Cyber-guided Deep Neural Network for Malicious Repository Detection in GitHub

Yiming Zhang[1], Yujie Fan[1], Shifu Hou[1], Yanfang Ye*[1], Xusheng Xiao[1], Pan Li[1],
Chuan Shi[2], Liang Zhao[3], Shouhuai Xu[4]
[1]Department of Computer and Data Sciences, Case Western Reserve University, OH, USA
[2]School of CS, Beijing University of Posts and Telecommunications, Beijing, China
[3]Department of Information Science and Technology, George Mason University, VA, USA
[4]Department of Computer Science, University of Texas at San Antonio, TX, USA
[1]yanfang.ye@case.edu, [2]shichuan@bupt.edu.cn, [3]lzhao9@gmu.edu, [4]shxu@cs.utsa.edu

***Abstract*—As the largest source code repository, GitHub has played a vital role in modern social coding ecosystem to generate production software. Despite the apparent benefits of such social coding paradigm, its potential security risks have been largely overlooked (e.g., malicious codes or repositories could be easily embedded and distributed). To address this imminent issue, in this paper, we propose a novel framework (named *GitCyber*) to automate malicious repository detection in GitHub at the first attempt. In *GitCyber*, we first extract code contents from the repositories hosted in GitHub as the inputs for deep neural network (DNN), and then we incorporate cybersecurity domain knowledge modeled by heterogeneous information network (HIN) to design cyber-guided loss function in the learning objective of the DNN to assure the classification performance while preserving consistency with the observational domain knowledge. Comprehensive experiments based on the large-scale data collected from GitHub demonstrate that our proposed *GitCyber* outperforms the state-of-the-arts in malicious repository detection.**

***Keywords*-Cyber-guided DNN; heterogeneous information network; malicious repository detection;**

I. Introduction

With the broad scale proliferation of connected devices and systems expected to reach billions by 2025 [1], software has played a vital role in the increasingly connected cyberspace that permeates people's everyday lives. In recent years, the number of software has increased exponentially, whose market has grown into a multi-billion dollars industry [2]. Unlike conventional software development process where developers significantly rely on code handbooks to create software from scratch, more and more software products are now created with the support from a highly interoperable and collaborative social coding platforms such as GitHub, which is the largest source code repository hosting more than 100 million software projects maintained by over 40 million registered developers [3]. Within the modern software programming ecosystem, developers can reuse libraries or adapt existing ready-to-use projects to expedite software development. However, the popularity and openness of such social coding environment not only attract developers to contribute legitimate software but also attackers to disseminate malicious codes [4]. For example, as shown in Figure 1, malicious repositories that are intentionally hosted in GitHub by attackers could be directly forked by other developers; on the other hand, recent studies [5]–[7] have shown that the interplay between GitHub and other social media platforms is more active than we had thought (i.e., malicious repositories hosted in GitHub could easily disseminated through online programming discussion platforms such as Stack Overflow or social media platforms such as Reddit to generate the production software).

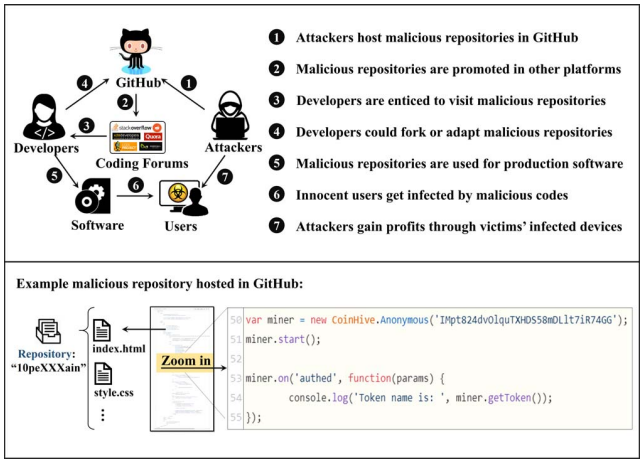

Figure 1: Attacks performed by using malicious repositories.

To maintain a productive yet secure ecosystem against malicious attacks, GitHub provides a security bug bounty site for code vulnerability reporting [8] as well as two products CodeQL and LGTM for semantic code analysis [9]. Nevertheless, such security policy is limited and merely code content-based analysis may not be sufficient to address those social coding security-related concerns. To enhance the security of modern software programming ecosystem against malicious attacks, there is apparent and urgent need to develop novel methodologies that can automate malicious repository detection in GitHub. Although there have been several studies on social coding platforms, such as insecure/toxic code snippet detection in Stack Overflow [10],

978-1-7281-8156-1/20/$31.00 ©2020 IEEE
DOI 10.1109/ICBK50248.2020.00071

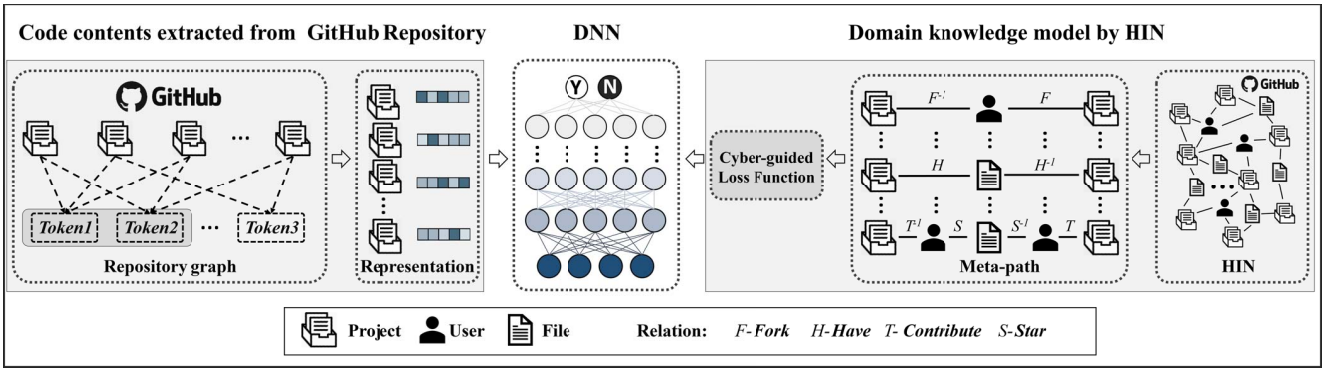

Figure 2: System overview of *GitCyber*.

user behavior/influence analysis in GitHub [11], interplay across platforms [7], to the best of our knowledge, *the topic of automatic detection of malicious repositories in GitHub has not yet been studied in the open literature so far*.

To address the imminent issue above, in this paper, we propose a novel framework to automate malicious repository detection in GitHub *at the first attempt*. An innovative insight brought by this work is to empower deep neural network (DNN) with observational cyber-guided knowledge modeled by structural heterogeneous information network (HIN). To this end, we first extract content-based features from the code repositories hosted in GitHub as the inputs for DNN. Although DNN based models have achieved tremendous success in various applications [12]–[14], one of the concerns regarding such black-box learning frameworks is the lack of interpretability of its classifications with respect to the known observational domain knowledge [15]. As the moral says "man is known by the company he keeps", in addition to code contents, a repository's legitimacy may be judged by the social information that it associates with in the modern coding platform [4]. How can we represent such knowledge leveraging the social coding properties (e.g., a malicious repository could be always associated with other malicious ones, and vice versa) and incorporate it into the DNN based learning framework? To answer this question, we introduce a structural HIN and present meta-path based approach to model neighborhood relations among code repositories hosted in GitHub; then, we incorporate the knowledge encoded by repositories' neighborhood relations to design cyber-guided loss function in the learning objective of the DNN to assure the classification performance while preserving consistency with the observational domain knowledge. We develop a system named *GitCyber* (shown in Figure 2) integrating our proposed method for malicious repository detection in GitHub. The major contributions of this work are summarized below:

- We proposes a novel cyber-guided DNN (i.e., CyberDNN) with the design of cyber-guided loss function in the learning objective, to ensure the classification performance of DNN while retaining consistency with the observational domain knowledge.
- In addition to code contents, we elegantly represent the domain knowledge by leveraging the social coding properties. Neighborhood relations among code repositories in GitHub are decoded by structural HIN and meta-path based method. The proposed solution provides a natural yet innovative way for cyber-guided knowledge representation.
- Comprehensive experiments based on large-scale data collected from GitHub demonstrate the performance of our developed system *GitCyber* in malicious repository detection, by comparisons with the state-of-the-arts and popular commercial security products. The source codes and benchmark datasets will be made publicly available after the review.

II. Proposed Method

In this section, we introduce our proposed method of cyber-guided DNN for malicious repository detection in GitHub in detail.

A. Content-based Repository Representation

A GitHub repository's legitimacy largely depends on the code contents which provide critical information about its functionality and intention. In order to represent a repository by using the content information, we first extract all the tokens from each of its source code files. These tokens are the high-level specifications of code behaviors, which can reflect the intent and goal as they often contain the important semantic information. For example, if the token of *"coin-hive"* occurs in a source code file, the intent of its associated repository could be related to cryptocurrency mining. We further select a subset tokens, defined as malicious-oriented keywords, by (1) calculating its repository frequency, and (2) measuring the difference of its distribution between malicious and benign repositories:

$$\|\mathbb{1}(\forall r \ni t)\|_1 > \varepsilon \text{ and } \frac{\|\mathbb{1}(\forall r^+ \ni t)\|_1}{\|\mathbb{1}(\forall r^- \ni t)\|_1} > \delta, \qquad (1)$$

where t means a token, the indicator function $\mathbb{1}$ takes a value of 1 if the judgment is true and 0 otherwise, $r = r^+ \cup r^-$, r^+ (r^-) denotes malicious (benign) repository. By applying Eq. (1), we filter out tokens and obtain the related malicious-oriented keywords (i.e., in this work, based on our data collection described in Section III-A, we obtain 2,900 malicious-oriented keywords). The top ten malicious-oriented keywords are *"didoptOut, CoinHive, miner, authedmine, anonymous, threads, credentials, coin, hive, wss"*. To comprehensively describe any given repository, instead of directly representing it as a feature vector of these extracted keywords, we build a knowledge graph where nodes represent repositories and keywords, edges among nodes denote whether a keyword occurs in a repository. The constructed knowledge graph is able to capture the global co-occurrences between repositories and keywords explicitly. To reduce high computational cost for graph mining, we exploit graph embedding technique DeepWalk [16] for node representation learning, which consists of random walk generation and skip-gram model. The learned low-dimensional repository representation can be directly feed to a DNN framework for malicious repositories detection.

B. Cyber-guided Deep Neural Network

To detect the malicious repositories, besides content features directly extracted from source files, the relationships among different repositories (e.g., two repositories are hosted by the same user in GitHub) could provide important information to determine the legitimacy of the repositories [17]–[19]. More specifically, a repository's legitimacy can be inferred by its neighbors in the social coding platform: a repository with more malicious neighbors holds a higher possibility of being malicious, and vice versa. As discussed above, the prediction by a DNN model using content-based features only may lack consistency or interpretability with respect to this observational knowledge. How can we represent such knowledge and further incorporate it into a DNN learning framework? To solve this problem, in this work, we propose a novel framework (named CyberDNN), in which we model the domain knowledge by using a structural HIN and meta-path based approach and then we formulate the represented domain knowledge into a cyber-guided loss function in the learning objective for the DNN classifier.

Domain Knowledge Modeled by HIN. The interoperable and collaborative properties of Github enlighten us to leverage the following five social coding relationships to assist with the representation of the observational knowledge:

- ***R1***: *user-fork-repository* relation denotes a user either proposes changes to other developer's repository or uses the repository as a starting point for further development;
- ***R2***: *user-comment-repository* relation describes whether a user posts a comment, question, or prop on a pull request's conversation tab;

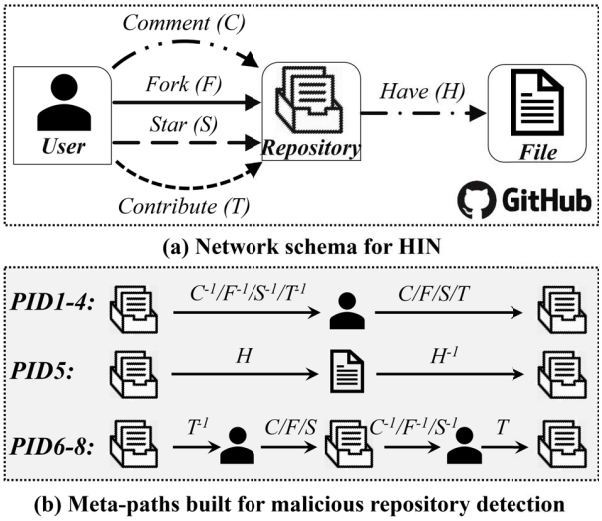

Figure 3: Network schema and meta-paths for HIN

- ***R3***: *user-star-repository* relation means a user stars a repository, denoting his/her interest to keep track of the repository;
- ***R4***: *user-contribute-repository* relation describes if a user contributes to a repository;
- ***R5***: *repository-have-file* relation denotes if a repository includes a source code file.

The multi-typed entities (i.e., user, repository, file) and relations (i.e., *R1-R5*) can be concisely modeled by a HIN. A ***heterogeneous information network (HIN)*** [20] is defined as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with an entity type mapping ϕ: $\mathcal{V} \rightarrow \mathcal{A}$ and a relation type mapping ψ: $\mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{V}$ denotes the entity set and $\mathcal{E}$ is the relation set, $\mathcal{A}$ denotes the entity type set and $\mathcal{R}$ is the relation type set, and the number of entity types $|\mathcal{A}| > 1$ or the number of relation types $|\mathcal{R}| > 1$. The ***network schema*** [20] for network $\mathcal{G}$, denoted as $\mathcal{T}_\mathcal{G} = (\mathcal{A}, \mathcal{R})$, is a graph with nodes as entity types from $\mathcal{A}$ and edges as relation types from $\mathcal{R}$. Based on the definitions, the network schema designed for our application is shown in Figure 3.(a).

In order to precisely and concretely define the neighbors of a repository in the social coding platform, we propose to utilize the concept of meta-path built on HIN to characterize such neighborhood relation. That is, if two repositories can be connected via a meta-path, they will be regarded as each other's neighbor. Formally, a ***meta-path*** [20] $\mathcal{P}$ is a path defined on the network schema $\mathcal{T}_\mathcal{G} = (\mathcal{A}, \mathcal{R})$, and is denoted in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \ldots \xrightarrow{R_L} A_{L+1}$, which defines a composite relation $R = R_1 \cdot R_2 \cdot \ldots \cdot R_L$ between types A_1 and A_{L+1}, where $\cdot$ denotes relation composition operator, and L is length of $\mathcal{P}$. Based on the network schema shown in Figure 3.(a), we design eight meaningful meta-paths (i.e., *PID1-PID8* shown in Figure

3.(b)), which define the neighborhood relationships from different views. For example, *PID1*: $repository \xrightarrow{fork^{-1}} user \xrightarrow{fork} repository$ denotes a neighborhood relation of two repositories if they are forked by the same user. We use a real-world example discovered in GitHub for further illustration: the user "Don****" in GitHub forks several malicious cryptocurrency mining repositories; by examining the repositories hosted by this user, we find that he/she may be a mail system developer who embeds codes from his/her forked repositories to his/her own developed software (i.e., injecting cryptocurrency mining service to mine Monero cryptocurrency). Another meth-path *PID5*: $repository \xrightarrow{have} file \xrightarrow{have^{-1}} repository$ depicts that two repositories are regarded as neighbors if they both include the same source file (e.g., a third-party library). Based on the large-scale data collected from GitHub, we annotate a benchmark dataset which includes 3,729 repositories (i.e., 1,492 are malicious, 2,237 are benign) with 53,648 source files related to 3,303 users. The benchmark dataset is annotated by anti-malware experts leveraging the results from VirusTotal [21] which consists of more than 70 anti-malware scanning tools. Resting on this dataset, as shown in Figure 4, guided by the eight designed meta-paths (i.e., *PID1-PID8*), we observe that the more malicious repositories the node (i.e., repository) neighbors the higher probability the node is classified as malicious, and vice versa. This observation further demonstrates the rationale and validity of using meta-paths built on HIN to define a repository's neighborhood relations.

By using meta-path to decode the neighborhood relations among code repositories in GitHub, given two repositories i and j, we formally define the observational domain knowledge, termed cyber-guided principle, as following:

$$\big(\Pr(i) - \Pr(j)\big) \cdot \big(f(p_i^+, p_i^-) - f(p_j^+, p_j^-)\big) > 0, \quad (2)$$

where $\Pr$ is the probability of a repository being predicted as malicious by a DNN model, p_i^+ (p_i^-) denotes the probability of repository i's malicious (benign) neighbors guided by a specific meta-path, $f(.)$ measures the difference between these two probabilities (i.e., p_i^+ and p_i^-).

Cyber-guided Loss Function. To preserve the consistency of the represented domain knowledge, for any given pair of repositories, i and j, if their predicted probabilities from the DNN using content-based representations dissatisfy the cyber-guided principle (Eq. (2)), a violation of this principle should be considered as a regularizer to be added to the learning objective. We regard this regularizer as cyber-guided loss, which is formulated as:

$$\mathcal{L}_{cyb} = \sum_{m=1}^{M} f(p_{i,m}^+, p_{i,m}^-) - f(p_{j,m}^+, p_{j,m}^-), \quad (3)$$

where M is the number of designed meta-path, $p_{i,m}^+$ ($p_{i,m}^-$) denotes repository i's malicious (benign) neighbors under the m-th meta-path. Then, given the dataset D to be the form of $D = \{\mathbf{x}_i, y_i\}_{i=1}^n$ of n repositories, where $\mathbf{x}_i \in \mathbb{R}^d$ is the representation of repository i learned from Section II-A, y_i is its class label ($y_i \in \{+1, -1\}$, +1: malicious, -1 benign), by incorporating the cyber-guided loss (Eq. (3)), the learning objective for our proposed CyberDNN is formulated as:

$$\arg\min_{\boldsymbol{\theta}} \mathcal{L} + \lambda\Omega(\boldsymbol{\theta}) + \lambda_{cyb}\mathcal{L}_{cyb}, \quad (4)$$

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \quad (5)$$

$$\hat{y}_i = DNN_{\boldsymbol{\theta}}(\mathbf{x_i}), \quad (6)$$

$$\Omega(\boldsymbol{\theta}) = \|\mathbf{W}\|_2, \quad (7)$$

where $\mathcal{L}$ is the empirical loss of the DNN model (i.e., mean squared error in our case), $\hat{y}$ is the predicted label from the DNN, $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{b}\}$ represents the set of weight and bias parameters across all hidden and output layers, $\Omega(\boldsymbol{\theta})$ is the L_2 regularization, $\mathcal{L}_{cyb}$ is the designed cyber-guided loss, λ and λ_{cyb} are the hyper-parameters determining the importance of regularization and cyber-guided loss.

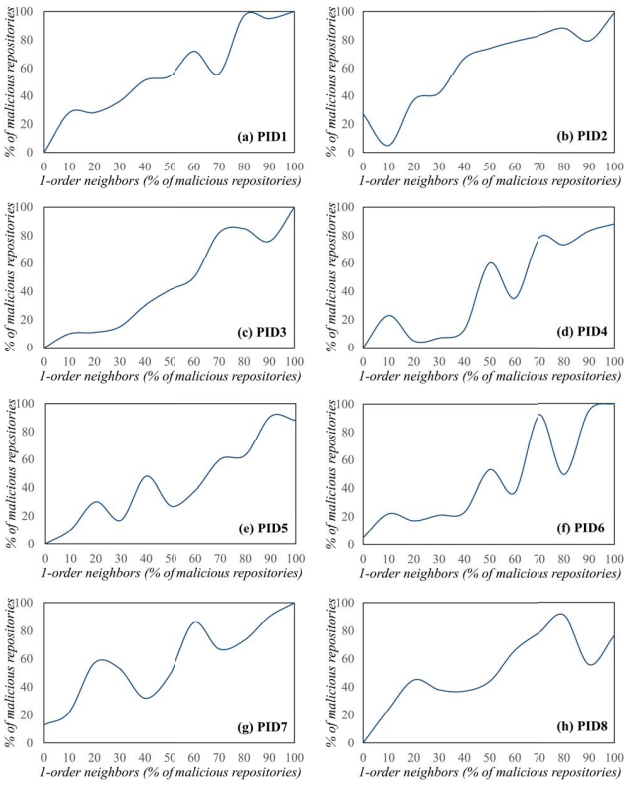

Figure 4: 1-order *repository-repository* neighborhood relations under different meta-path schemes (i.e., *PID1-PID8*).

III. Experimental Results and Analysis

In this section, we fully evaluate the performance of our proposed method for malicious repository detection in GitHub, by comparison with baseline methods and other popular commercial anti-malware products.

A. Experimental Setup

Data Collection and Annotation. With the population of cryptocurrency [22], in this work, we aim to investigate those repositories relevant to blockchain and cryptocurrency hosted in GitHub. Based on a set of designed cryptocurrency related keywords (e.g., Bitcoin, coin, mine, etc.), we develop a crawling tool to collect the open source repositories containing these keywords as well as the corresponding users' profiles from GitHub through Oct. 15, 2019 till Nov. 31, 2019: 4,178 GitHub repositories hosted by 3,689 users have been collected and preprocessed. In order to obtain the ground truth, we apply a two-step mechanism: (1) The collected repositories are first uploaded to VirusTotal [21] consisting of more than 70 anti-malware scanning tools to validate their legitimacy. Since there is a limit of file size in VirusTotal (i.e., 200 MB), we manually remove 449 oversize repositories. (2) Based on the scanned results from VirusTotal, we then ask anti-malware experts for further analysis to obtain the final annotated dataset containing 3,729 repositories (i.e., 1,492 are malicious, 2,237 are benign) with 53,648 source files related to 3,303 users. Based on the extracted features and designed network schema, the constructed HIN has 60,677 nodes (i.e., 3,726 nodes with type of repository, 3,303 nodes with type of GitHub user, 53,648 nodes with type of source file) and 81,097 edges including relations of *R1-R5*. To quantitatively validate the performance, we use the measures shown in Table I.

Table I: Performance indices of malicious repository detection.

Indices	Description
TP	# of repositories correctly classified as malicious
TN	# of repositories correctly classified as benign
FP	# of repositories mistakenly classified as malicious
FN	# of repositories mistakenly classified as benign
ACC	$(TP+TN)/(TP+TN+FP+FN)$
F1	$2*Precision*Recall/(Precision+Recall)$
TPR	$TP/(TP+FN)$
FPR	$FP/(FP+TN)$

Baseline Methods. We first evaluate the performance of our proposed *GitCyber* for malicious repository detection by comparison with the following baseline methods:

- **BoW-DNN:** This method first represents each repository as a bag-of-words feature vector based on the tokens extracted using Eq. (1), and then feeds the feature vector into a generic 5-layer DNN.
- **BoW-SVM:** This method replaces DNN in BoW-DNN with Support Vector Machine (SVM) as the detection module.
- **Git-DNN:** Instead of directly using bag-of-words as the feature vectors for repositories, this model applies the method introduced in Section II-A to learn the low-dimensional repository representations as the inputs fed to the DNN;
- **Git-SVM:** Similar to the setting of Git-DNN, it replaces the DNN model with SVM;
- **M2V-DNN:** We enhance the HIN described in Section II-B by incorporating the repository's content information, i.e., adding extracted keywords as entities and then applying metapath2vec [23] for repository representation learning. The learned representations are then fed to the DNN for training and prediction.
- **M2V-SVM:** This method replaces DNN in M2V-DNN by SVM as the classifier.

Commercial Security Products. We further validate the performance of our developed *GitCyber* by comparisons with other security products such as LGTM (a semantic code security analysis tools provide by GitHub) and over 70 popular anti-malware products which are integrated in VirusTotal [21].

Hyper-parameters. The experimental studies are conducted under the environment of Ubuntu 16.04 operating system, plus Intel i9-9900k CPU, GeForce GTX 2080 Ti Graphics Cards and 64 GB of RAM. All DNN models are implemented using the Keras package [24] with a batch size of 1000 and a maximum number of epochs of 100,000. The value of λ is set to 1 in all experiments. Other parameters include the dimension of node embedding $d = 128$, neighborhood size $w = 5$, iteration time $epoch = 5$ for skip-gram model. For SVM, we apply sklearn.svm with RBF kernel in our experiments and the penalty is empirically set to 16 while other parameters are set by default. To facilitate the comparisons, we use 10-fold cross validations.

Table II: Comparisons with different baseline methods

Method	ACC	F1	TPR	FPR
BoW-DNN	0.8172	0.7721	0.7740	0.1507
BoW-SVM	0.8305	0.7810	0.7553	0.1632
Git-DNN	0.8546	0.8143	0.7970	0.1354
Git-SVM	0.8649	0.8267	0.8056	0.1296
M2V-DNN	0.8842	0.8495	0.8170	0.1221
M2V-SVM	0.8945	0.8704	0.8263	0.1158
GitCyber	**0.9146**	**0.8893**	**0.8663**	**0.0892**

B. Comparisons and Analysis

Based on the above dataset, we first show the performances of *GitCyber* and all the baseline methods introduced above in malicious repository detection. The experimental results are illustrated in Table II. From the results, we observe that:

- BoW-based method (i.e., BoW-DNN and BoW-SVM) using the traditional bag-of-words as feature vectors obtains the worst detection performance, which shows such representation fails to depict the higher-level semantics in malicious repository detection.
- M2V-DNN and M2V-SVM which incorporate the repository's content information with domain knowledge (formulated as the neighborhood relations among repositories) achieve better outcomes than the models merely considering content information, such as Git-DNN and Git-SVM. This demonstrates that the domain knowledge helps the performance of malicious repository detection.
- Our proposed system *GitCyber* which applies cyber-guided loss to regularize the learning objective of traditional DNN model significantly outperforms all baseline methods. This shows the superiority of using the observational domain knowledge in the learning objective for improving generalization performance.

Table III: Comparisons with other commercial security products.

Method	Version	ACC	F1	TPR	FPR
LGTM	-	0.6799	0.5696	0.6498	0.6199
Antiy-AVL	3.0.0.1	0.8971	0.8526	0.7431	0.1717
Avast	18.4.3895.0	0.9073	0.8705	0.7779	0.1484
AVG	18.4.3895.0	0.9129	0.8796	0.7940	0.1377
Avira	8.3.3.8	0.8671	0.8021	0.6722	0.2190
BitDefender	7.200	0.9035	0.8631	0.7592	0.1609
ClamAV	0.102.0.0	0.8521	0.7780	0.6468	0.2360
Comodo	31649.000	0.8899	0.8590	0.8375	0.1086
Cyren	6.2.2.2	0.9003	0.8584	0.7545	0.1641
DrWeb,	7.0.41.7240	0.9137	0.8819	0.8040	0.1310
Emsisoft	2018.12.0.1641	0.9014	0.8596	0.7538	0.1645
FireEye	29.7.0.0	0.9006	0.8583	0.7518	0.1658
Fortinet	5.4.247.0	0.8917	0.8441	0.7318	0.1793
F-Prot	4.7.1.166	0.8599	0.7890	0.6542	0.2311
F-Secure	12.0.86.52	0.8617	0.7928	0.6602	0.2271
GData	25.23778	0.9073	0.8714	0.7839	0.1444
Ikarus	0.1.5.2	0.9076	0.8712	0.7806	0.1466
Jiangmin	16.0.100	0.7752	0.6698	0.5692	0.2879
Kaspersky	15.0.1.13	0.8939	0.8506	0.7538	0.1645
MAX	2019.9.16.1	0.9033	0.8627	0.7585	0.1614
Mcafee	6.0.6.653	0.8695	0.8057	0.6756	0.2168
Microsoft	1.1.16500.1	0.8711	0.8413	0.8528	0.0983
NANO-AV	1.0.134.24859	0.9070	0.8734	0.8007	0.1332
Rising	25.0.0.24	0.9057	0.8668	0.7659	0.1565
Sophos	4.98.0	0.8438	0.7578	0.6100	0.2606
Zillya	2.0.0.3933	0.7998	0.6673	0.5010	0.3335
GitCyber	-	**0.9146**	**0.8893**	**0.8663**	**0.0892**

We then evaluate the detection performance of our developed *GitCyber* in comparisons with twenty-five popular commercial anti-malware products. From Table III, we can see that *GitCyber* yields the highest accuracy, F1-measure , True positive rate (TPR) and False positive rate (FPR) in the detection of cryptocurrency-related malicious repository. To put this into perspective, *GitCyber* achieves a 2% accuracy improvement in comparison with Kaspersky and 4% with Mcafee. This again demonstrates that our developed framework *GitCyber* can significantly improve the detection performance in real world data as it not only takes the repository's content into consideration, but also incorporates the domain knowledge into the DNN-based learning framework for regularization.

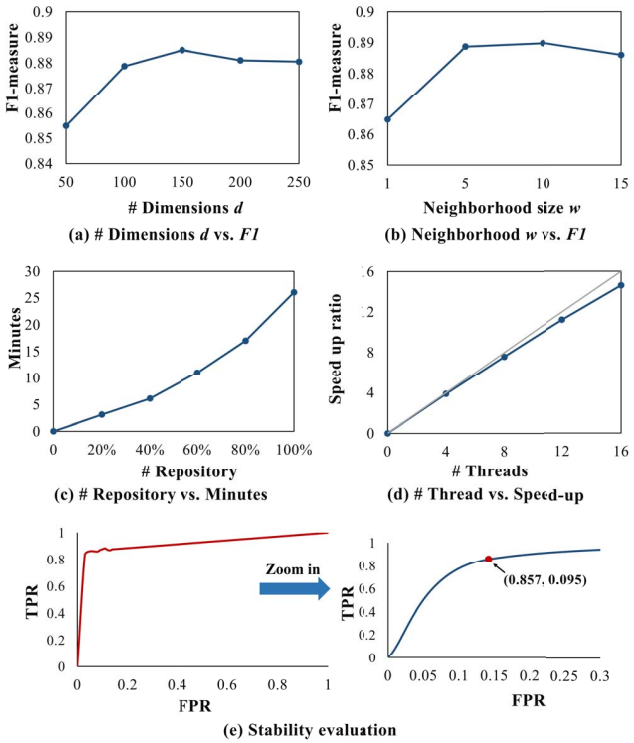

Figure 5: Parameter sensitivity, stability and scalability

C. Parameter Sensitivity, Stability and Scalability

In this set of experiments, we first conduct the sensitivity analysis of how different choices of parameters (i.e., vector dimension d and neighborhood size w) will affect the performance of *GitCyber* in malicious repository detection. From the results shown in Figure 5.(a), we observe that the performance tends to be stable once d reaches around 150; similarly, from Figure 5.(b) we can see that the performance inclines to be stable when w increases to 5. Overall, *GitCyber* is not strictly sensitive to these parameters and is able to reach high performance under a cost-effective parameter choice. We then further evaluate the scalability of *GitCyber*. Figure 5(c) shows the running time of *GitCyber* with different sizes of the dataset, which illustrates that the running time is quadratic to the number of samples. When dealing with more data, approximation or parallel algorithms can be developed. We also run experiments using the default parameters with different number of threads (i.e., 1, 4, 8, 12, 16), each of which utilizes one CPU core. Figure 5(d) shows the speed-up of *GitCyber* deploys multiple threads over the single-threaded case, which shows that the model achieves acceptable sub-linear speed-ups as the line is close to the

optimal line. For stability evaluation, Figure 5(e) shows the overall receiver operating characteristic (ROC) curves of *GitCyber* based on the 10-fold cross validations; it achieves an impressive 0.857 average TP rate at the 0.095 average FP rate for malicious repository detection. From the results and analysis above, *GitCyber* is efficient, scalable and stable for practical use.

IV. Related Work

There have been many works on knowledge discovery from social coding platforms [7], [25]. However, most of these works only focus on code semantics and user behaviors, but rarely address the issue of coding security analysis. The only exceptions appear to be [29] and [10], which both exploit code content-based information to detect code clones in social coding platform. Though those research results are promising, they fail to consider domain knowledge in solving the related problems. Different from existing works, in this paper, we take the domain knowledge (i.e., social coding properties) into consideration for malicious repository detection.

HIN is proposed to model different types of entities and relations and has been applied to various applications, such as scientific publication network analysis [20], [30], health intelligence [31], [32] and cybersecurity [33]–[38]. Several measures (e.g., meta-path [20], [39], meta-structure [40], [41], meta-graph [42], [43]) have already been proposed for relevance computation over HIN entities. Different from the existing works, in this paper, we first model the cybersecurity domain knowledge by using HIN and the concept of meta-path and then we incorporate the represented knowledge as cyber-guided loss to devise the DNN. The proposed cyber-guided DNN framework is able to preserve the consistency of predictions with observational domain knowledge to assure the detection performance.

V. Conclusion

To address the imminent code security issues in social coding platforms, in this paper, we propose a cyber-guided DNN framework, named *GitCyber*, to automate malicious repository detection in GitHub. We bring a new insight to empower deep neural network with observational knowledge. In *GitCyber*, we first learn the content-based representations of the code repositories hosted in GitHub; and then we introduce a structural HIN and the concept of meta-path to model the observational domain knowledge encoded by social coding properties (i.e., a malicious repository is always associated with malicious ones in the social coding platform, and vice versa); finally, we incorporate the represented domain knowledge to design the cyber-guided loss to regularize the learning objective in the DNN model to assure the detection performance. Comprehensive experiments based on the annotated repositories hosted in GitHub demonstrate the performance of our developed system *GitCyber* in malicious repository detection, by comparisons with the baseline methods and the popular commercial security products.

Acknowledgment

Y. Zhang, Y. Fan, S. Hou, Y. Ye's work is partially supported by the NSF under grants of CNS-2034470, IIS-2027127, IIS-1951504, CNS-1940859, CNS-1946327, OAC-1940855 and CNS-1814825, and the NIJ 2018-75-CX-0032.

References

[1] D. R. Carrie MacGillivray, *Worldwide Global DataSphere IoT Device and Data Forecast, 2019–2023*, 2019, https://www.idc.com/getdoc.jsp?containerId=US45066919.

[2] Statista, "Statistics and market data on software," https://www.statista.com/markets/418/topic/484/software/, 2018.

[3] GitHub, "Github: Built for developers," https://github.com/about, 2020.

[4] Y. Ye, S. Hou, L. Chen, X. Li, L. Zhao, S. Xu, J. Wang, and Q. Xiong, "Icsd: An automatic system for insecure code snippet detection in stack overflow over heterogeneous information network," in *ACSAC*. ACM, 2018, pp. 542–552.

[5] B. Vasilescu, V. Filkov, and A. Serebrenik, "Stackoverflow and github: Associations between software development and crowdsourced knowledge," in *ICSC*. IEEE, 2013, pp. 188–195.

[6] G. Silvestri, J. Yang, A. Bozzon, and A. Tagarelli, "Linking accounts across social networks: the case of stackoverflow, github and twitter," in *KDWeb*, 2015, pp. 41–52.

[7] Y. Fan, Y. Zhang, S. Hou, L. Chen, Y. Ye, C. Shi, L. Zhao, and S. Xu, "idev: Enhancing social coding security by cross-platform user identification between github and stack overflow," in *IJCAI*, 2019, pp. 2272–2278.

[8] P. Ducklin, "Github starts scanning millions of projects for insecure components," 2017.

[9] Semmle, *Semmle*, 2019, https://semmle.com/.

[10] C. Ragkhitwetsagul, J. Krinke, M. Paixao, G. Bianco, and R. Oliveto, "Toxic code snippets on stack overflow," *IEEE Transactions on Software Engineering*, 2019.

[11] K. Blincoe, J. Sheoran, S. Goggins, E. Petakovic, and D. Damian, "Understanding the popular users: Following, affiliation influence and leadership on github," *Information and Software Technology*, vol. 70, pp. 30–39, 2016.

[12] T. Alashkar, S. Jiang, S. Wang, and Y. Fu, "Examples-rules guided deep neural network for makeup recommendation," in *AAAI*, 2017.

[13] T.-B. Xu and C.-L. Liu, "Data-distortion guided self-distillation for deep neural networks," in *AAAI*, vol. 33, 2019, pp. 5565–5572.

[14] G. Ning, Z. Zhang, and Z. He, "Knowledge-guided deep fractal neural networks for human pose estimation," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1246–1259, 2017.

[15] A. Karpatne, W. Watkins, J. Read, and V. Kumar, "Physics-guided neural networks (pgnn): An application in lake temperature modeling," *arXiv*, 2017.

[16] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD*. ACM, 2014, pp. 701–710.

[17] Y. Ye, T. Li, S. Zhu, W. Zhuang, E. Tas, U. Gupta, and M. Abdulhayoglu, "Combining file content and file relations for cloud based malware detection," in *KDD*. ACM, 2011.

[18] D. H. P. Chau, C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos, "Polonium: Tera-scale graph mining and inference for malware detection," in *SDM*. SIAM, 2011, pp. 131–142.

[19] A. Venzhega, P. Zhinalieva, and N. Suboch, "Graph-based malware distributors detection," in *WWW*. ACM, 2013, pp. 1141–1144.

[20] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," *VLDB Endowment*, vol. 4, no. 11, pp. 992–1003, 2011.

[21] G. Sood, *virustotal: R Client for the virustotal API*, 2017, r package version 0.2.1.

[22] D. Dasgupta, J. M. Shrein, and K. D. Gupta, "A survey of blockchain from security perspective," *Journal of Banking and Financial Technology*, vol. 3, no. 1, pp. 1–17, 2019.

[23] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *KDD*, 2017.

[24] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[25] Y. Acar, C. Stransky, D. Wermke, M. L. Mazurek, and S. Fahl, "Security developer studies with github users: Exploring a convenience sample," in *UPS*, 2017, pp. 81–95.

[26] S. Horawalavithana, A. Bhattacharjee, R. Liu, N. Choudhury, L. O. Hall, and A. Iamnitchi, "Mentions of security vulnerabilities on reddit, twitter and github," in *ICWI*, 2019, pp. 200–207.

[27] F. Thung, T. F. Bissyande, D. Lo, and L. Jiang, "Network structure of social coding in github," in *CSMR*. IEEE, 2013, pp. 323–326.

[28] D. Pletea, B. Vasilescu, and A. Serebrenik, "Security and emotion: sentiment analysis of security discussions on github," in *MSR*, 2014, pp. 348–351.

[29] M. Gharehyazie, B. Ray, M. Keshani, M. S. Zavosht, A. Heydarnoori, and V. Filkov, "Cross-project code clones in github," *Empirical Software Engineering*, vol. 24, no. 3, pp. 1538–1573, 2019.

[30] Y. Sun, R. Barber, M. Gupta, C. C. Aggarwal, and J. Han, "Co-author relationship prediction in heterogeneous bibliographic networks," in *ASONAM*. IEEE, 2011, pp. 121–128.

[31] Y. Fan, Y. Zhang, Y. Ye, and X. Li, "Automatic opioid user detection from twitter: Transductive ensemble built on different meta-graph based similarities over heterogeneous information network," in *IJCAI*, 2018.

[32] Y. Fan, Y. Zhang, Y. Ye, X. Li, and W. Zheng, "Social media for opioid addiction epidemiology: Automatic detection of opioid addicts from twitter and case studies," in *CIKM*, 2017, pp. 1259–1267.

[33] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, p. 41, 2017.

[34] S. Hou, Y. Ye, Y. Song, and M. Abdulhayoglu, "Hindroid: An intelligent android malware detection system based on structured heterogeneous information network," in *KDD*. ACM, 2017.

[35] Y. Fan, S. Hou, Y. Zhang, Y. Ye, and M. Abdulhayoglu, "Gotcha-sly malware! scorpion a metagraph2vec based malware detection system," in *KDD*, 2018, pp. 253–262.

[36] S. Hou, Y. Fan, Y. Zhang, Y. Ye, J. Lei, W. Wan, J. Wang, Q. Xiong, and F. Shao, "αcyber: Enhancing robustness of android malware detection system against adversarial attacks on heterogeneous graph based model," in *CIKM*, 2019.

[37] Y. Ye, S. Hou, L. Chen, J. Lei, W. Wan, J. Wang, Q. Xiong, and F. Shao, "Out-of-sample node representation learning for heterogeneous graph in real-time android malware detection," in *IJCAI*, 2019.

[38] Y. Zhang, Y. Fan, Y. Ye, C. Shi, and L. Zhao, "Key player identification in underground forums over attributed heterogeneous information network embedding framework," in *CIKM*, 2019.

[39] Y. Zhang, Y. Fan, W. Song, S. Hou, Y. Ye, X. Li, L. Zhao, C. Shi, J. Wang, and Q. Xiong, "Your style your identity: Leveraging writing and photography styles for drug trafficker identification in darknet markets over attributed heterogeneous information network," in *WWW*, 2019, pp. 3448–3454.

[40] Z. Huang, Y. Zheng, R. Cheng, Y. Sun, N. Mamoulis, and X. Li, "Meta structure: Computing relevance in large heterogeneous information networks," in *KDD*, 2016, pp. 1595–1604.

[41] Y. Zhang, Y. Fan, S. Hou, J. Liu, Y. Ye, and T. Bourlai, "idetector: Automate underground forum analysis based on heterogeneous information network," in *ASONAM*. IEEE, 2018, pp. 1071–1078.

[42] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *KDD*, 2017, pp. 635–644.

[43] Y. Fan, Y. Zhang, Y. Ye, and X. Li, "Automatic opioid user detection from twitter: Transductive ensemble built on different meta-graph based similarities over heterogeneous information network." in *IJCAI*, 2018, pp. 3357–3363.