Using Twitter to Predict When Vulnerabilities will be Exploited

Haipeng Chen*, Rui Liu
Dartmouth College
Hanover, NH, USA
haipeng.chen,rui.liu.gr@dartmouth.edu

Noseong Park
George Mason University
Fairfax, VA, USA
npark9@gmu.edu

V.S. Subrahmanian
Dartmouth College
Hanover, NH, USA
vs@dartmouth.edu

ABSTRACT

When a new cyber-vulnerability is detected, a Common Vulnerability and Exposure (CVE) number is attached to it. Malicious "exploits" may use these vulnerabilities to carry out attacks. Unlike works which study if a CVE will be used in an exploit, we study the problem of predicting *when* an exploit is first seen. This is an important question for system administrators as they need to devote scarce resources to take corrective action when a new vulnerability emerges. Moreover, past works assume that CVSS scores (released by NIST) are available for predictions, but we show on average that 49% of real world exploits occur before CVSS scores are published. This means that past works, which use CVSS scores, miss almost half of the exploits. In this paper, we propose a novel framework to predict when a vulnerability will be exploited via Twitter discussion, without using CVSS score information. We introduce the unique concept of a family of CVE-Author-Tweet (CAT) graphs and build a novel set of features based on such graphs. We define recurrence relations capturing "hotness" of tweets, "expertise" of Twitter users on CVEs, and "availability" of information about CVEs, and prove that we can solve these recurrences via a fix point algorithm. Our second innovation adopts Hawkes processes to estimate the number of tweets/retweets related to the CVEs. Using the above two sets of novel features, we propose two ensemble forecast models FEEU (for classification) and FRET (for regression) to predict when a CVE will be exploited. Compared with natural adaptations of past works (which predict if an exploit will be used), FEEU increases F1 score by 25.1%, while FRET decreases MAE by 37.2%.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Machine learning**; • **Networks** → *Network algorithms*;

KEYWORDS

Vulnerability Exploit Prediction; Web Data Mining; Cyber Security

ACM Reference Format:
Haipeng Chen*, Rui Liu, Noseong Park, and V.S. Subrahmanian. 2019. Using Twitter to Predict When Vulnerabilities will be Exploited. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3292500.3330742

1 INTRODUCTION

The number of software vulnerabilities disclosed every year is staggering (cf. Figure 1), leading to increasing risks for system security officers. Because patching is expensive (time for patch installation time, patch purchase and risk of disruption to production systems), many vulnerabilities go unpatched because of limited resources to tackle thousands of patching tasks. [1] It is therefore critical that we prioritize patching by predicting when a vulnerability will be exploited. Intuitively, vulnerabilities that exist in an enterprise that are likely to be exploited soon should be patched before those that might be exploited later.

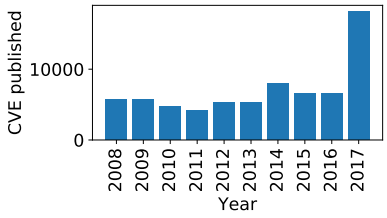


Figure 1: Number of CVEs published by NIST in the NVD over the past decade.

When a white hat hacker or security firm reports a new vulnerability, a Common Vulnerability and Exposure (CVE) numbering authority [2] assigns a CVE number to it at time t_{cve}. Later, after analysis of the vulnerability, the US National Institutes of Standards and Technology (NIST) releases a Common Vulnerability Scoring System (CVSS) score at time $t_{cvss} > t_{cve}$. For instance, on Jul. 31, 2017, MITRE assigned CVE-2017-11882 to a new vulnerability, along with the comment: *A variety of Microsoft Office versions, including Microsoft Office 2007, 2010, and 2016, allow an attacker to run arbitrary code in the context of the current user by failing to properly handle objects in memory, aka "Microsoft Office Memory Corruption Vulnerability".* This vulnerability was exploited on Nov. 22, 2017, but a CVSS score was released only on Dec. 15 2017. Though the information disclosed at this stage is limited, it can provide enough information for hackers to craft attacks.

If t^e_{cve} is the earliest time at which a given CVE is exploited, then we must consider two cases: (i) $t_{cvss} \leq t^e_{cve}$. In this case, an exploit uses the vulnerability after NIST assigns a CVSS score to it. (ii) $t^e_{cve} < t_{cvss}$. In this case, the vulnerability is exploited before NIST assigns the CVSS score. There have been various past works on predicting *if* a vulnerability will be exploited. However, most of these past works are methodologically flawed:

- [8, 22] use 10-fold cross validation. Because 10-fold cross-validation splits data randomly into 10-folds, the 9 folds

* Corresponding author.

[1] https://www.forbes.com/sites/jasonbloomberg/2018/04/16/to-patch-or-not-to-patch-surprisingly-that-is-the-question/#6eb7f8ff58fe
[2] The National Cybersecurity FFRDC at MITRE Corporation is the principal CVE numbering authority.

used for training will almost certainly contain data from the future, and then use those to "predict" some exploits used in the past. This happens because they ignore the fact that the time frames t_{cve}, t_{cvss} and t^e_{cve} must be taken into account. This flaw was also described in [2, 4].

- The later two studies [2, 4] however have another flaw. They, along with other past works on this problem, assume that CVSS scores are available at the time of prediction. However, as shown in Figure 2, there is a gap of several months between t_{cve} and t_{cvss}. In fact, on average, $t_{cvss} - t_{cvs}$ is 132.7 days. In our dataset spanning July 1 2016 to May 31 2018, we found that over 85% of PoC exploits and 49.46% of real world exploits occur during this gap. Because all past works on this problem including [4] and [2] use CVSS scores, they too end up using data from the future (CVSS scores) to make predictions about the past.

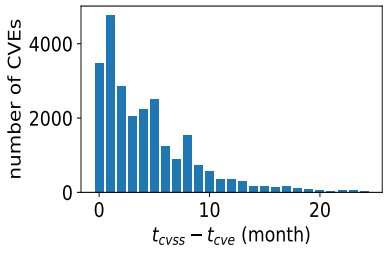


Figure 2: Time between the date NIST publishes CVSS score of a CVE and the date MITRE assigns a CVE ID to that CVE.

Moreover, all past efforts only try to predict ***if*** *a CVE will be used in an exploit, not* ***when****.* However, knowing when a CVE will be used is very important. System security officers need to prioritize which vulnerabilities to patch and when because patching is expensive. Intuitively, higher priority should be given to vulnerabilities that are likely to be exploited soon. We make the first attempt to predict *when* a vulnerability will be exploited, using *only CVE IDs and Twitter discussion data* without the need for CVSS scores. We further make the following contributions:

- First, to capture the popularity property of the CVEs, we propose the novel concept of a family of CVE-Author-Tweet (CAT) graph whose nodes are CVEs, Authors (Twitter user), and Tweets, respectively. We define recursive equations linking three popularity measures: the "hotness" of a tweet mentioning a CVE, the "expertise" of authors speaking about CVEs, and the "availability" of information about a CVE. We propose a Tri-Fixpoint Algorithm (TFIX) to solve this system of equations and prove that it has a unique solution that TFIX is guaranteed to find. These popularity measures add new features for our forecasting problem.
- Second, because our data is time-sensitive, all our testing uses the first t_{train} units of time for training, and then predicts out for the remaining time. As we cannot look at tweets after t_{train}, in order to improve prediction, we apply a Hawkes process model to estimate the retweet volume of a CVE after the training period. While retweet volume prediction with Hawkes process models was first proposed in [27], our contribution lies in applying it to get a new set of features for exploit timing prediction.
- Third, as shown in Figure 3, our FEEU (Forecasting Ensemble for Exploit Timing) and FRET (Forecasting Regression for Exploit Timing) predictors use an ensemble of retweet volume predictor with classical machine learning models. It extracts three types of features, i.e., basic Twitter discussion features which are used in prior art, the CVE popularity score features which are extracted from the CAT graphs we defined, and the future retweet volume feature that is obtained from the Hawkes process predictor.
- We conduct extensive experiments to evaluate FEEU and FRET. Though there are no past works that predict when a CVE will be exploited, we show that FEEU and FRET averagely outperform natural adaptations of past work [2, 4, 8, 22, 26] by 25.1% for classification (in terms of F1 score) and 37.2% for regression (in terms of MAE). Our methods are *statistically better* than adaptations of past works with p-value < 0.001. FRET is able to accurately predict the time of PoC exploits within 36 days and the time of real-world exploits within 12 days. We also show through case studies that FEEU and FRET are able to predict the real-world exploits of the two popular CVEs in 2017 with high probability.

2 PROBLEM DESCRIPTION

2.1 Vulnerability Exploit Prediction Problem

There are two types of exploits. *Real-world exploits* are exploits that were used in real-world attacks, while *proof-of-concept (PoC) exploits* are ones where someone (usually a white hat) generates sample exploit code to demonstrate the vulnerability. As shown in Figure 1, thousands of vulnerabilities are disclosed each year, and the number of vulnerabilities being disclosed is increasing. Because of this, system security officers (SSOs) should patch vulnerabilities as soon as possible. However, this is not always done because patching induces considerable cost [9] (e.g. patch installation time, patch purchase, disruption to production system, etc.). Predicting when a vulnerability will be exploited is therefore incredibly helpful in determining what to patch and when.

2.2 Related Work

Several past studies have used machine learning to predict if a vulnerability will be exploited — but none predict *when*. [3] were the first to use machine learning for PoC exploit prediction. The features used were derived from the OSVDB[3] (Open Source Vulnerability Database) [13] and CVE [6]. [8] used several ML approaches with features derived from the National Vulnerability Database (NVD) [19] and ground truth from EDB [23]. These two studies predict PoC exploits rather than real world exploits. [22] developed methods to predict both PoC and real-world exploits. They obtained ground truth about real-world exploits from the Symantec Attack Signature [25] and Intrusion Protection Signature [24] data and

[3] OSVDB is no longer publicly available since 2016.

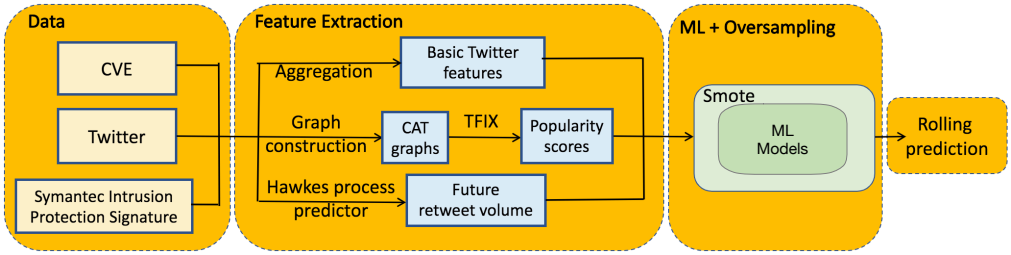


Figure 3: General framework of FEEU and FRET. FEEU predicts whether a CVE will be exploited within the next 1/3/6/9/12 months. FRET predicts exactly the date at which a CVE will be exploited.

Table 1: Related work on vulnerability exploit prediction.

Paper	Datasets	ML approach	Task	Notes
[3]	OSVDB, MITRE	Linear SVM	PoC	The first work on ML-based vulnerability exploit prediction
[22]	NVD, OSVDB, Twitter, EDB, Symantec	Linear SVM	PoC & real	First to use social media
[8]	NVD, RF, EDB, Symantec	SVM, RF, NB	PoC	RF (Recorded Future) is commercial
[4]	NVD, EDB, Twitter	Linear SVM	PoC	Studies the effect of train/test splits and imbalance of data
[2]	NVD, EDB, ZDI, DW, Symantec	SVM, RF, NB, LR	real	ZDI (Zero-Day Initialtive) is commercial
[26]	NVD, EDB, DW	Embedding + SVM/RF	real	First to use paragraph embedding for feature pre-processing

from the NVD and OSVDB. They were the first to show that prediction accuracy could be improved by using (one year of) Twitter data. Bullough et al. [4] at MIT pointed out major drawbacks in previous studies. Because [3, 8, 22] did not consider the temporal relationship between the time t_{cvss} NIST publishes the CVSS score for a CVE and the time t^e_{cve} when an exploit using the CVE is first seen, they used 10-fold cross validation. In 10-fold CV, the data is randomly split into 10-folds. Iteration over the 10-folds is done by selecting one fold for testing, and training the model on the remaining 9 folds. The problem with this approach is that the test fold may contain data from the past, while the training folds might contain data about the future. As a result, prediction results are flawed because it is much easier to predict the past than the future.

Real-world exploits were recently studied in [2], where, in addition to NVD and EDB[4], data from the Zero Day Initiative (ZDI) [12] and darkweb and deepweb (DW) posts are used. The ground truth is derived from the Symantec Attack Signature and Intrusion Protection Signature. Different machine learning algorithms are examined for the final prediction, including SVM, NB, LR and RF. Further exploration of DW data was made in [26], where DW posts were preprocessed with the *paragraph embedding* [18]. The embeddings are used as additional features.

Table 1 provides a summary of papers related to vulnerability exploit prediction. In addition to the flaw of using cross validation instead of a rolling window prediction (which is more appropriate for temporal data), all existing studies (except one who used the OSVDB which is no longer publicly available [3]) cited above use CVSS scores which, as shown in Figure 2, are published on average 132.7 days *after* CVE numbers are assigned (e.g. by MITRE). Again, this means that past works use *future* data (CVSS score) to predict what happened in the past. In fact, past works show that CVSS scores appear to be among the most important features. This means that the results are unreliable, especially as our 23 months of data shows that 85.75% of PoC exploits and 49.46% of real world exploits happen before NIST publishes CVSS scores, making the past works incapable of detecting them before they occur.

[4]Note that EDB is used to extract features, not ground truth.

2.3 Dataset Description

Our dataset consists of 23 months of data (1/Jul/2016 - 31/May/2018).

CVE [6] We study the set of all CVEs (26,093 in all) maintained by MITRE during this time frame. Each vulnerability is assigned a CVE ID. Figure 4 shows a distribution of the CVEs published over the 23 months.

CVE-related Twitter Discussion As in past work [22], we extract all tweets that have the "CVE" keyword in them during the reference time period. In total, our Twitter dataset contains 632,873 tweets and 51,214 authors. Figure 4 shows statistics about Twitter discussion volume over the target period.

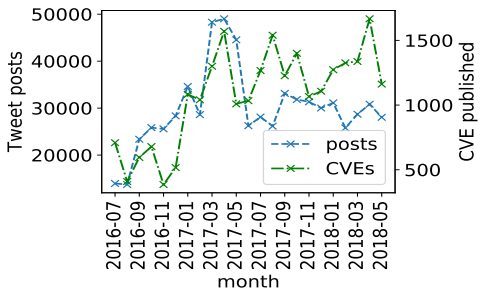


Figure 4: Distribution of Twitter discussion and CVEs published during 01/Jul/2016-31/May/2018. The x-axis denotes the time in month, the left y-axis is the number of tweets, and the right y-axis is the number of CVEs published.

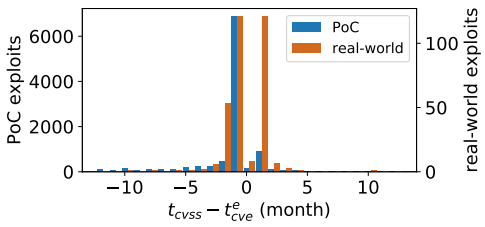


Figure 5: Distribution of time difference between the date the PoC (resp. real-world) exploit of a CVE is published in EDB (resp. Symantec's Intrusion Protection Signature) and the date the CVE is published in NVD. The x-axis is the date of NVD publishing minus the exploit date (in months), and the y-axis represents the number of exploits.

Exploit DB (EDB) [23] Maintained by Offensive Security, EDB is a CVE compliant archive of public PoC exploits and the corresponding vulnerable software. EDB tells us when a PoC exploit was first seen for a given CVE. Note that a PoC exploit may use multiple CVEs, and a CVE might be used in one or more PoC exploits. For example, CVE-2017-0148 has two associated PoC exploits in the EDB (exploit41891 and xploit41987). If two or more PoC exploits are found for one CVE, we use the earliest date among all the exploits as the date that the CVE was first exploited.

Symantec Intrusion Protection Signature [24] This dataset records up-to-date real-world exploits of vulnerabilities, together with the reported date that the exploit was first seen. We use this dataset to generate the ground truth labels for our training and test data.

2.4 Preliminary Data Analysis

In this subsection, we discuss two key facts that motivate our Twitter-based vulnerability exploit prediction algorithm.

Fact 1. *As shown in Figure 2, during our 23-month period of study (July 1 2016 to May 31 2018), there is an average (resp. median) gap of* 132.7 *(resp.* 103*) days between the time a CVE number is assigned by MITRE and the time that associated CVSS scores are published by NIST.*

Fact 2. *As shown in Figure 5, for CVEs during our period of study, 85.75% of their PoC exploits and 49.46% of their real-world exploits happened before NIST publishes the CVSS score.*

These two facts indicate that a huge portion of vulnerability exploits cannot be predicted by previous studies since these studies rely heavily on CVSS scores extracted from NVD. Thus, the overall goal of this paper is to *use the CVE IDs from MITRE's CVE [6] dataset, together with features extracted from CVE-related Twitter data, to predict when the CVE will be used by either 1) a PoC exploit or 2) a real-world exploit.* We propose two broad methods to do so: we build classifiers to predict whether an exploit will be used in 1/3/6/9/12 months and then we build a regression model to predict the exact day when a vulnerability will be exploited. The two methods complement each other and provide a comprehensive understanding of vulnerability exploit timing.

3 FORECAST ENSEMBLE FOR EXPLOIT TIME

This section introduces the prediction method. We start by defining two novel sets of features used by both FEEU and FRET. First, we define a family of CVE-Author-Tweet (CAT) graphs to define the intrinsic popularity property of the CVEs under equilibrium. Second, we apply a separate predictor that estimates *future* CVE-related tweet/retweet volume using the Hawkes process model. The third subsection describes some basic features which are also used in past work. The basic features, the meta CVE popularity features, the Hawkes process predictor, plus a suite of ML algorithms are fed into a late fusion ensemble predictor to generate the final result.

3.1 Extract Meta-Features from a Multi-Layer Graph of CVE, Author and Tweets

Graphs have been extensively used in identifying malicious URLs[17], review fraud[1], and predicting spread of malware[14]. In this paper, we define a novel concept of the CVE-Author-Tweet (CAT) family of graphs based on which we define new "popularity" features of the nodes using a recursively defined linear equation system. The intuition is that the more "popular" a CVE is in social media (Twitter exposure in our setting), the more likely that CVE is to be exploited. Second, the "hotness" of tweets, the "expertise" of the authors and the "available" information on the CVE are mutually influenced by each other. We show that this set of equations can be solved by a Tripartite Fixpoint algorithm (TFIX) and that it returns a unique solution to the system of equations.

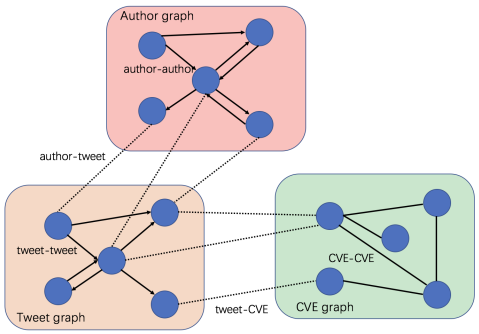


Figure 6: The multi-layer graph model for exploit probability prediction which consists of three layers (sub-graphs) and cross-layer edges. The solid-line edges are within-layer edges, while the dotted-line edges are cross-layer edges.

3.1.1 CAT: a Multi-Layer Graph Representation of CVE, Author, Tweet Relations. We now define a directed multi-layer graph[7, 15] called CAT (CVE-Author-Tweet). Figure 6 is a sample of our multi-layer CAT graph consisting of three interlinked layers: a CVE-graph, an author graph, and a tweet graph. We will specify the different components as follows.

Tweet graph. A *tweet graph* $G_T = (V_T, E_T)$ is a directed graph where each node in the set V_T is a tweet. We allow three types of tweet-tweet edges in E_T: (i) There is a *retweet edge* $e^T_{ji} \in E_T$ if tweet v^T_j is a retweet of tweet v^T_i. (ii) There are two *tweet-cve-tweet edges* $e^T_{ji}, e^T_{ij} \in E_T$ if both tweets v^T_j and v^T_i mention the same CVE. (iii)

There are two *hashtag edges* $e^T_{ji}, e^T_{ij} \in E_T$ if both tweets v^T_j and v^T_i share the same hashtag topic.

Author graph. An *author graph* $G_A = (V_A, E_A)$ is a directed graph where each node in the set V_A is a Twitter user. An author-author edge $(v^A_i, v^A_j) \in E_A$ indicates a relation between two author nodes. Two types of author-author edges are defined: (i) There is a *follower-followee edge* $e^A_{ji} \in E_A$ if an author v^A_j is a follower of v^A_i. (ii) There is a *mention edge* $e^A_{ji} \in E_A$ if author v^A_i is mentioned by v^A_j.

CVE graph. A *CVE graph* $G_C = (V_C, E_C)$ is a directed graph where each node $v^C_i \in V_C$ represents a CVE. A CVE-CVE edge ($e^C_{ji} \in E_C$ indicates that CVE v^C_j is linked to CVE v^C_i. We define: There are two *CVE-tweet-CVE edges* $e^C_{ji}, e^C_{ij} \in E_C$ if two CVEs v^C_i and v^C_j are both mentioned in the same tweet.

Cross-layer edges. Except for the edges within each sub-graph (layer), CAT graphs also contain cross-layer edges: (i) An *author-tweet edge* $(e^{AT}_{ij}) \in E$ exists if the tweet $v^T_j \in V_T$ is created by the author $v^A_i \in V_A$. (ii) A *tweet-CVE edge* $e^{TC}_{ij} \in E$ exists if the tweet $v^T_i \in V_T$ mentions CVE $v^C_j \in E_C$.

Note that we do not consider direct edges between CVEs and authors as they are indirectly linked by the tweet nodes. Thus, a CAT graph $G = (G_C, G_A, G_T, E)$ consists of three subgraphs ("layers"), G_T, G_A, G_C as well as some additional "cross edges" E across the layers. With different definitions of edges within each subgraph, we have 3 types of tweet-tweet edges, 2 types of author-author edges and 1 type of CVE-CVE edges. This in total constitutes a combination of $3 \times 2 \times 1 = 6$ types of CAT graphs. For all types of CAT graphs, the two kinds of cross-layer edges always exist.

3.1.2 Popularity Properties of the Nodes. For each node $v^x_i, x \in \{C, A, T\}$, we define a popularity score $ps(v^x_i)$ to measure the importance of a node within the CAT graph, where $ps(v^A_i)$ indicates the "expertise" of the author v^A_i, $ps(v^T_i)$ reveals the "hotness" of the tweet v^T_i, and $ps(v^C_i)$ shows the "availability" of the CVE v^C_i. We would like the popularity scoring method to satisfy some axioms.

Axiom 1. *Consider two identical CAT graphs with one exception. In the first graph (as compared to the second), there is one node $v^x_i, x \in \{C, A, T\}$ with an additional incoming edge e_{ji} from another node $v^{x'}_j, x' \in \{C, A, T\}$. In this case, the node v^x_i in the first graph should have a higher popularity score than the same node in the second graph, i.e. $ps(v^x_i) \leq ps'(v^x_i)$ should hold.*

This axiom indicates that a node which has a higher in-degree in the CAT graph should be more popular.

Axiom 2. *Consider two identical CAT graphs with one exception. In the first graph (as compared to the second), there is one node $v^x_i, x \in \{C, A, T\}$ such that one of its incoming edges (from node $v^{x'}_j$) is replaced so that it comes from another node $v^{x''}_k$ such that the latter has a higher popularity score than the former. In this case, the popularity score of v^x_i in the second graph should be higher than its popularity in the first graph.*

This axiom indicates that if a node is connected to a more popular node, its popularity score should go up.

3.1.3 Node Popularity under Equilibrium. Based on the axioms above, we now derive a system of linear equations to define the equilibrium popularity measure of all the three types of nodes. Before that, we first define a weighted graph with the following two types of edge weights.

Definition 1 (Type I edge weights). *The edge weights $w^{xx'}, x, x' \in \{C, A, T\}$, which are determined only by the types (i.e., $x, x' = C, A, T$) of the two nodes, are called Type I edge weights.*

This weight is used to distinguish the importance of different types of endpoints of an edge. For instance, an incoming edge from an author node should have higher importance than an incoming edge from a tweet node.[5] Note that the Type I edge weights do not depend on the specific starting/ending endpoints, but only on the types of the nodes.

Definition 2 (Type II edge weights). *The edge specific weights $z^{xx'}_{ji}, x, x' \in \{C, A, T\}$, which are determined by the specific starting endpoint v^x_j to the ending endpoint $v^{x'}_i$ of layers G_x and $G_{x'}$, are called Type II edge weights.*

Type II edge weights capture importance of the start/end points of an edge. For example, if two authors are connected to the same author via two author-author "mention" edges, the weights of the two edges should depend on the number of mentions. Similarly, if two authors are connected to the same CVE via author-CVE "mention" edges, the weights should depend on the number of mentions. For each node, the sum of weights of all outgoing edges of the same type of edges equals 1, i.e.,

$$\sum_{e^{xx'}_{ij} \in G} z^{xx'}_{ij} = 1, \forall x' \in \{C, A, T\}. \tag{1}$$

These three equations per node (one each for $\{C, A, T\}$) normalizes type II edge weights.

The final edge weight $W^{xx'}_{ji}$ from node v^x_j to node $v^{x'}_i$ is a product of the two, i.e., $W^{xx'}_{ji} = w^{xx'} \cdot z^{xx'}_{ji}$. With the above definition of weights and node popularity score, we define a recursive set of linear equations to characterize the mutual influence of the popularity scores among the different nodes in the CAT graph. For the final edge weights, we require the sum of weights of all outgoing edges from a node to be 1, i.e.,

$$\sum_{e^{xx'}_{ij} \in G} w^{xx'} z^{xx'}_{ij} = 1 \tag{2}$$

Note that unlike Eq. (1) where the summation includes edges which go to a same type x' of end node, the summation in this equation includes all the outgoing edges.

We now define our mutually recursive system of linear equations. First, the popularity score of an author $v^A_i \in G_A$ is represented as a weighted sum of popularity scores of his neighboring authors and tweets. The popularity score of an author is designed to reveal the "expertise" level of the author on CVEs and exploits.

$$ps(v^A_i) = \sum_{e^A_{ji} \in G_A} w^{AA} z^{AA}_{ji} ps(v^A_j) + \sum_{e^{AT}_{ji} \in E} w^{AT} z^{TA}_{ji} ps(v^T_j). \tag{3}$$

[5] As one author can create multiple tweets, an author is on average much more important than a tweet.

Input: Edge weights matrix $W = \langle W_{ij}^{xx'} \rangle$ in the CAT graph;
Output: Equilibrium popularity score $ps(v^x Ii), \forall v^x Ii \in G$;
Initialize $ps(v_i^x) = 1, \forall v_i^x \in G_x, \forall x \in \{C, A, T\}$;
while *$ps(v_i^x)$ does not converge* $\forall v_i^x \in G$) **do**
  for $v_i^A \in G_A$ **do**
    Get $ps'(v_i^A)$ using Eq. (3)
  for $v_i^T \in G_T$ **do**
    Get $ps'(v_i^T)$ using Eq. (4);
  for $v_i^C \in G_C$ **do**
    Get $ps'(v_i^C)$ using Eq. (5);
  $ps(v_i^x) = ps'(v_i^x)$ for each vertex v_i^x

Algorithm 1: TFIX

The popularity score of a tweet $v_i^T \in G_T$ is a weighted sum of its neighbouring tweets, author, CVEs. The popularity score of a tweet captures the "hotness" of a the tweet.

$$ps(v_i^T) = \sum_{e_{ji}^T \in G_T} w^{TT} z_{ji}^{TT} ps(v_j^T) + \sum_{e_{ji}^{AT} \in E} w^{TA} z_{ji}^{AT} ps(v_A^j) + \sum_{e_{ji}^{CT} \in E} w^{TC} z_{ji}^{CT} ps(v_C^j) \tag{4}$$

The popularity score of a CVE $v_i^C \in G_C$ is a weighted sum of the popularity score of its neighboring tweets and CVEs. The popularity score of a CVE characterizes the "availability" of information about a CVE to the public.

$$ps(v_i^C) = \sum_{e_{ji}^{TC} \in E} w^{CT} z_{ji}^{TC} ps(v_j^T) + \sum_{e_{ji}^{CC} \in G_C} w^{CC} z_{ji}^{CC} ps(v_j^C). \tag{5}$$

Since there are no cross-layer edges between the author and CVE graphs, the author and CVE nodes cannot be direct neighbors.

3.1.4 Equilibrium Popularity Score Calculation and Convergence. A straightforward way to solve our recursive equations is to iteratively perform matrix multiplication until convergence. However, the sizes of the weight matrix and popularity vector are enormous (the largest CAT graph constructed on our data has 710,180 nodes and 205,325,205 edges), and our machines do not have enough memory to store such large networks. Hence, we develop the Tri-Fixpoint Algorithm TFIX (Algorithm 11) to calculate the equilibrium popularity scores. The algorithm starts with a uniform initialization of the popularity score of all the nodes. It then proceeds in an iterative manner. In each iteration, the popularity scores are calculated using Eqs. (3)-(5). The process iterates until the popularity score converges for all the nodes in the CAT graph, which means the differences of the popularity scores calculated in two consecutive iterations are within a small threshold.

Theorem 3.1. *If the CAT graph is strongly connected, then Algorithm TFIX is guaranteed to converge, and the equilibrium popularity score calculated is unique.*

Proof. With a bit abuse of notation, we denote the weight matrix of CAT as W, and the popularity score vector as s, and the number of nodes in the CAT graph as n. In our TFIX algorithm, each iteration is equivalent to performing a matrix multiplication of the weight matrix W and the popularity score vector s. As a result, proving Theorem 3.1 is equivalent to proving that there exists a unique stationary s, such that $s = Ws$, which is further equivalent to proving that W has a unique eigenvector $\lambda = 1$.

First, the strong connectivity of our CAT graph implies that the weight matrix W is irreducible. According to the Perron–Frobenius theorem for non-negative matrices [10], a non-negative matrix has a unique largest real valued if it is irreducible. Thus, W has a unique largest real valued eigenvalue $\lambda_0 \in \mathcal{R}$, i.e., there exists an equilibrium s and a unique $\lambda_0 \in \mathcal{R}$ such that $s = \lambda_0 Ws$. In the next, we prove that $\lambda_0 = 1$.

According to the definition of eigenvalue, the transpose W^T of W has the same eigenvalue as W. Therefore, we have $W^T s' = \lambda_0 s'$, where s' is an eigenvector of W^T. That is $\sum_j W'_{ij} s'_j = \lambda_0 s'_i$, for $i = 1, \ldots, n$. Substituting W'_{ij} with W_{ji}, and summing up the equations for each $i = 1, \ldots, n$, we have

$$\sum_i W_{i1} s'_1 + \ldots + \sum_i W_{in} s'_n = \lambda_0 (s'_1 + \ldots + s'_n).$$

By rewriting Eq. (2) in the form of W and n, the coefficient of each element s'_i is 1, i.e.,

$$\sum_i W_{ij} = 1, \forall j = 1, \ldots, n.$$

Thus, we have

$$s'_1 + \ldots + s'_n = \lambda_0 (s'_1 + \ldots + s'_n),$$

which implies that $\lambda_0 = 1$.

□

Remark 1. While CAT graphs need not always be strongly connected as required by Theorem 3.1, we observed that all the different types of CAT graphs are very dense graphs, and the TFIX algorithm converges in all the cases.[6]

Remark 2. Our problem resembles HITS [16] where an iterative algorithm is used to obtain a solution. A major difference is that the transition (weight) matrix in their problem is a symmetric matrix (more specifically, a uniform matrix), while the weight matrix in our problem is not necessarily symmetric. Our problem also has similarities to the PageRank problem [20], where the row summation of the weight matrices for both of the two problems is 1. But PageRank assumes that each node has a probability of transferring to any other node (i.e., random surfing assumption) causing PageRank's weight matrix to be strictly positive which is not the case for us, making the proof of our theorem more complex.

3.2 Predicting Future CVE-Related Retweets Using Hawkes Process

As discussed previously, we perform a "rolling window" prediction where we learn models from data seen up to some time t_0 (t_0 would be greater than t_{cve}) and then predict whether the CVE would be used in an exploit in months $(t_0+1), (t_0+3), (t_0+6), (t_0+9), (t_0+12)$. Our 23 months of CVE-related Twitter data shows that 49.6% of real world exploits occur before the CVSS score is published and the number rises to over 85% for PoC exploits. Clearly, it would be helpful to know the future number of retweets about a CVE for making these predictions, but this information would not be in the training data. We therefore use *Hawkes process estimations* [11] to predict future retweet volume of a CVE after t_0 (which is when

[6] Moreover, even if CAT is not strongly connected, we can always treat the separate subgraphs as strongly connected graphs and extract popularity features from these subgraphs and then independently apply TFIX to each component.

the training data ends) and then use these estimates to improve prediction.

A Hawkes process is a self-exciting point-process model describing cascades of certain events originally used to model earthquakes. It has since been adapted to predict retweet activities [21, 27]. We use the Hawkes process model to estimate the number of future retweets that are related to each currently disclosed CVE. Note that our contribution does not lie in building the Hawkes process model for retweet volume prediction, but in applying the retweet volume prediction in existing works [21, 27] to improve our vulnerability exploit prediction.

Let R_t denote the total number of tweets (including retweets) that are related to a CVE. We define the *intensity* of tweets related to the CVE at time t as

$$\lambda_t \equiv \lim_{\Delta \to 0} \frac{R_{t+\Delta} - R_t}{\Delta}.$$

Formally, we model the intensity λ_t as

$$\lambda(t) = p(t) \sum_{k:t_k<t} n_k \phi(t - t_k) \tag{6}$$

where $p(t)$ is the infection rate at time t. Intuitively, the infection rate captures the probability of a Twitter user retweeting a tweet when the user is exposed to the tweet. $t_k, k = 1 \ldots, R_t$ is the occurrence time of the k^{th} tweet/retweet. n_k is the degree of the author node who writes the k^{th} tweet (i.e., the number of followers). $\phi(\cdot)$ is a function that describes the exponential decay of tweets along time defined as follows:

$$\phi(s) = \begin{cases} c, & \text{if } 0 < s \leq s_0 \\ c(s/s_0)^{-(1+\theta)}, & \text{if } s > s_0 \end{cases} \tag{7}$$

s_0 is interpreted as the reaction time of a tweet. c is a small constant. For a tweet event k related to a given CVE (that happened before time t), $n_k \phi(t - t_k)$ denotes the intensity of exposures to other users of the tweet k. Therefore, $p(t) n_k \phi(t - t_k)$ can be interpreted as the estimated intensity of retweet events related to the CVE.

The parameters c, s_0 and θ in Eq. (7) can be estimated using regression models. $p(t)$ can be estimated using the following formula:

$$\hat{p}(t) = \frac{\sum_{k=1}^{R_t} K_t(t - t_k)}{\sum_{k=0}^{R_t} n_k \int_{t_k}^{t} K_t(t - s) \phi(s - t_k) ds} \tag{8}$$

where $K_t(s)$ is a smoothing kernel. We use the same triangular kernel as in [27]:

$$K_t(s) = \max\{1 - 2s/t, 0\}. \tag{9}$$

For one thread of tweets/retweets that are related to one CVE, the final predicted number of tweets plus retweet size is

$$\hat{R}_\infty = R_t + \alpha_t \hat{p}_t (N_t - N_t^e)/(1 - \gamma \hat{p}_t n^*), \tag{10}$$

where $N_t = \sum_{k=0}^{R_t} n_k$ is the sum of degrees of all the authors for the retweets $k = 0, \ldots, R_t$, $N_t^e = \sum_{k=0}^{R_t} n_k \int_{t_k}^{t} \phi(s - t) ds$ is the effective sum of degrees of the authors for the retweets, n^* is the average degree of the tweet network. N_t can be interpreted as the total number of exposures of a tweet when there is no information decay, while N_t^e considers the decaying effect. $\alpha_t \in [0, 1]$ and $\gamma_t \in [0, 1]$ are parameters to offset the overlap of retweet activities of neighboring tweets. We leave out the derivation of Eqs. (8) and (10) and refer to [27] for the details. Our final prediction of all the future tweets/retweets related to a CVE summarizes all the estimated number of tweets $\hat{R}_\infty$ of different individual threads of tweets.

3.3 Basic Features

The set of basic features used in this paper (which are also used in existing approaches) include #tweets, #retweets, #replies of tweets and retweets, #tweets favorited, average #hashtags/URLs/user mentions per tweet, #verified accounts, average age of accounts, and average #tweets per account.

3.4 Ensemble Forecasting Model

As shown in Figure 3, our ensemble prediction models FEEU and FRET utilize a set of basic features, the popularity score features dervied from the CAT graphs, and the retweet volume features derived from the Hawkes process predictors, together with a suite of classical machine learning approaches. We also use rolling prediction as described earlier.

Machine learning models FEEU uses a set of classifiers (Support Vector Machines - SVM, Logistic Regression - LR, Naive Bayes - NB, Random Forest - RF, and XGBoost), while FRET uses a set of regressors Linear, Bayes, Random Forest, XGBoost, Lasso, and Ridge for predicting the exact date of exploit usage.

Over-sampling Exploit labels are imbalanced for *proof-of-concept exploits*, where 1, 459 of 26, 093 CVEs are exploited, and even more imbalanced for *real-world exploits*, where 141 are exploited. Thus, we use *SMOTE* over-sampling in the classification tasks [5].

Late fusion To further improve prediction, we also perform a late fusion over our proposed ML models. Basically, late fusion enhances prediction results by combining prediction results of multiple base ML models, each of which is trained by a specific feature or model. In our problem, late fusion does a weighted combination of the 5 base classifiers of FEEU for classification, and the 6 base regressors of FRET for regression.

4 EXPERIMENTAL EVALUATION

In this section, we evaluate the performance of FEEU and FRET. Due to space constraints, for both classification and regression tasks, we only show 1) the best of th baselines, 2) results of FEEU/FRET with different machine learning models, and 3) the late fusion results of our FEEU/FRET models. The detailed description of experimental settings is in the supplementary material.

4.1 FRET Regression Results

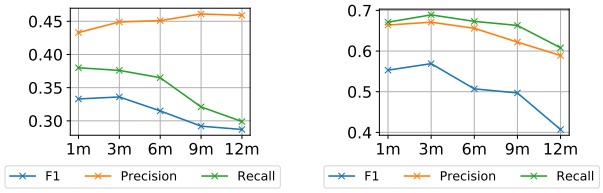
Table 3 compares the MAE of FRET with the baseslines [2, 4, 8, 22, 26]. The results show that (i) For both PoC and real-world exploit timing, FRET performs significantly better than the best of the baselines. In terms of real-world exploit prediction, FRET with late fusion decreases MAE by 37.2% compared with the best of baselines. This demonstrates that the popularity score and future tweet volume features are able to substantially improve our prediction. (ii) On average, the MAE of FRET-LateFusion is 35.71 days for PoC exploit prediction, and 11.90 days for real-world exploit prediction. The small p-value of 0.02 implies that the result is statistically significant.

Table 2: PoC classification, p-value < 0.001

Method	1 month			3 months			6 months			9 months			12 months		
	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall
FEEU-RF	0.297	0.433	0.242	0.319	0.449	0.255	0.291	0.442	0.220	0.273	0.426	0.204	0.255	0.409	0.191
FEEU-XGBoost	0.333	0.401	0.310	0.336	0.408	0.299	0.315	0.406	0.261	0.292	0.380	0.242	0.263	0.362	0.213
FEEU-SVM	0.108	0.125	**0.380**	0.098	0.105	0.303	0.109	0.120	**0.365**	0.113	0.160	0.151	0.098	0.140	**0.299**
FEEU-LR	0.186	0.170	0.253	0.197	0.189	0.223	0.173	0.184	0.272	0.168	0.221	0.142	0.165	0.210	0.231
FEEU-NB	0.265	0.215	0.368	0.29	0.239	**0.376**	0.287	0.248	0.344	0.288	0.263	**0.321**	0.287	0.277	**0.299**
FEEU-LateFusion	**0.344**	0.399	0.372	**0.345**	0.444	0.274	**0.322**	0.446	0.231	**0.316**	0.438	0.230	**0.301**	0.442	0.216
best of baselines (SMOTE)	0.248	0.405	0.192	0.271	0.432	0.212	0.265	0.451	0.199	0.260	0.461	0.194	0.259	0.459	0.197
best of baselines	0.171	**0.549**	0.109	0.187	**0.549**	0.118	0.181	**0.554**	0.109	0.181	**0.565**	0.109	0.172	**0.580**	0.102

Table 3: Rolling FRET regression MAE results, p-value=0.02

	PoC	Real-world
FRET-Bayes	36.50	11.95
FRET-RandomForest	45.06	19.75
FRET-XGBoost	38.54	19.95
FRET-LateFusion	**35.71**	**11.90**
best of baselines	50.04	18.95



(a) PoC tasks (b) Real-world tasks

Figure 7: Average performance for PoC and real-world rolling prediction tasks. X-axis denotes different prediction windows. Y-axis represents the metrics value.

4.2 FEEU Classification Results

To further evaluate the advantage of our two proposed sets of novel features, we decompose the regression task into a suite of classification tasks, which is predicted by the FEEU framework. For classification, we compare the F1 score, precision and recall with the baselines from [2, 4, 8, 22, 26]. Though the baselines didn't use sampling techniques, we present the baseline results both with and without *SMOTE* oversampling.

PoC Exploits Rolling Classification Table 2 shows the prediction results for PoC exploits. We note that: (i) In all cases, *the F1 scores and recalls of the baselines are improved with SMOTE oversampling, while precision is decreased due to the oversampling of positive labels.* (ii) Comparing FEEU with best of baselines with SMOTE, we see that *popularity score features and future retweet volume features substantially improve performance of PoC prediction w.r.t. all metrics.* Note that the p-value of this comparison is smaller than 0,001, which implies that this observation is statistically significant. Although the precision of best of baselines (without SMOTE) is the highest in all cases, they achieve a very low recall because without oversampling, the training dataset of the baselines is very imbalanced towards negative labels. (iii) *Nearer-future PoC exploits are more predictable than longer-term ones.* We can see that the best values of the performance metrics for nearer-future PoC predictions are higher than those for longer-term predictions. This observation is also demonstrated in Figure 7(a). (iv) Late fusion yields significant improvement compared with the best of base FEEU classifiers in terms of F1 score.

Real-World Exploits Rolling Classification Table 4 shows the rolling prediction results for *real-world* exploits. We see: (i) similar results to those for PoC exploit prediction are achieved - in particular, trend of prediction accuracy (cf. Figure 7(b)) is similar. (ii) We can see that, while FEEU with XGBoost has sometimes slightly lower precision than best of baselines (with or without SMOTE oversampling), it generally has much higher recall and F1 score values, with a p-value smaller than 0.001. Using late fusion over our FEEU models, we are able to increase F1 score by 25.1% compared with the best of baselines with oversampling.

4.3 Case Study

To further demonstrate the effectiveness of FEEU and FRET, we now discuss case studies of two of the most popular CVEs in 2017.

CVE-2017-0143 was involved in the infamous "EternalBlue" exploit targeting a Windows OS vulnerability (in its Samba protocol implementation) to carry out massive ransomware attacks. This CVE was created on 09/09/2016 and its first PoC exploit occurred on 04/14/2017 (217 days later). The CVE is discussed by 1,570 tweets and 1,080,049 retweets. CVE-2017-0143 has an average popularity score of 46.5 compared with 25.3 of the others. The retweet volume of CVE-2017-0143 predicted by the Hawkes process model is 13.3 times the average value for the other CVEs. Our proposed FEEU framework gives a high probability > 0.60 that the CVE will be exploited in six months. FRET framework predicts the CVE will be exploited in 113.84 days (compared with the ground truth of 189 days).

CVE-2017-11882 is a vulnerability in MS Office which allows malicious code to be remotely executed and takes over the machine to carry out its will. The CVE was assigned on 07/31/2017 and the first exploit occurred on 11/22/2017 (114 days later). This CVE triggers 6,404 tweets and 756,761 retweets. The average popularity score by of this CVE is 89.87 compared with 27.6 of the others. The total retweet volume predicted by the Hawkes process model is 19.7 times the average value of all the other CVEs. FEEU gives high probability (> 0.9) that this CVE will be exploited in three months. FRET predicts the CVE will be exploited in 117.32 days (compared with the ground truth of 114 days).

Table 4: Real-world exploit prediction, p-value < 0.001

Method	1 month			3 months			6 months			9 months			12 months		
	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall
FEEU-RF	0.435	0.619	0.373	0.487	0.640	0.432	0.432	**0.656**	0.352	0.332	**0.622**	0.249	0.276	0.580	0.200
FEEU-XGBoost	**0.553**	0.549	0.671	0.569	0.545	0.660	0.507	0.569	0.511	0.497	0.606	0.472	0.407	0.589	0.345
FEEU-SVM	0.030	0.016	0.431	0.048	0.026	0.518	0.052	0.029	0.518	0.050	0.033	0.344	0.051	0.034	0.421
FEEU-LR	0.065	0.039	0.334	0.054	0.032	0.252	0.053	0.035	0.144	0.056	0.037	0.176	0.055	0.037	0.114
FEEU-NB	0.330	0.281	**0.655**	0.343	0.254	**0.689**	0.346	0.252	**0.673**	0.355	0.256	**0.663**	0.365	0.267	**0.608**
FEEU-LateFusion	0.326	0.277	**0.655**	**0.593**	0.558	0.682	**0.523**	0.564	0.491	**0.514**	0.577	0.439	**0.488**	**0.590**	0.320
best of baselines (SMOTE)	0.477	**0.664**	0.413	0.452	0.671	0.392	0.349	0.616	0.268	0.260	0.590	0.186	0.158	0.529	0.103
best of baselines	0.290	0.587	0.207	0.283	**0.691**	0.193	0.203	0.599	0.123	0.113	0.501	0.064	0.031	0.345	0.016

5 CONCLUSION AND FUTURE WORK

In this paper, we show that existing studies are flawed in the sense that they utilize information from the future (CVSS scores) in predicting if a CVE will be exploited. To address this issue, we make the *first attempt to predict when a vulnerability will be exploited before NIST assigns CVSS scores to the CVEs, using only Twitter discussion data and the CVE IDs assigned by MITRE*. Furthermore, we are the first to predict not only *if*, but also *when* exploits will take place. To improve prediction, we propose the FEEU and FRET frameworks with two sets of novel features. First, we design a set of meta CVE popularity features which are derived from constructing a family of multi-layer graphs. The popularity features are defined in a mutually reinforcing manner. The calculation of which is solved with the proposed TFIX algorithm with proofs of the convergence and uniqueness properties. Second, we apply the Hawkes process model to estimate the future retweet volume of a CVE as an additional feature. Though no existing studies have addressed the problem of when a CVE will be exploited, the experimental results show that, in terms of F1 score, FEEU outperforms natural adaptations of existing studies by 25.1% on average. Moreover, FRET predicts the day on which a vulnerability will be exploited (from the date of CVE assignment) and is shown to be accurate to 35.71 days on PoC exploits and 11.90 days on real world exploits. Potential future work includes predicting vulnerability exploits with cross-platform dataset (e.g., Reddit, Facebook, Deepweb and Darkweb), and using the prediction results to prioritize vulnerability patching within an optimization framework.

ACKNOWLEDGMENTS

This work is supported by ONR grants N00014-18-1-2670 and N00014-16-1-2896 and ARO grant W911NF-13-1-0421.

REFERENCES

[1] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. 2013. Opinion Fraud Detection in Online Reviews by Network Effects. *International AAAI Conference on Weblogs and Social Media (ICWSM)* 13 (2013), 2–11.

[2] Mohammed Almukaynizi, Eric Nunes, Krishna Dharaiya, Manoj Senguttuvan, Jana Shakarian, and Paulo Shakarian. 2017. Proactive identification of exploits in the wild through vulnerability mentions online. In *International Conference on Cyber Conflict (CyCon US)*. 82–88.

[3] Mehran Bozorgi, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. 2010. Beyond heuristics: learning to classify vulnerabilities and predict exploits. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SigKDD)*. 105–114.

[4] Benjamin Bullough, Anna Yanchenko, Christopher Smith, and Joseph Zipkin. 2017. Predicting exploitation of disclosed software vulnerabilities using open-source data. In *Proceedings of the ACM on International Workshop on Security And Privacy Analytics*. 45–53.

[5] Nitesh Chawla, Kevin Bowyer, Lawrence Hall, and Philip Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research* 16 (2002), 321–357.

[6] The MITRE Corporation. 2018. Common Vulnerabilities and Exposures. (2018). https://cve.mitre.org/cve/.

[7] Manlio De Domenico, Albert Solé-Ribalta, Emanuele Cozzo, Mikko Kivelä, Yamir Moreno, Mason A Porter, Sergio Gómez, and Alex Arenas. 2013. Mathematical formulation of multilayer networks. *Physical Review X* 3, 4 (2013), 041022.

[8] Michel Edkrantz and Alan Said. 2015. Predicting Cyber Vulnerability Exploits with Machine Learning. In *Scandinavian Conference on Artificial Intelligence (SCAI)*. 48–57.

[9] Katheryn A Farris, Ankit Shah, George Cybenko, Rajesh Ganesan, and Sushil Jajodia. 2018. VULCON: A System for Vulnerability Prioritization, Mitigation, and Management. *ACM Transactions on Privacy and Security (TOPS)* 21, 4 (2018), 16.

[10] Georg Frobenius, Ferdinand Georg Frobenius, Ferdinand Georg Frobenius, Ferdinand Georg Frobenius, and Germany Mathematician. 1912. Über Matrizen aus nicht negativen Elementen. (1912).

[11] Alan G Hawkes. 1971. Spectra of some self-exciting and mutually exciting point processes. *Biometrika* 58, 1 (1971), 83–90.

[12] Zeroday Initialtive. 2018. Rewarding Security Researchers for Privately Disclosing Vulnerabilities. (2018). http://www.zerodayinitiative.com/.

[13] Jerichoattrition. 2016. Open sourced vulnerability database (OSVDB). (2016). https://blog.osvdb.org/.

[14] Chanhyun Kang, Noseong Park, B Aditya Prakash, Edoardo Serra, and VS Subrahmanian. 2016. Ensemble models for data-driven prediction of malware infections. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM)*. 583–592.

[15] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P Gleeson, Yamir Moreno, and Mason A Porter. 2014. Multilayer networks. *Journal of complex networks* 2, 3 (2014), 203–271.

[16] Jon M Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)* 46, 5 (1999), 604–632.

[17] Heeyoung Kwon, Mirza Basim Baig, and Leman Akoglu. 2017. A Domain-Agnostic Approach to Spam-URL Detection via Redirects. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. 220–232.

[18] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning (ICML)*. 1188–1196.

[19] NIST. 2018. Vulnerability Metrics. (2018). https://nvd.nist.gov/.

[20] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

[21] Marian-Andrei Rizoiu, Lexing Xie, Scott Sanner, Manuel Cebrian, Honglin Yu, and Pascal Van Hentenryck. 2017. Expecting to be HIP: Hawkes intensity processes for social media popularity. In *Proceedings of the International Conference on World Wide Web (WWW)*. 735–744.

[22] Carl Sabottke, Octavian Suciu, and Tudor Dumitras. 2015. Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting Real-World Exploits.. In *USENIX Security Symposium*. 1041–1056.

[23] Offensive Security. 2018. Offensive Security's Exploit Database Archive. (2018). https://exploit-db.com/.

[24] Symantec. 2018. Symantec A-Z Listing of Threats & Risks. (2018). https://www.symantec.com/security-center/a-z.

[25] Symantec. 2018. Symantec Attack Signatures. (2018). https://www.symantec.com/security_response/attacksignatures/.

[26] Nazgol Tavabi, Palash Goyal, Mohammed Almukaynizi, Paulo Shakarian, and Kristina Lerman. 2018. Darkembed: Exploit prediction with neural language models. In *AAAI Conference on Innovative Applications of Artificial Intelligence (IAAI)*.

[27] Qingyuan Zhao, Murat A Erdogdu, Hera Y He, Anand Rajaraman, and Jure Leskovec. 2015. Seismic: A self-exciting point process model for predicting tweet popularity. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SigKDD)*. 1513–1522.

Table 5: Weights of FRET base regressors in late fusion

Tasks	Linear Regression	Bayesian Regression	Randomforest Regression	XGBoost Regression	Lasso Regression	Ridge Regression
PoC	0	0.8	0.2	0	0	0
real world	0	0.67	0	0.33	0	0

A CONFIGURATIONS

Hardware: The server we host our project is equipped with 24-core Intel E5-2650 CPU and 64GB RAM.

Software: The project is written in Python with version 3.7.0. Specifically, we use pandas 0.23.4 and nltk 3.4 for data processing, numpy 1.16.0, and numba 0.42.0 for scientific computation, scikit-learn 0.20.2 and xgboost 0.81 for implementation of basic classifiers and regressors, and imbalanced-learn 0.4.3 for sampling techniques.

B ROLLING PREDICTION SETTING

For each month, we use all the prior Twitter discussion data to make the prediction. For our proposed classifier FEEU, we predict whether the newly published CVEs are going to be exploited in the next 1, 3, 6, 9, 12 months. We start the rolling prediction from 01/01/2017, thus 17 months are involved in the 1-month prediction task, 15 months are involved in the 3-month prediction task respectively, etc. For our proposed regressor FRET, we predict when the newly published CVEs are going to be exploited. Similarly, in the regression task, we start the rolling regression from 01/01/2017 to 01/01/2018 with 1-month window.

Model parameters: Table 6 shows the parameters for the rolling window predictions and the rolling regressions.

Table 6: Model parameters

Models	Parameters
Randomforest Classifier	# trees=100
XGBoost Classifier	# trees=100
SVM	kernel=linear, C=100
Logisitic Regression	regularization=L2
Naive Bayes	kind=Bernoulli
Randomforest Regression	# trees=100
XGBoost Regression	# trees=100

Late fusion: The late fusion model in FEEU classification task take advantage of five classifiers, namely, Randomforest (RF), XGBoost, SVM, Logisitic Regression (LR), and Naive Bayes (NB). In FRET regression task, the late fusion model uses six regression models, including Linear Regression, Bayesian Regression, Randomforest Regression, XGBoost Regression, Lasso Regression, and Ridge Regression. In both tasks, the late fusion models learn weights from the data prior to 01/01/2017. Specifically, we train the basic models with 80% of the data and optimize the objective by searching the weights of the models. Tables 5 and 7 show the weights of the late fusion models.

Table 7: Weights of FEEU base classifiers in late fusion

Tasks	RF	XGboost	SVM	LR	NB
PoC 1m	0.0	0.8	0.0	0.0	0.2
PoC 3m	0.36	0.18	0.18	0.0	0.27
PoC 6m	0.45	0.18	0.0	0.0	0.36
PoC 9m	0.44	0.0	0.22	0.11	0.22
PoC 12m	0.5	0.0	0.0	0.0	0.5
real world 1m	0.0	0.0	0.0	0.0	1.0
real world 3m	0.0	0.5	0.25	0.0	0.25
real world 6m	0.4	0.2	0.0	0.0	0.4
real world 9m	0.0	0.5	0.0	0.0	0.5
real world 12m	0.18	0.36	0.0	0.0	0.45