic* model.

Any exploit where $t_{i,e} \leq t_{i,s}$ for vulnerability i is considered a zero-day exploit, where $t_{i,e}$ is the time of exploitation. In practical terms, this means that any exploit published in EDB prior to publication in NVD is considered a zero-day exploit. In this experiment, we put those vulnerabilities back into the model to study the difference in performance. This experiment is similar to the experiment performed by Bullough et al. [3] with the caveat that their zero-day vulnerabilities were removed from their baseline model.

In Figure 5, we see the relationship between precision and recall for both our naive and realistic models when including zero-day exploits (blue lines) and our baseline models which exclude zero-day exploits (orange lines). In Table 5, we see the maximum F_1 scores achieved by our models along with their respective precision and recall values.

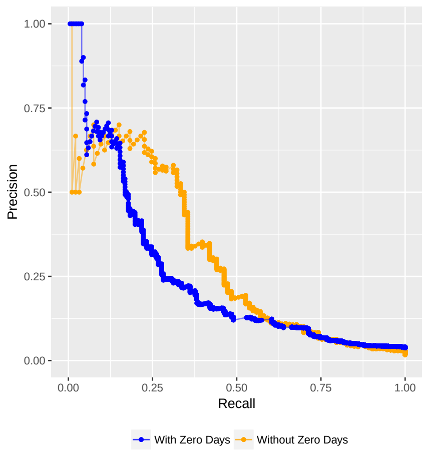
We observe that both our experimental models perform worse than the baseline models. In Table 5, we observe that the *naive* and *realistic* model have F_1 scores of 0.407 and 0.511 respectively when excluding zero-days compared to 0.284 and 0.391 when including zero-days.

The fact that our naive model perform worse when including zero-day exploits is the opposite effect of what Bullough et al. observed. Moreover, the fact both models showed worse performance indicates that the relative time frame is not likely the cause of our result being different from Bullough et al.'s observations. We think our results show the opposite due to our zero-day exploits not making up the majority of our labels. In their case, their zero-day exploits make up approximately 90%[1] of their total number of exploits. In comparison, zero-day exploits make up approximately 27% of our exploits in the *naive* model and 30%

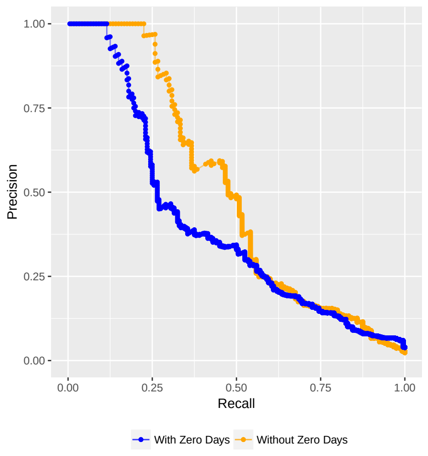
[1] This percentage is estimated from Figure 5 in their report[3].

in the *realistic* model. When the zero-day exploits were removed from Bullough et al.'s model, their class balance went from 17% to about 1.4%. In such a scenario, a significant performance decrease should be expected.

A possible explanation to why our performance is decreased when including zero-days is that those vulnerabilities are of a different class, i.e. their representations in the data are different than vulnerabilities that were exploited after publication.



Naive Model (a)



Realistic Model (b)

Fig. 5. Performance comparison of our naive model (a) and our realistic model (b) when including zero-day exploits in the data.

Table 5. Comparison of including zero-days for the naive, realistic and the results from Bullough et al. [3]. Values of precision and recall have been chosen to maximize the F_1 score.

	Naive		Realistic		Bullough et al.	
	Baseline	Experiment	Baseline	Experiment	Baseline	Experiment
Precision	0.525	0.335	0.578	0.352	0.171	0.519
Recall	0.333	0.247	0.458	0.440	0.027	0.334
F_1 Score	0.407	0.284	0.511	0.391	0.046	0.406

4.3 Experiment II: Temporal Intermixing

When using supervised learning on models that exhibit concept drift (see Section 2.2), one needs to keep the training and test sets temporally separated to establish the performance of the model when applied to new samples. This experiment will test if the data used for predicting exploits shows signs of concept drift. To test this, we compare a temporally separated training and test set with temporally intermixed training and test set using $d' = $ 2017-08-08 as the separating day between the training and the test set. The amount of concept drift will be the difference in performance between the intermixed and separated model.

As computed from the Table 7, the naive model has a 43.0% relative increase in maximum F_1 score, and the realistic model showed a 25.1% relative increase in F_1 score. These results indicate that both the *naive* and *realistic* models are prone to concept drift. For comparison, Bullough et al.'s model achieved a performance gain of 782.6% [3]. In the following paragraphs, we list three differences in our models that are likely to have contributed to the vast difference in performance gain between our models and the ones of Bullough et al.

Class balance: Some of the difference in performance between the baseline model and the experiment is due to a difference in class balance between the test sets of the baseline model and the experiment. To highlight this difference, we have included a Δ column in Table 6, which shows the difference in class balance between the regular model and the experiment in relative percentage. Bullough et al. [3] report a $\Delta = 55.7\%$[2]. Looking at the *realistic* model, we observe a $\Delta = 23.0\%$ for the test set. The extreme concept drift from Bullough et al.'s experiment should be partially attributed to their Δ being significantly higher than our *realistic* model.

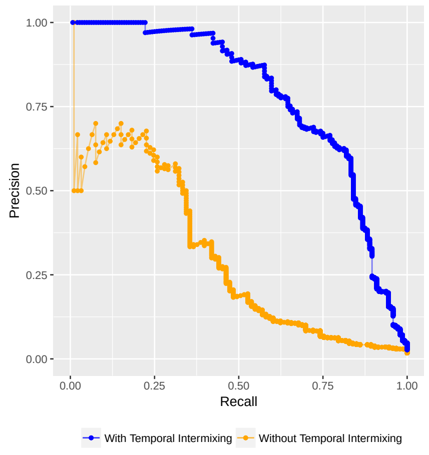
Absolute time frame: Since we use data that span 3 years (January 2015 to February 2018) our model has less time for concept drift to occur, compared to Bullough et al. [3] whose data span 6 years (2009 to 2015). Thus, Bullough et al.'s larger absolute time frame could be a possible explanation for their model exhibiting concept drift much larger than ours.

Relative time frame: When comparing the concept drift between the *naive* and *realistic* models, we are effectively comparing how the aggregation time frame of each sample impacts concept drift. As mentioned earlier, the *naive* model is more prone to concept drift than the *realistic*. This means that both the *absolute time frame* and *relative time frame* affect concept drift.

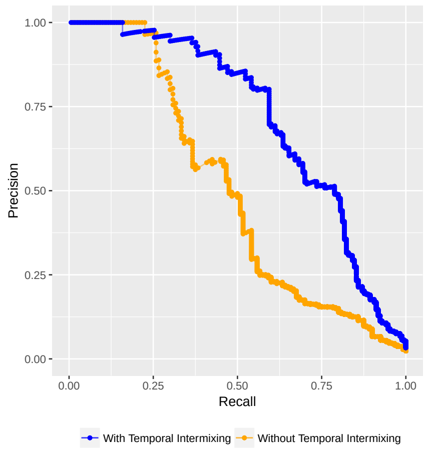
Table 6. Properties of our training and test sets for the naive and realistic models when randomly sampling their respective observations. The Δ column shows the relative difference in percentage of samples from the baseline model to the experiment model.

Samples		Naive			Realistic		
		Baseline	Δ	Experiment	Baseline	Δ	Experiment
Exploits	Total	757 (2.86%)	0%	757 (2.86 %)	842 (3.3 %)	0 %	842 (3.36 %)
	Dropped	354 (1.33%)	-100%	0 (0 %)	66 (0.03%)	-22.7%	51 (0.02 %)
	Train	310 (1.85%)	43.0 %	548 (2.88%)	656 (3.59%)	-4.3 %	686 (3.36 %)
	Test	93 (1.74%)	42.8 %	144 (2.72%)	120 (2.33%)	23.0 %	156 (3.05 %)

[2] The Δ was computed from their reported class percentage of their test set which was 16.7% in their random split experiment and 9.3% in their temporally split model.



Naive Model (a)



Realistic Model (b)

Fig. 6. Performance comparison of our naive model (a) and our realistic model (b) of splitting training and test sets temporally and using random sampling of training and test.

Table 7. The classifier results from the maximum F_1 score when doing temporal intermixing of training and test sets. We included results from previous research for comparison as well as a model that does not use temporal intermixing [3, 12, 5].

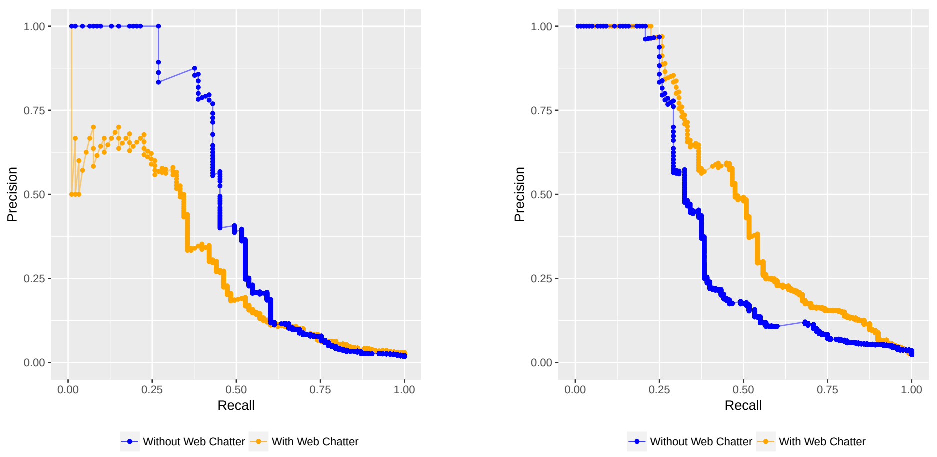
	Naive		Realistic		[3]		[12]	[5]
	Baseline	Experiment	Baseline	Experiment	Baseline	Experiment	Experiment	Experiment
Precision	0.525	0.664	0.578	0.801	0.171	0.519	≈ 0.20	0.8158
Recall	0.333	0.770	0.458	0.594	0.027	0.334	≈ 0.70	0.8302
F_1 Score	0.407	0.713	0.511	0.682	0.046	0.406	0.31	0.8229

4.4 Experiment III: Excluding Web Chatter

In this experiment we remove any features from our data set that relate to *web chatter* data. The reason for this experiment is to determine the role of web chatter's impact on our predictions. Previous work by Bullough et al. showed that including web chatter features had negligible impact on the performance of their model. To test this result, we designed similar models which excludes the web chatter feature group from our data frames, and compare performance of the resulting classifiers with their respective baseline models (*naive* and *realistic*).

In Figure 7a, we observe that excluding web chatter (blue line) from the *naive* model achieves higher precision for recall values in the range 0 to ≈ 0.60. This means that the naive model benefits from excluding *web chatter*. However, as seen in Figure 7b, excluding web chatter (blue line) in the *realistic* model achieves considerably worse precision for recall values in the range 0.25 and ≈ 1.

This result indicates that the web chatter features are adding irrelevant information under the *naive* model. Conversely, the realistic model's performance decreased significantly when excluding web chatter features. This result indicates that the web chatter has a positive impact under the realistic data model. Finally, when comparing the performance of the naive model excluding web chatter (blue line in Figure 7a), to the realistic model including web chatter (orange line 7b), we observe that the realistic data model still makes better predictions. This result indicates that the information disseminated during the early days of a vulnerability is enough to make the best exploit prediction possible.



Naive Model (a) Realistic Model (a)

Fig. 7. Performance comparison of models that exclude *web chatter* features with the baseline models.

Table 8. The classifier results from the maximum F_1 score on the test set when using *web chatter*. We included results from previous research for comparison as well as a model that does not use temporal intermixing.

	Realistic		Naive		Bullough et al.	
	Baseline	Experiment	Baseline	Experiment	Experiment	Baseline
Precision	0.525	0.740	0.578	0.777	0.466	0.426
Recall	0.333	0.430	0.458	0.291	0.342	0.311
F_1 Score	0.407	0.544	0.511	0.424	0.394	0.359

5 Discussion

In Section 4.3, we observed that the model exhibited concept drift during the time period 2015 to 2018. When a model exhibits concept drift, its ability to predict new samples degrades over time. This result corroborates the results of Bullough et al. who first made this discovery in 2017 [3]. Knowing more about when and how concept drift occurs would not only benefit predictive models for exploits but could yield insight about trends in exploit development.

The drastic decrease in percentage of zero-day exploits, from $\approx 90\%$ of all exploits published during the time period 2009 to 2015 [3] to about 30% during 2015 to early 2018, indicates that concepts or labels vary over time. This makes individual studies difficult to compare. However, when we included zero-day exploits in our model, we observed a decrease in performance which was the opposite of what previous research observed [3]. A possible explanation for this decrease is that zero-day exploits constitute a third class and introduce confusion to our model when trying to fit the new class with the larger exploit class.

Our results show that having a smaller relative time of aggregation can reduce concept drift for a fixed absolute time frame. However, we never compare different absolute time frames. It is possible that the effect of the relative time frame has been exaggerated since the absolute time frame is still large. Untangling the problem of the relative and absolute time frame's impact on concept drift is an interesting line of inquiry, that has practical implications for when the model needs to be retrained, and is left for future work.

The utility of any predictive model is contingent on its quality of labels. Exploit DB is known to contain many proof of concept exploits which usually require advanced skills to be used in attacks against a system. Therefore, Exploit DB's credibility as a proxy for real world exploits is questionable. For any model to have real world application, its ground truth needs to be upgraded or treated differently to reflect real world threats.

6 Conclusions

In this paper, we investigated the feasibility of predicting exploits using a realistic model of aggregating data. Through this model, we have found two conflicting results to those presented by Bullough et al. [3]. We found that open web data increases the predictive power of exploits and that using zero-day vulnerabilitieshas a negative impact on exploits. We also learned that the data in this domain has undergone concept drift during the time period between January 2015 and February 2018. This result is in agreement with those of Bullough et al. [3] which means that more effort have to be devoted to understand when concept drift occurs to make timely updates of models that predict exploits.

Our main finding in this paper is that to make realistic predictions on vulnerabilities, it is imperative to use a model that reflects the early state of knowledge of vulnerabilities. This is likely the information that exploit developers use to decide which vulnerabilities to focus their attention on.

7 Acknowledgements

The research leading to these results has been partially supported by the Swedish Civil Contingencies Agency (MSB) through the project "RICS" and by the European Community's Horizon 2020 Framework Programme through the UNITED-GRID project under grant agreement 773717.

We would also like to thank Staffan Truvé and Michel Edkrantz at Recorded Future for inspiration, access to data and the environment to perform the current study.

References

1. Allodi, L., Massacci, F.: Comparing Vulnerability Severity and Exploits Using Case-Control Studies. ACM Trans. Inf. Syst. Secur. **17**(1), 1:1–1:20 (Aug 2014). https://doi.org/10.1145/2630069, http://doi.acm.org/10.1145/2630069
2. Bozorgi, M., Saul, L.K., Savage, S., Voelker, G.M.: Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 105–114. KDD '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1835804.1835821
3. Bullough, B.L., Yanchenko, A.K., Smith, C.L., Zipkin, J.R.: Predicting Exploitation of Disclosed Software Vulnerabilities Using Open-source Data. In: Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics. pp. 45–53. IWSPA '17, ACM, New York, NY, USA (2017), http://doi.acm.org/10.1145/3041008.3041009
4. Chen, T., He, T., Benesty, M., et al.: Xgboost: extreme gradient boosting. R package version 0.4-2 pp. 1–4 (2015)
5. Edkrantz, M., Said, A.: Predicting cyber vulnerability exploits with machine learning. In: SCAI (2015)
6. Exploit-DB Offensive Securitys Exploit Database Archive. https://www.exploit-db.com/, accessed: 2017-08-24
7. Friedman, J.H.: Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics **29**(5), 1189–1232 (2001), http://www.jstor.org/stable/2699986
8. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM computing surveys (CSUR) **46**(4), 44 (2014)
9. National Vulnerability Database Computer Security Resource Center. https://nvd.nist.gov/, accessed: 2017-08-24
10. Recorded Future's threat intelligence platform
11. Roytman, M.: Quick Look: Predicting Exploitability, Forecasts for Vulnerability Management (2018), https://www.rsaconference.com/videos/quick-look-predicting-exploitabilityforecasts-for-vulnerability-management
12. Sabottke, C., Suciu, O., Dumitras, T.: Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting Real-World Exploits. In: 24th USENIX Security Symposium. USENIX Association, Washington, D.C. (2015)
13. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine learning **23**(1), 69–101 (1996)