User Profiling and Vulnerability Introduction Prediction in Social Coding Repositories: A Dynamic Graph Embedding Approach

Agrim Sachdeva, Ben Lazarine, Hongyi Zhu, Sagar Samtani
Indiana University, University of Texas at San Antonio

Monday, August 7, 2023

This material is based upon work supported by the National Science Foundation under Grant No. NSF OAC-1917117 (CICI)

Introduction: Software Vulnerabilities Due to Human Error

- Introduction of several vulnerabilities is caused by human error
  - Most software vulnerabilities are mistakes, not malicious attacks [1]
  - Critical-severity vulnerability make it through code review just as easily as low-severity ones [2]
  - Vulnerabilities typically go undetected for 218 weeks (over four years) before being disclosed (Fig. 1)
- Context: Open Source and GitHub
  - Open-source code is vital to the global economy; services and technology from banking to healthcare; > $100 billion impact (Fig. 1)
  - **Many scientific CIs and their users host and share their research source code on Social Coding Repositories (SCR) such as GitHub**
  - Susceptible due to distributed development; evolving risks; zero-day vulnerabilities
- **"Shifting left" is an important mitigation strategy**
  - **Early detection, or the prediction of the introduction of vulnerabilities by users**

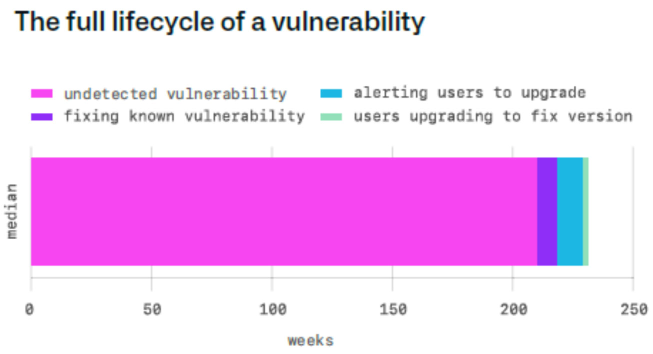


Figure 1. Lifecycle of a vulnerability [3]

Sources: 1: https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html
2: https://blog.gitguardian.com/state-of-secrets-sprawl-2021/
3. State of the Octoverse, GitHub, 2020

Current Solutions

Code-first

- DevSecOps-enabled: Integrated vulnerability scanning or detection systems
- Dependabot, Dependency Review, Dependency Graph, CodeQL and Secret Scanning

Identify vulnerabilities after they have been committed to code
Work at the repository-level, and do not focus on users or provide feedback
Require extensive configuration

User-first

- Developer security awareness training

More effective in reducing vulnerabilities than embedded tools in interfaces (Sedova, 2017)
Quickly loses efficacy if non-targeted
Causes training fatigue

Shift from detecting vulnerabilities in code after they have been introduced to predicting user errors and prevent introduction of vulnerabilities

Example GitHub User

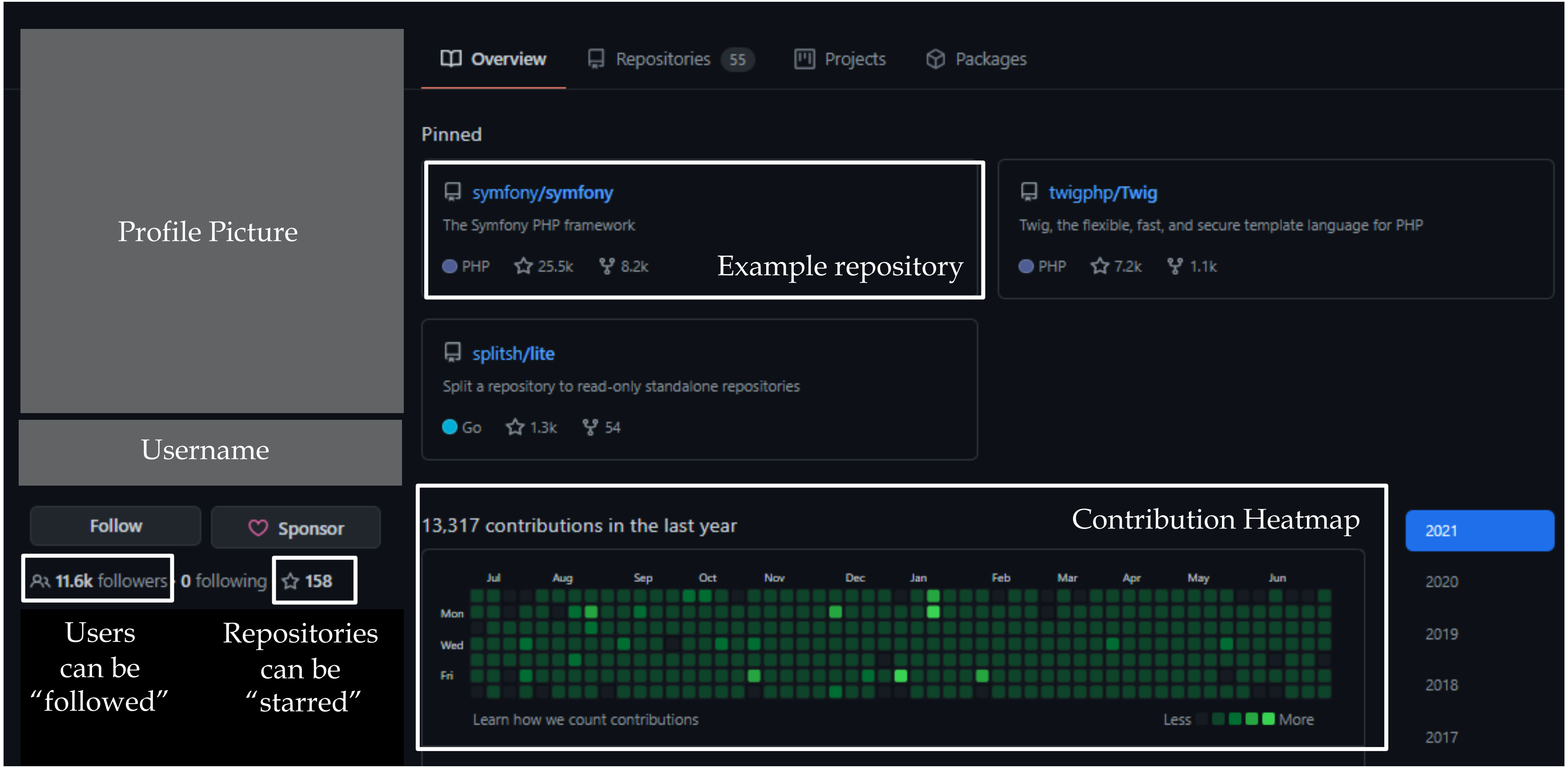


Fig. 2. A GitHub Profile

Vulnerability Introduction Prediction

Vulnerabilities can be introduced and propagated by users

Vulnerability introduction can be impacted by:
- Direct and indirect exposure
- Propagation of knowledge and information between developers

Objective: Predict the introduction of a vulnerability by a user into a repository.

Can enable proactive risk management, e.g., targeted security awareness trainings

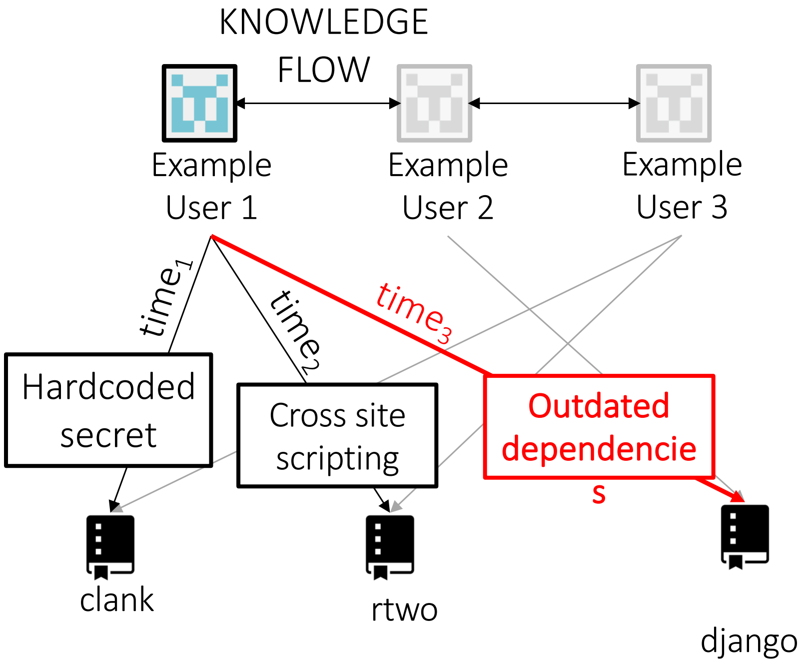


Fig. 3. Example Vulnerability Introduction Prediction within the organization Cyverse

Literature Review: Vulnerability Management for SCR

Year	Author	Platform/ Dataset	Focus	Method	User Profiling	Vulnerability Assessment	Temporal Dynamics
2022	Shahid et al.	Network Metadata	Hybrid CNN + Cookie Analysis	CNN	Yes	No	No
2022	Zerouali et al.	GitHub: npm, RubyGems	Vulnerability dependency networks	Empirical	No	Yes	Yes
2021	Shaji et al.	GitHub: organizations	Non-intrusive vulnerability detection	Probabilistic model	No	Yes	No
2021	Rabheru et al.	GitHub: Wordpress	Novel vulnerability detection	GRU + GCN	No	Yes	No
2021	Mazeura-Rozo et al.	Source Code Representations	Comparing DL to ML models, e.g., Google's AutoML	Empirical	No	Yes	No
2020	Lazarine et al.	GitHub	Cluster attributes and vulnerabilities	TADW	Yes	4 scanners	No
2020	Zhang et al.	GitHub	Malicious blockchain repository detection	HIN	No	3 scanners	No
2019	Gong et al.	GitHub	Detection of malicious online accounts	Phased LSTM	Yes	No	No
2019	Meli et al.	GitHub	Data leakage	Regular Expressions	No	1 scanner	No

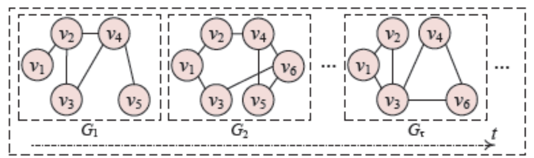
Table 1. Literature Review of studies that use GitHub data for vulnerability analysis or modeling users

Note: CNN: Convolutional Neural Network, TADW: Text Attributed Deep Walk, HIN: Heterogenous Information Network, LSTM: Long short-term memory, GRU: Gated Recurrent Unit, GCN: Graph Convolutional Network

- **Key Observations:**
  1. Most recent studies do not focus on prediction of vulnerabilities, or incorporate a vulnerability assessment on code, overlooking the incidence and nature of vulnerabilities.
  2. Although GitHub as a time-evolving interaction network, studies do not capture the temporal dynamics need to be captured and modelled, reducing the accuracy of the data representation, and therefore, the underlying phenomenon.
  3. The results of vulnerability analysis contains valuable information such as severity levels, which most studies omit.

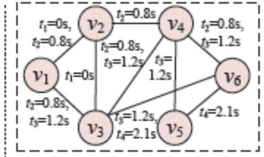
Literature Review: Dynamic Graph Representation Learning

Dynamic Graph Representation Learning



Evenly-Spaced Snapshot Sequence

Graph Summarization
Matrix Factorization
RBM-Based Temporal Models
RNN-Based Temporal Models
Attention-Based Temporal Models



Unevenly-Spaced Snapshot Sequence

Temporal Random Walk
Temporal Point Processes
Time-Encoded Sequential Models
Deep Continuous-Time Processes

Figure 4. Dynamic Graph Representation Learning

GitHub as a time-evolving interaction network wherein individual edges and nodes are inserted or deleted over time in a continuous manner.

Based on the characteristics of our data, i.e., a time-stamped dynamic interaction network, we review dynamic graph representation learning techniques.

By representing users and repositories as nodes, and vulnerabilities as links between the nodes, the task of link prediction would help us predict the introduction of vulnerabilities

Literature Review: Overview of Continuous Propagation and Evolution (CoPE)

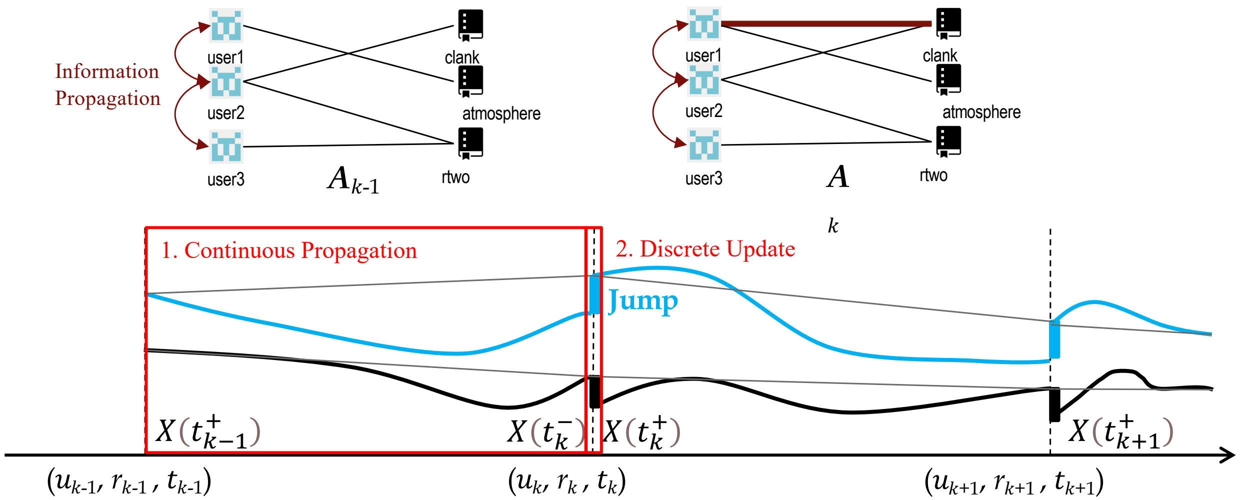


Figure 5. CoPE Forward Pass (Blue, Black), JODIE (Grey)

User embeddings: vulnerability propagation score/ propensity to introduce vulnerabilities
Repository embeddings: Risk level, or vulnerability level

The propagation of information is modeled as non-linear node dynamic evolution between interactions, which captures the knowledge and communication between users that impacts the introduction of a vulnerability.

Literature Review: Dynamic Graph Representation Learning

Sequence	Methods	Algorithm	Contribution	Discrete/ Continuous	Information Propagation	Concurrent Interactions	Authors	Year
Evenly-Spaced Snapshot Sequence	Dynamic Latent Space Models	DSNL	Forward-backward algorithm on the Markov chain over timesteps	Discrete	No	No	Sarkar and Moore	2005
	Incremental SVD	TIMERS	Set the restart time to reduce accumulated error	Discrete	No	No	Zhang et al.	2018
Unevenly-Spaced Snapshot Sequence	Random Walk Based Methods	CTDNE	Temporal random walks that contain a sequence of edges in order	Continuous	No	No	Nguyen et al.	2018
		DNE	Extension for the Skip-gram based network embedding methods	Discrete	No	No	Du et al.	2018
		DynamicTriad	Learn dynamic embeddings by modeling the triadic closure	Discrete	No	No	Zhou et al.	2018
	Graph Neural Network Methods	DyRep	Temporal attention layer to capture the neighbors' interactions	Continuous	No	No	Trivedi et al.	2019
		JODIE	Coupled recurrent neural network model; learns embedding trajectories of two types of nodes (e.g. users and items)	Continuous	No	No	Kumar et al.	2019
		CoPE	Modeling continuous propagation and evolution	Continuous	Yes	Yes	Zhang et al.	2021
		TGAT	Attention layer to efficiently aggregate temporal-topological neighborhood features	Continuous	No	Yes	Xu et al.	2020
		TGN	"Memory" module to update node embeddings	Continuous	No	Yes	Rossi et al.	2020

Table 2. Dynamic Graph Representation Learning Methods; *Note:* DSNL: Dynamic Social Network In Latent Space; TIMERS: Theoretically Instructed Maximum-Error-bounded Restart of SVD; CTDNE: Continuous-Time Dynamic Network Embeddings; GNN: Graph Neural Networks (GNNs); DyRep: Dynamic Representation; CoPE: Continuous Propagation and Evolution

- **Key Observations:**
  1. Prevailing dynamic graph representation Learning for continuous-time interaction graphs are neural-network based methods, which can further be classified into RNN-based (DyREP, JODIE, CoPE), or Attention-based (TGAT, TGN)
  2. CoPE can capture the evolution of nodal embeddings based on the propagation of information, i.e., the impacts of the users on neighboring users and repositories between interactions

Research Questions

Extant security education, training, and awareness (SETA) research does not comment on open-source software security awareness trainings, or the timing of the training to be delivered. The personalization and the timing of delivery are important to security training outcomes.

Baseline link prediction using dynamic graph methods does not consider the severity of the vulnerabilities when they spread, or the relative influence of users.

Based on these research gaps, we pose the following research questions:

1. **How can we predict the introduction of vulnerabilities by users into repositories while accounting for information propagation?**

2. **How can we adapt dynamic graph link prediction methods to incorporate rich feature sets for users, repositories, and vulnerabilities, and capture the relative influence of high-risk repositories and actors?**

Research Question

Shortcomings in existing approaches necessitate the need for **personalized and targeted training**

The prediction of vulnerabilities in source code will allow the creation of risk profiles, and enable proactive and personalized security awareness training.

How can we extend and adapt CoPE to incorporate rich feature sets for users, repositories, and vulnerabilities, and capture the relative influence of high-risk repositories and actors?

Research Design

1. Data Collection

Repository Collection

Vulnerability Scanning

NCAR

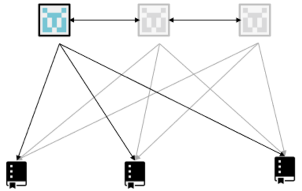
176 users, 77 repos, 3326 vulnerabilities

Flawfinder

2. Graph Formulation

<u>Feature Extraction</u>
User, Repository, Interaction

G = (U, R, E)
E: Vulnerability Introduction
Features: $\boldsymbol{f}(u_k, r_k, t_k)$



3. Proposed SeCoPE Model

Weighted Continuous Propagation Unit

Nodal Embedding Generation

Link Prediction

4. Evaluation

<u>Experiment 1</u>
seCoPE vs. RNN-based Methods

<u>Experiment 2</u>
seCoPE vs. Attention-based Methods

Figure 6. Research Design and Testbed

Research Design: Data Collection

NCAR: Federally funded R&D center for climate science, atmospheric chemistry, solar-terrestrial interactions; founded in 1956; collaborates with 115 universities; **176 users, 77 repositories, 3326 vulnerabilities**

We selected four open-source vulnerability assessment scanners based on language categories of vulnerabilities they scan for, languages, usability and age, i.e., Bandit, Flaw Finder, Gitrob, Trufflehog

Type	Sub-category	Vuln.	Description	Example	Bandit	Flaw Finder	Gitrob	Trufflehog
Secret	Secret	Secret	A potential password/key	739b6afec22ff801a132fc89200a0614953211cd	Yes	No	Yes	Yes
		Password	Word password found	irods://user:pass@host:port/destination	No	No	Yes	Yes
	Cryptography	Weak crypto.	Insufficient crypto. Method	iv_size = crypt.key_size();	Yes	Yes	No	No
	Filetype	Filetype	File that may contain secrets	Django configuration file	No	No	Yes	No
Insecure	Permission	File permission	File may have dang. Permissions	int err_code = chmod(filePath, 0664);	Yes	Yes	No	No
	Insecure method	Insecure function	Function can be vulnerable	Use of insecure and deprecated function (mktemp).	Yes	Yes	No	No
		Insecure module	Module can be vulnerable	Pickle can be unsafe when used to deserialize untrusted data	Yes	No	No	No
		Depricated library	Library no longer supported	The pyCrypto library and its module atfork are no longer actively maintained	Yes	No	No	No
	Internet	Insecure conn.	Dangerous internet connections	Requests call with verify=False disabling SSL certificate checks, security issue.	Yes	No	No	No
Attack	Injection	Insecure input	Dangerous handling of user input	Use of unsafe yaml load. Allows instantiation of arbitrary objects.	Yes	Yes	No	No
		SQL injection	Hardcoded SQL expressions	Possible SQL injection vector through string-based query construction.	Yes	No	No	No
	XSS	XML attack	Dangerous XML library	Using xmlrpclib to parse untrusted XML data is known to be vulnerable.	Yes	No	No	No
		XSS vulnerability	Dangerous library usage	By default, jinja2 sets autoescape to False.	Yes	No	No	No

Table 3. Key Vulnerabilities Returned from Scanners

Research Design: Graph Formulation

- **Interaction Graph**
  - An interaction graph G = (U ∪ R, E) is a user-repository bipartite graph, where each edge $e = (u, r, t) \in$ E represents the introduction of a vulnerability

- **Interactions**
  - Given the user set U and repository set R, sequential interactions between users and repositories can be organized as ordered set E = $\{(u_k, r_k, t_k)\}^{n}_{k=1}$, where $u_k \in$ U, $r_k \in$ R and $0 = t_0 \leq t_1 \leq t_2 \leq \ldots \leq t_n \leq T$
  - Each interaction may be associated with a vector $\boldsymbol{f}(u_k, r_k, t_k)$
  - Time range of sequential interactions can be normalized to [0, 1], thus we have $t_0 = t_1 = 0$ and $t_n = T = 1$.

- **Intuition**
  - We represent vulnerabilities as edges such that the downstream task, i.e., link prediction, can predict the introduction of vulnerabilities
  - The features of the nodes, i.e., users and repositories capture the properties that would influence the introduction of vulnerabilities, such as activity levels, cumulative number of vulnerabilities introduced at time t, programming language, etc. (Lazarine et al. 2020)

Research Design: Graph Formulation

- **Observable Graph**
  - Observable graph at time t is the subgraph with edges, i.e., interactions, happened before time t.
  - The adjacency matrix of the observable graph at time $t \in (t_k, t_{k+1})$ is denoted by $\boldsymbol{A}_k = \begin{bmatrix} 0 & \boldsymbol{B}_k \\ \boldsymbol{B}^{\top}_k & 0 \end{bmatrix}$ where $\boldsymbol{B}_k \in \Re^{|U| \times |R|}$ is the bi-adjacency matrix; element $\boldsymbol{B}_{k,ur}$ denotes the number of interactions between u and r before time t_{k+1}, i.e.,
  $\boldsymbol{B}_{k,ur} = |\{(u', r', t') \in E \,|\, u' = u \wedge r' = r \wedge t' < t_k+1\}|$.

- **Temporal Embedding of Interaction Graph**
  - The goal of temporal embedding is to learn a function $\boldsymbol{x} : (U \cup R) \times [0,T] \to \Re^d$ that reflects the continuous evolution of users and repositories over time.
  - $\boldsymbol{x}(u, t)$: d-dimensional embeddings of user u at time t
  - $\boldsymbol{x}(r, t)$: d-dimensional embeddings of repository r at time t

Research Design: Graph Formulation and Features

The selected features can be categorized by the element of the interaction that impacts the introduction of the vulnerability:

- **User-related features:** The experience, and previous activities of the developers, as well as the frequency and severity of the vulnerabilities introduced influence the propensity of a developer introducing a vulnerability. These factors are operationalized by considering the number of repositories owned, comment activities, a cumulative sum of the vulnerabilities and associated severity before the interaction.

- **Repository-related features:** Characteristics of the repositories, such as the language, its popularity, the number of developers collaborating on it, the number of pre-existing vulnerabilities, as well as pre-existing vulnerabilities and their nature influence the propensity of vulnerabilities being introduced to a repository. These factors are operationalized by the number of stars, open issues, comments, as well as the number and type of pre-existing vulnerabilities.

- **Interaction-related features:** The severity of the vulnerability would influence the propensity of vulnerability introduction. For instance, the incidence of low-severity vulnerabilities is higher than high-severity vulnerabilities.

Research Design: Graph Formulation and Features

Category	Feature	Description	Rationale	Reference	Example
Nodal Features: User	Repositories Owned	The number of repositories owned by a user.	Captures developer experience	Bao et al. 2019	5
	Repositories Starred	The number of repositories starred by a user.	Captures developer activity	Bao et al. 2019	5
	Cumulative Comments	Cumulative number of comments made by a user.	Captures developer activity	Bao et al. 2019	30
	Public Repositories	The number of public repositories starred by a user.	Captures developer activity	Bao et al. 2019	5
	Cumulative Vulnerabilities	Cumulative vulnerabilities before interaction	Risk carried by developer	Lazarine et al. 2019	30
	Cumulative Severity Score	Cumulative severity score of vulnerabilities introduced	Risk severity carried by developer	Lazarine et al. 2019	50
Nodal Features: Repositories	Language	The primary language of the repository.	Vulnerabilities are language specific	Bao et al. 2019	Python
	Fork	Whether the repository is forked.	Captures novel development	Bao et al. 2019	1
	Count of Open Issues	The number of open issues at the time of data collection.	Measure of collaboration and activity	Bao et al. 2019	10
	Stars	Number of stars that a repository has.	Measure of popularity	Bao et al. 2019	10
	Watches	Number of times a repository has been watched.	Measure of popularity	Bao et al. 2019	10
	Forks	Number of times a repository has been forked.	Measure of popularity	Bao et al. 2019	10
	Pull requests	Number of pull requests for the repository.	Measure of popularity	Bao et al. 2019	10
	Size	The size of the repository.	Captures functionality	Bao et al. 2019	5000
	Cumulative Vulnerabilities	Cumulative vulnerabilities before interaction	Risk carried by repository	Lazarine et al. 2019	30
	Cumulative Severity Score	Cumulative severity score of vulnerabilities introduced	Risk severity carried by repository	Lazarine et al. 2019	50
Edge Features	Vulnerability Severity	Severity of vulnerability	Nature of vulnerability associated with specific commit	Lazarine et al. 2019	1 (one hot)

Table 4. Features potentially affecting vulnerability introduction, categorized by graph components

Proposed SeCoPE: Weighted Continuous Propagation Unit

- **GOAL:** What $X(t)(t \in t_k, t_{k+1})$ will be, given $X({t_k}^+)$, before the next interaction

- Node representations at time t: $X(t) = X({t_k}^+) + \int_{t_k}^{t} h \, d\tau$ where the function h is a GNN-based function to model continuous propagation and evolution (CGNN)

- a (spectral radius) controls the extent of impact of the center node on its neighbors

$$L_k = \boxed{a'}(I + D_k^{-\frac{1}{2}} A_k D_k^{-\frac{1}{2}})$$

where A_k is the adjacency matrix of the observable graph at time $(t \in t_k, t_{k+1})$, $\boldsymbol{D}_k$ is the degree matrix and $\alpha \in (0, 1)$ is a parameter controlling the spectral radius of $\boldsymbol{L}_k$

- Thus, we have the following GNN:

$$\frac{d}{dt}\boldsymbol{X}(t) = (\boldsymbol{L_k} - \boldsymbol{I})\boldsymbol{X}(\boldsymbol{t}) + \boldsymbol{E}, (t \in t_k, t_{k+1})$$

- **Intuition: Users have differential impact based on their relative position and influence in the network**

- **The knowledge, information and code from highly influential users would be referenced more: greater impact of influential developers on codebases, and influential codebases on developers**

- **The extent of influence on neighboring nodes can be captured using centrality measures**

- **Eigenvector centrality can capture the transitive influence/ relative prestige score with respect to the entire network, reflecting the hierarchical structures within organizations**

Evaluation

Experiment		Justification	Methods	Description	References	Evaluation Metrics*
1	SeCoPE against RNN-based methods	Recurrent neural network models are commonly used to train nodal embeddings for link prediction	JODIE	Coupled recurrent neural network model; learns embedding trajectories of two types of nodes (e.g., users and repositories)	Kumar et al. 2019; Zhang et al. 2021	Accuracy, Precision, Recall, F1-Score
			CoPE	Modeling continuous propagation and evolution		
2	SeCoPE against Attention-based methods	Evaluate attention-based methods that account for neighboring nodes' attributes	DyREP	Temporal attention layer to capture the neighbors' interactions	Trivedi et al. 2019; Xu et al. 2020; Rossi et al. 2020	
			TGN	"Memory" module to update node embeddings		
			TGAT	Attention layer to efficiently aggregate temporal-topological neighborhood features		

The gold standard dataset is the result of the vulnerability assessment (introduction of vulnerability).

Precision is an important metric because the costs of False Positive is high. In vulnerability introduction prediction, an interaction that will not introduce a vulnerability has been identified as vulnerability introduction.

F1-score is an important metric for comparing models, as it is not sensitive to data imbalance.

80% of the timestamps are used for training, 10% for validation, and 10% for testing.

Results

Dataset	Method	Accuracy	Precision	Recall	F1-Score
EXPERIMENT 1					
NCAR	JODIE	0.569**	0.54**	0.634**	0.556**
	CoPE	0.910**	0.657**	0.663**	0.660**
	seCoPE	**0.959**	**0.764**	**0.704**	**0.733**
EXPERIMENT 2					
NCAR	TGN	0.589**	0.562**	**0.913**	0.692**
	DyREP	0.473**	0.413**	0.383**	0.33**
	TGAT	0.585**	0.556**	0.884**	0.681**
	seCoPE	**0.959**	**0.764**	0.704**	**0.733**

Table 5. Results **: p < 0.01

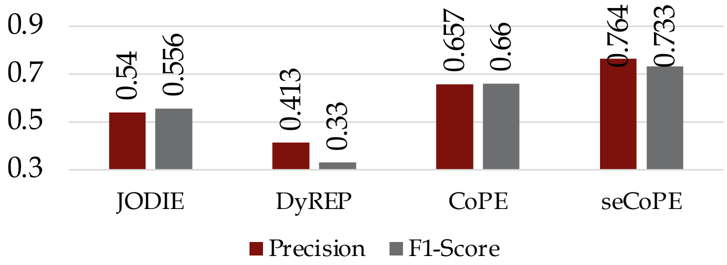


Figure 7. Precision and F1-Score for Experiment 1

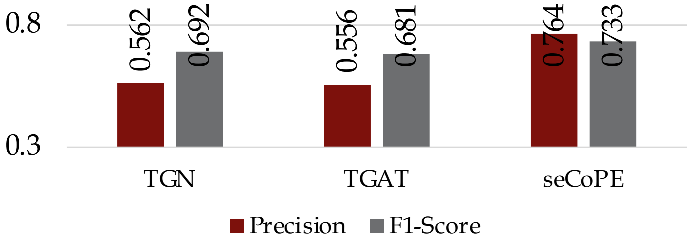


Figure 8. Precision and F1-Score for Experiment 2

Key Observations:

- seCoPE outperforms state-of-the-art RNN-based dynamic graph deep learning
- We perform a one-sided t-test to check for statistical significance and compare each method with the best performing method, i.e., seCoPE for all metrics.
- 16.2% increase for precision for NCAR.

Case Study: Provision of Personalized and Timely Security Trainings

- To demonstrate the practical value of seCoPE, we conduct a case study illustrating the timely identification of high-risk actors it enables.
- The case study demonstrates how seCoPE can be utilized by various stakeholders to identify individual developers such that personalized trainings can be delivered
- Personalized security trainings will better engage developers and reduce vulnerability incidence due to human error

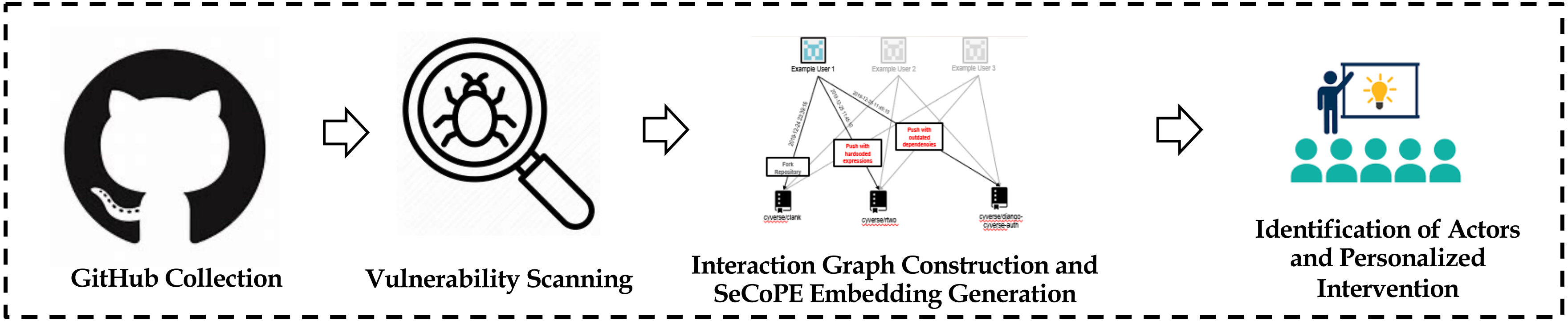


Figure 13. Case Study

Case Study: Provision of Personalized and Timely Security Trainings

- ***For whom is the research significant?***
  - Managed Security Service Provider (MSSP): Monitoring and management of security devices and systems is often outsourced to MSSP's, who can provide personalized security awareness trainings.
  - Internal SOC: An information security team that monitors, detects and analyzes events on the network or system to prevent and resolve issues, and can conduct internal security awareness trainings.

- ***How is the research significant?***
  - Instrumental benefits: Enabling stakeholders to identify high risk actors for targeted security awareness trainings, optimal assignment of developers to repositories, as input in developer scorecards.

- ***How is the research operationalized?***

  The following steps can be performed by security analysts at MSSP's or within organizations:

  1. Collect the GitHub repositories for the organization.
  2. Conduct a vulnerability assessment at the commit-level to obtain vulnerabilities introduced by each user.
  3. Generate an interaction graph.
  4. Train the model, using seCoPE to generate risk profiles (nodal embeddings) for users and repositories.
  5. Create user-repository pairs for varying periods of time. For any given developer and repository, the introduction of a vulnerability can be predicted over the given time period.

Case Study: Provision of Personalized and Timely Security Trainings

- To illustrate the identification of developers, we run the seCoPE model for the first 25% of the dataset. The predictions made by the model are then used to hypothesize about the subsequent training efficacy.
- We plot the number of low severity vulnerabilities (CWE-119/CWE-120/CWE-362/CWE-190) introduced by user "nief" into Cyverse repositories over time.
- We can see that within a category of vulnerabilities, vulnerability introduction can be used to predict future behavior.
- When run on the first 25% of the dataset, the seCoPE model is able to successfully predict the encircled vulnerability, i.e., commit #cd22f58.

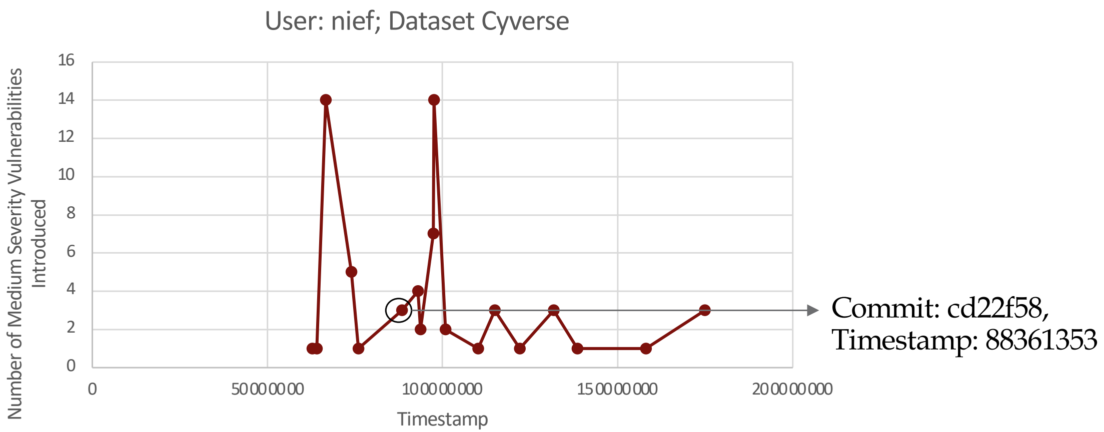


Figure 14. Vulnerability Introduction by User nief

Case Study: Provision of Personalized and Timely Security Trainings

- CWE 119/ 120 can lead to potential memory overflows, CWE-362 can lead to attackers opening malicious files on the device, and CWE-120 can lead to buffer overflows when copying.

- If targeted security training is provided to user before time 97547292, the introduction of the vulnerabilities in the highlighted area can potentially be mitigated.

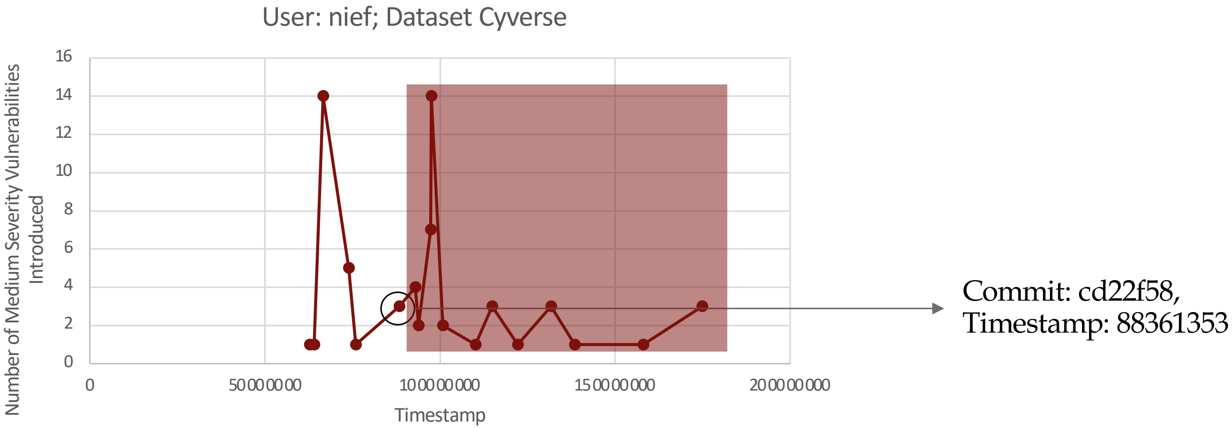


Figure 15. Vulnerability Introduction by User nief

Conclusion and Future Directions

Implications for practice:

- Reduce the incidence of vulnerabilities
- Efficiently utilize training resources
- Investment in human capital through workforce development

Several promising directions for future research:

- This design artifact can be deployed in a field experiment to compare the user perceptions and long-term efficacy of using targeted vs. non-targeted trainings.
- Future research can contextualize the proposed link prediction approach to identify awareness and training needs at different granularities, e.g., developers, teams, or departments, or in different contexts, e.g., technologies or projects
- The predictive model can be improved by incorporating data available to the firm that could impact the propagation of knowledge, such as communication logs and prior trainings

Thank you!

Questions or Comments?

—

Agrim Sachdeva
Kelley School of Business, Indiana University
agsach@iu.edu

References

Al-Rubaye, A., and Sukthankar, G. 2020. "Scoring Popularity in Github," 2020 International Conference on Computational Science and Computational Intelligence (CSCI): IEEE, pp. 217-223.

Alfadel, M., Costa, D. E., and Shihab, E. 2021a. "Empirical Analysis of Security Vulnerabilities in Python Packages," 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER): IEEE, pp. 446-457.

Alfadel, M., Costa, D. E., Shihab, E., and Mkhallalati, M. 2021b. "On the Use of Dependabot Security Pull Requests," 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), pp. 254-265.

Balcilar, M., Renton, G., Héroux, P., Gaüzère, B., Adam, S., & Honeine, P. (2020, July). Spectral-designed depthwise separable graph neural networks. In Proceedings of Thirty-seventh International Conference on Machine Learning (ICML 2020)-Workshop on Graph Representation Learning and Beyond (GRL+ 2020).

Bidoki, N. H., Schiappa, M., Sukthankar, G., and Garibay, I. 2020. " Modeling Social Coding Dynamics with Sampled Historical Data," Online Social Networks and Media (16), p. 100070.

Bruna, Joan, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. "Spectral networks and locally connected networks on graphs." arXiv preprint arXiv:1312.6203 (2013).

Gong, Q., Zhang, J., Chen, Y., Li, Q., Xiao, Y., Wang, X., and Hui, P. 2019. "Detecting Malicious Accounts in Online Developer Communities Using Deep Learning," Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 1251-1260.

Gonzalez, D., Zimmermann, T., and Nagappan, N. 2020. "The State of the Ml-Universe: 10 Years of Artificial Intelligence &Amp; Machine Learning Software Development on Github," in: Proceedings of the 17th International Conference on Mining Software Repositories. Seoul, Republic of Korea: Association for Computing Machinery, pp. 431–442.

Goyal, P., and Ferrara, E. 2018. "Graph Embedding Techniques, Applications, and Performance: A Survey," Knowledge-Based Systems (151), pp. 78-94.

Grieco, G., Grinblat, G. L., Uzal, L., Rawat, S., Feist, J., and Mounier, L. 2016. "Toward Large-Scale Vulnerability Discovery Using Machine Learning," Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, pp. 85-96.

Guo, Q., Chen, S., Xie, X., Ma, L., Hu, Q., Liu, H., Liu, Y., Zhao, J., and Li, X. 2019. "An Empirical Study Towards Characterizing Deep Learning Development and Deployment across Different Frameworks and Platforms," 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE): IEEE, pp. 810-822.

References

Guo, Z., Tang, L., Guo, T., Yu, K., Alazab, M., and Shalaginov, A. 2021. " Deep Graph Neural Network-Based Spammer Detection under the Perspective of Heterogeneous Cyberspace," Future Generation Computer Systems (117), pp. 205-218.

Kumar, S., Zhang, X., and Leskovec, J. 2019. " Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks," Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining), pp. 1269-1278.

Lazarine, B., Samtani, S., Patton, M., Zhu, H., Ullman, S., Ampel, B., and Chen, H. 2020. "Identifying Vulnerable Github Repositories and Users in Scientific Cyberinfrastructure: An Unsupervised Graph Embedding Approach," 2020 IEEE International Conference on Intelligence and Security Informatics (ISI)), pp. 1-6.

Lazarine, B., Zhang, Z., Sachdeva, A., Samtani, S., and Zhu, H. 2022. "Exploring the Propagation of Vulnerabilities from Github Repositories Hosted by Major Technology Organizations," Proceedings of the 15th Workshop on Cyber Security Experimentation and Test, pp. 145-150.

Lü, L., and Zhou, T. 2011. "Link Prediction in Complex Networks: A Survey," Physica A: Statistical Mechanics and its Applications (390:6), pp. 1150-1170.

Oostema, P., and Franchetti, F. 2021. "Leveraging High Dimensional Spatial Graph Embedding as a Heuristic for Graph Algorithms," 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW): IEEE, pp. 539-547.

Pakhira, M. K., Bandyopadhyay, S., and Maulik, U. 2004. "Validity Index for Crisp and Fuzzy Clusters," Pattern recognition (37:3), pp. 487-501.

Sedova, M. 2017. "Comparing Educational Approaches to Secure Programming: Tool Vs.{Ta}," Thirteenth Symposium on Usable Privacy and Security (SOUPS) 2017, July 12–14, 2017, Santa Clara, California.

Shahapure, K. R., and Nicholas, C. 2020. "Cluster Quality Analysis Using Silhouette Score," 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA): IEEE, pp. 747-748.

Shahid, W. B., Aslam, B., Abbas, H., Khalid, S. B., and Afzal, H. 2022. "An Enhanced Deep Learning Based Framework for Web Attacks Detection, Mitigation and Attacker Profiling," Journal of Network and Computer Applications (198), p. 103270.

Shaji, E., and Subramanian, N. 2021. "Assessing Non-Intrusive Vulnerability Scanning Methodologies for Detecting Web Application Vulnerabilities on Large Scale," 2021 International Conference on System, Computation, Automation and Networking (ICSCAN): IEEE, pp. 1-5.

Sonnenburg, S., Rätsch, G., Henschel, S., Widmer, C., Behr, J., Zien, A., Bona, F. d., Binder, A., Gehl, C., and Franc, V. 2010. "The Shogun Machine Learning Toolbox," The Journal of Machine Learning Research (11), pp. 1799-1802.

Stanton, B., Theofanos, M. F., Prettyman, S. S., and Furman, S. 2016. "Security Fatigue," IT Professional (18:5), pp. 26-32.

Vance, A., Eargle, D., Jenkins, J. L., Kirwan, C. B., and Anderson, B. B. 2019. " The Fog of Warnings: How Non-Essential Notifications Blur with Security Warnings " Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)), pp. 407-420.

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. 2019. "Deep Graph Infomax," ICLR (Poster) (2:3), p. 4.

Von Krogh, G., and Von Hippel, E. 2006. "The Promise of Research on Open Source Software," Management Science (52:7), pp. 975-983.