Detecting Malicious Accounts in Online Developer Communities Using Deep Learning

Yang Chen
School of Computer Science
Fudan University
chenyang@fudan.edu.cn

Joint work with Q. Gong, J. Zhang, X. Wang (Fudan),
Q. Li (Tsinghua), Y. Xiao (Aalto), P. Hui (U. Helsinki / HKUST)

About Me

2000-2009

2009-2011

2011-2014

2014-now

- Associate Professor at Fudan University &
  Nokia Visiting Professor at Aalto University
- Leading the mobile systems and networking (MSN) group
- Focusing on computer networks, computational social systems and applied machine learning

Background of Developer Communities

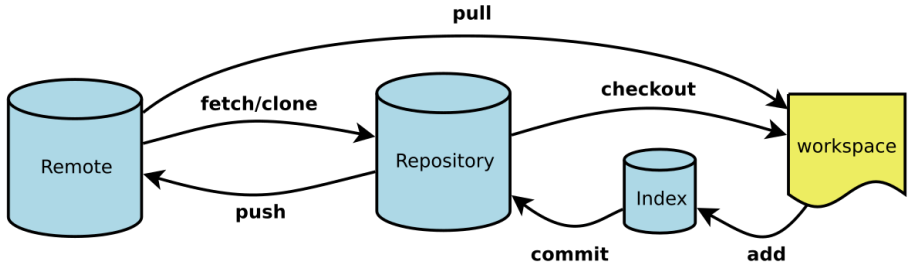
- Online developer communities provide platforms for millions of developers

- Such communities themselves become a unique type of online social networks (OSNs)

GitHub ABC

- GitHub has attracted 31 million developers, holding 96 million repositories

- GitHub regards each user activity as an event



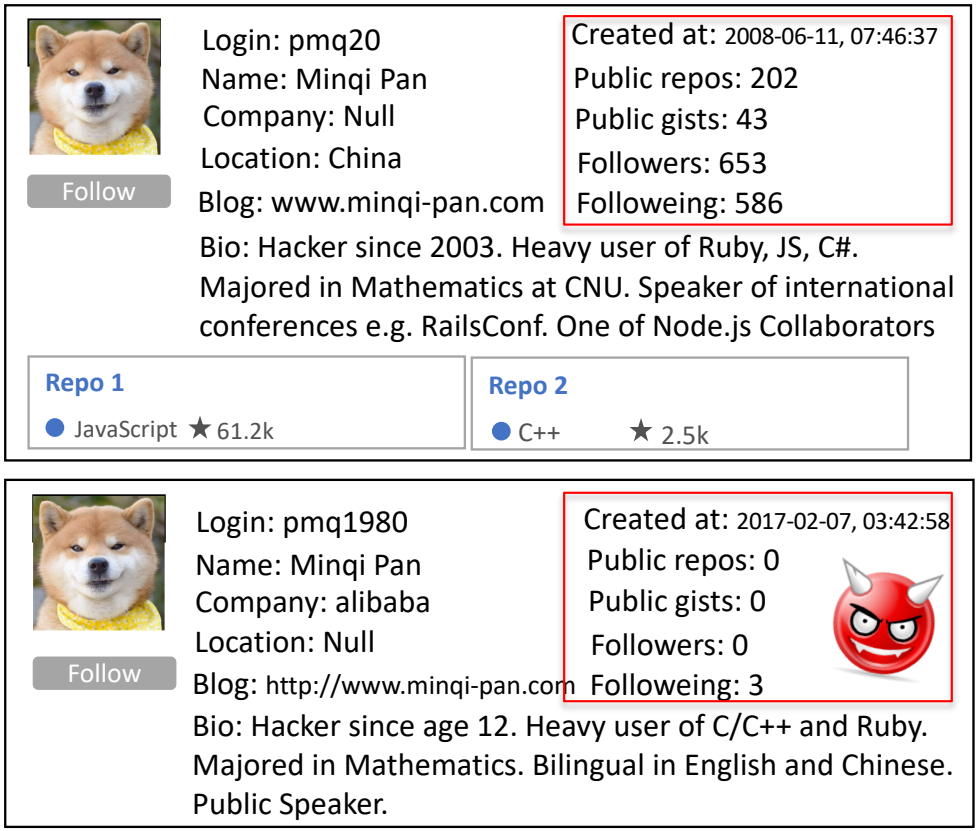
- Developers interact with each other with a main focus on collaborative development and code sharing
  - Forming a special kind of *social network*

github SOCIAL CODING

Malicious Users from GitHub Dataset

1/3 Example

- Identity impersonation
  - Copy famous developers' homepage to attract visitors

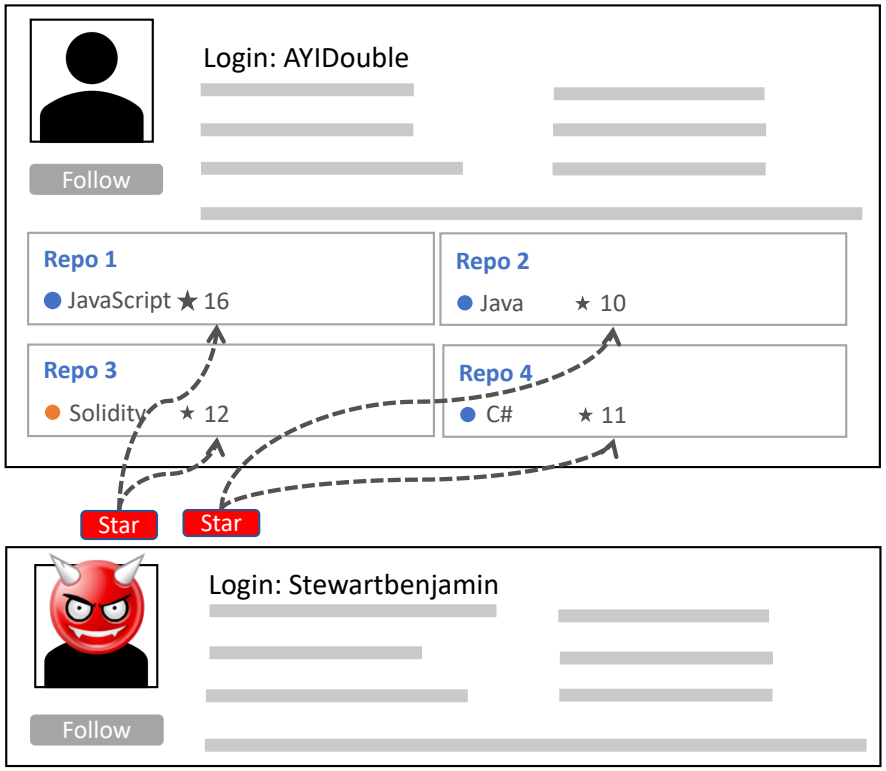


Help the attackers exploit the reputation of the victims

Malicious Users from GitHub Dataset

2/3 Example

- Fake stars
  - Star a batch of repositories for bonus

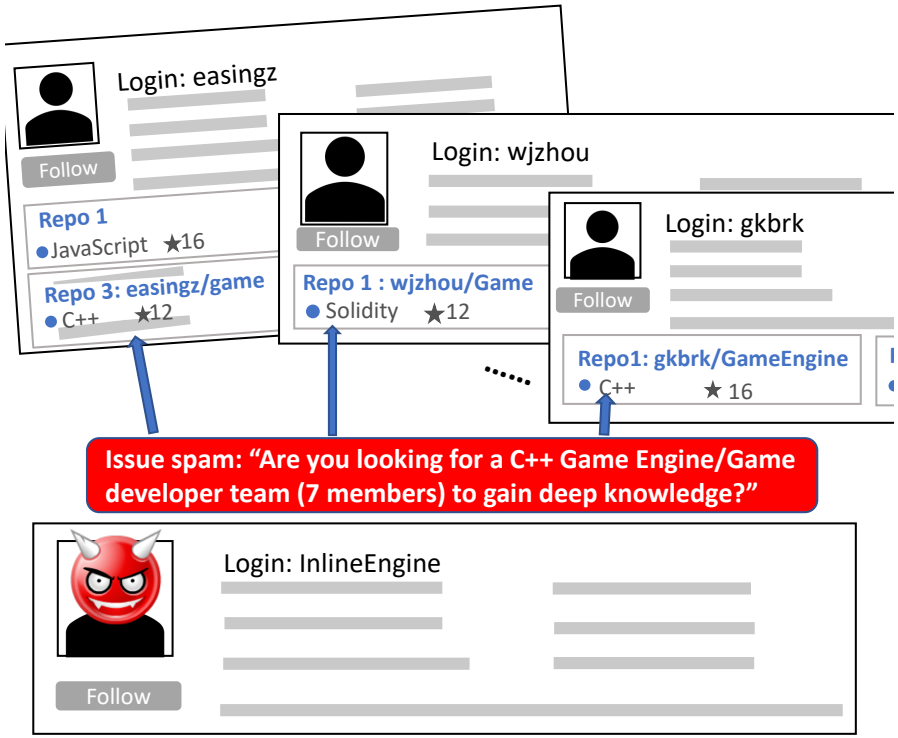


Make one user's repositories look popular

Malicious Users from GitHub Dataset

3/3 Example

- Issue spams to related repositories



Send a "Game developer" advertisement to as many game-related repositories as possible

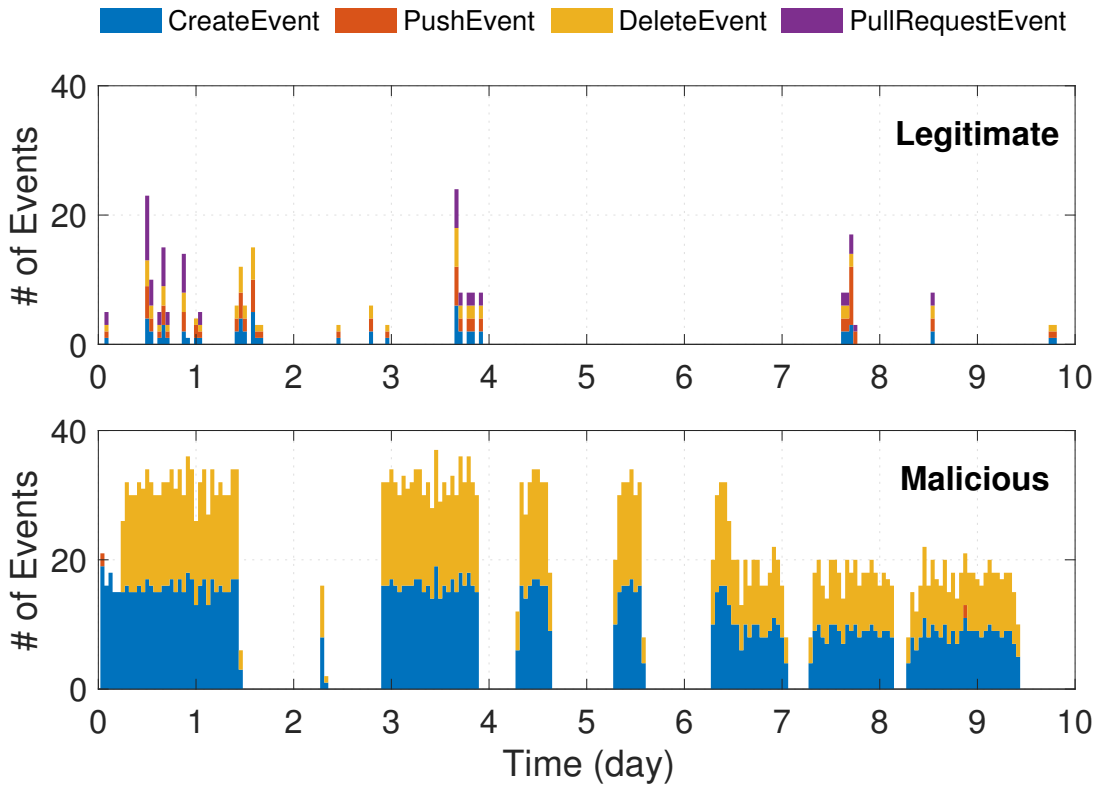
Data Collection and Labeling

- Time period: Jun. 20, 2018 – Aug.27,2018
- Amount: 10,667,583 randomly selected GitHub users (21.5% malicious v.s. 78.5% legitimate)
- Format:
  - Demographic information (user ID, gender, registration date, number of followings/followers,…)
  - Statistical information about historical activities (number of public repositories, public gists,…)
  - Label signifying whether the account has been banned by GitHub (homepage “404” & accessible via API)
  - Historical (dynamic) events from GH Archive (https://www.gharchive.org/)

Ethical Issues

- All information we collected was publicly accessible
- We follow the “terms of service” of GitHub
- We have consulted GitHub about our research

Difference on Activities between Legitimate and Malicious Users



- Legitimate users tend to conduct more types of activities, in an irregular temporal mode
- Malicious users conduct similar types of activities, usually continuously

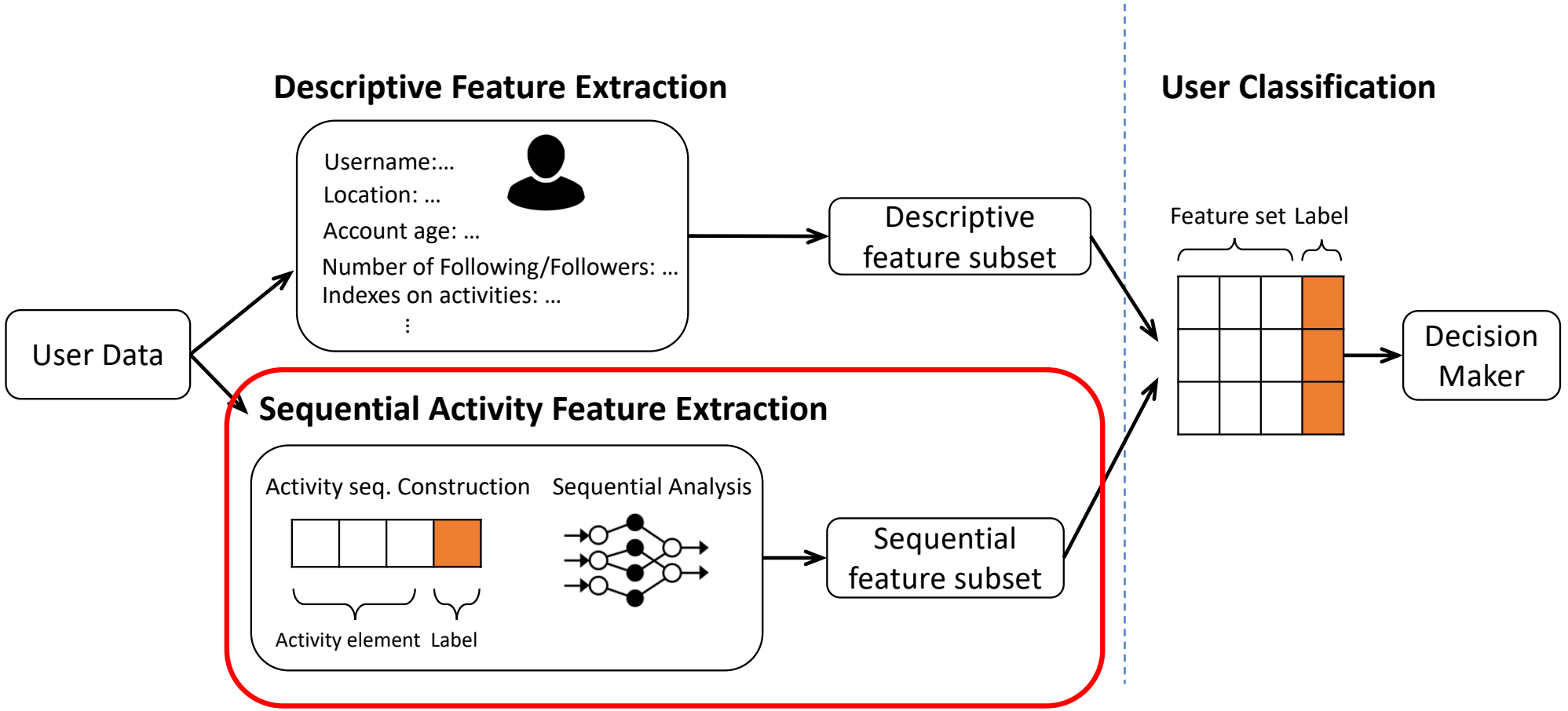
Challenges in GitHub User Activities Analysis

- Rich types of possible activities (e.g. 42 event types on GitHub)
- Irregular activity timeline, highly dependent on working schedules

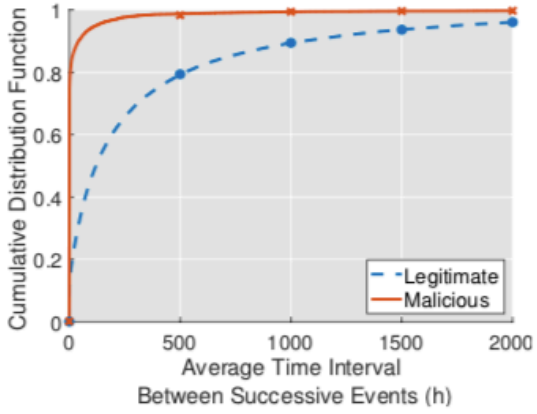
Feature Selection

- The public data of each GitHub user consists of a descriptive part and a dynamic part
  - The descriptive part mainly refers to the information about a user's profile and a set of statistical metrics of her activities
  - The dynamic part covers the fine-grained records of the activities users have generated

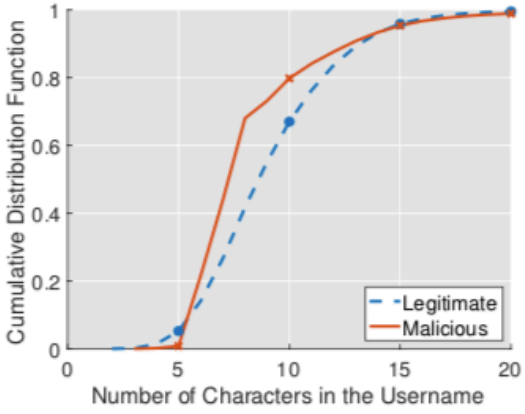
GitSec Design



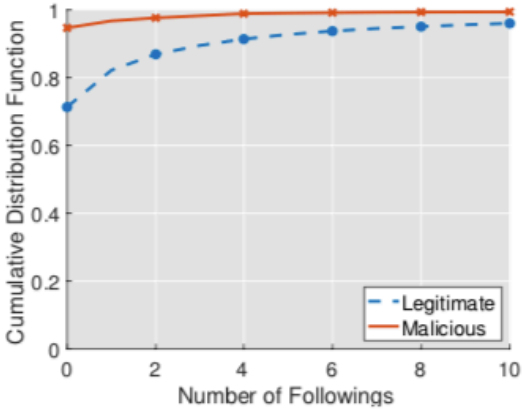
Behavioral Difference Between Legitimate and Malicious Users (Descriptive Features)



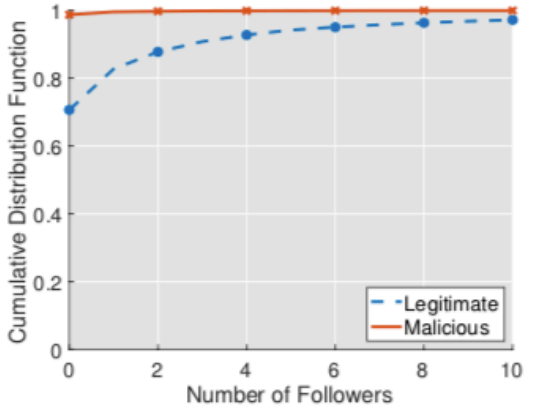
(a) Average Time Interval Between Successive Events



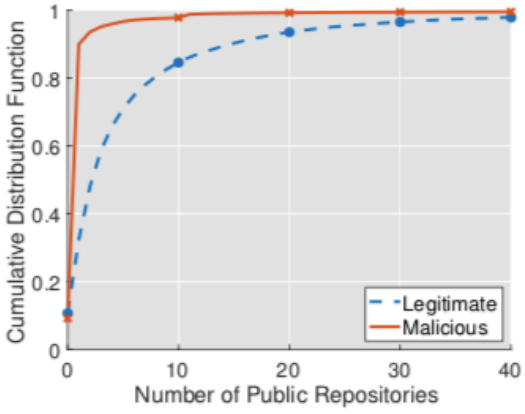
(b) Number of Characters in Username



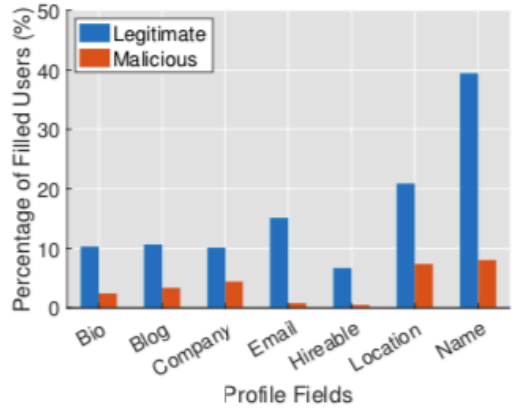
(c) Number of Followings



(d) Number of Followers

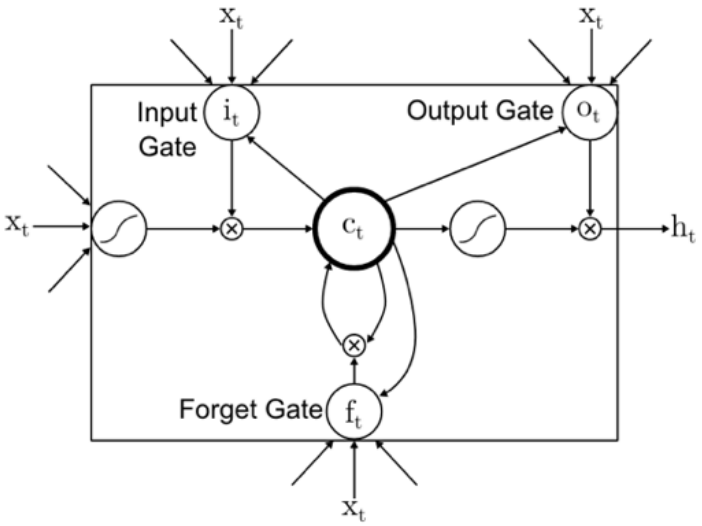


(e) Number of Repositories

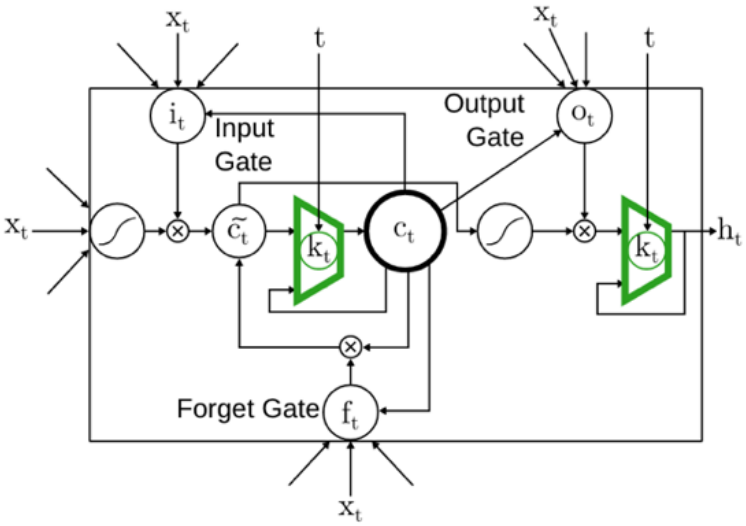


(f) Profile Fields

LSTM v.s. Phased LSTM



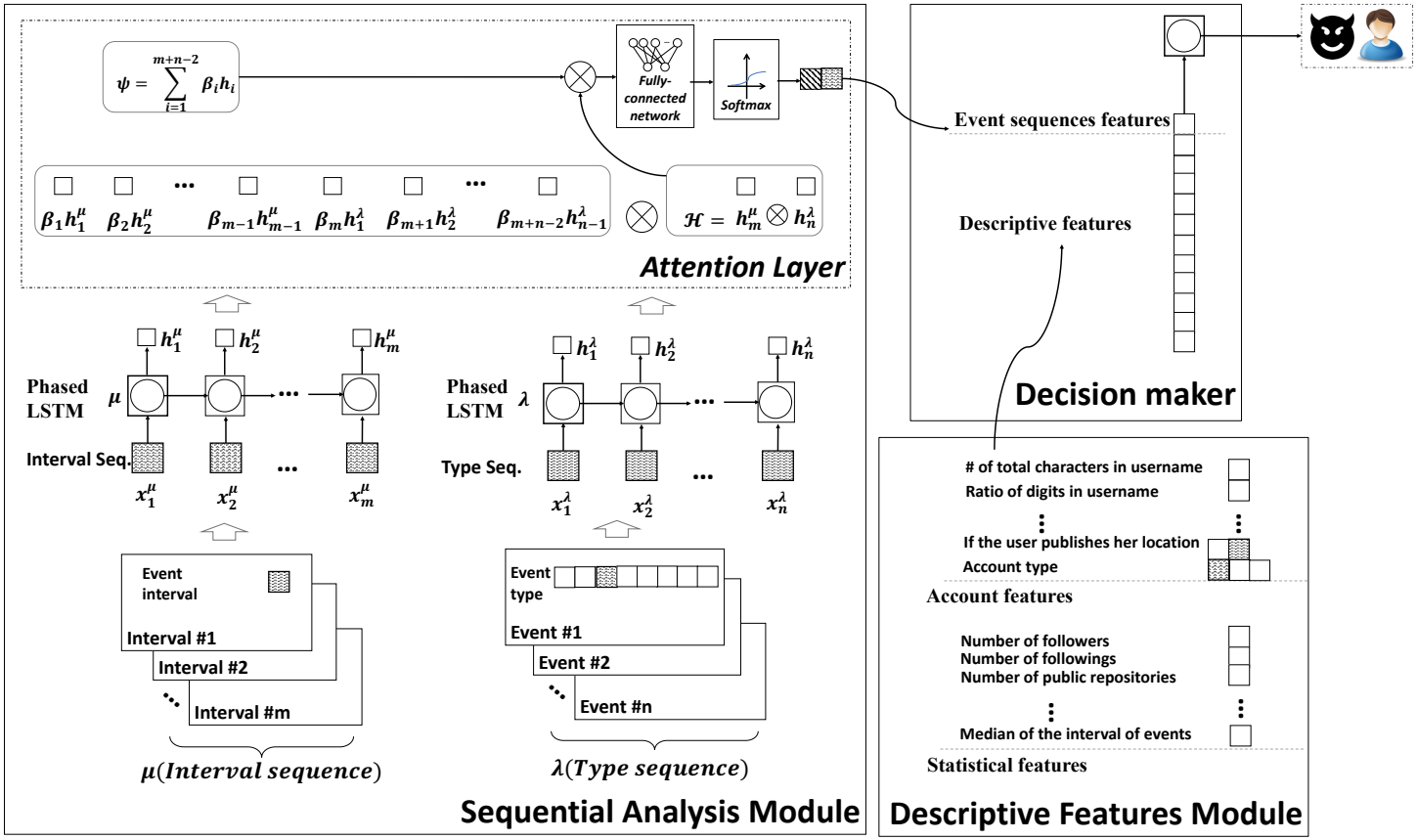
LSTM



Phased LSTM

- LSTM networks *[Hochreiter et al., Neural Computation'97]* regard the elements in the input sequence equally and update the cell state when processing each element
  - GitHub events are often sparse and distributed in a wide time range
- Phased LSTM *[Neil et al., NIPS'16]* extends the standard LSTM network by adding an additional gate over the updates of the cell status
  - Phased LSTM can deal with long & sparse sequences efficiently

GitSec: Multi-source activity analysis with coupled DNNs



- Two event sequences: event interval seq. & event type seq.
- Coupled deep neural networks to deal with different event seq.
- Attention mechanism to connect the two PLSTMs

Implementation

- Phased LSTM-based time series analysis
  - TensorFlow
- Decision maker
  - Scikit-learn
- Dataset for evaluation
  - Randomly selected 59,875 users (44,892 vs 14,965)
  - 7:3 for training and test datasets

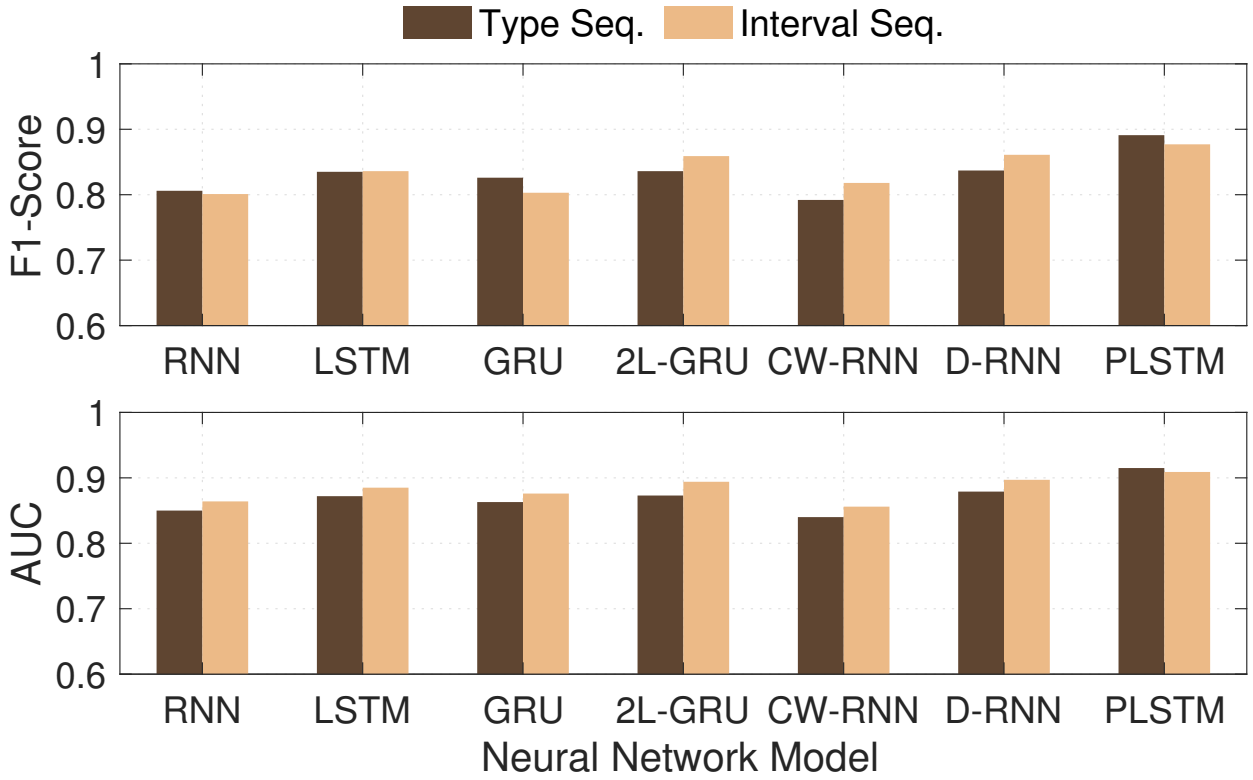
Metrics

- Precision
  - The fraction of predicted malicious accounts who are really harmful
- Recall
  - The fraction of malicious users who are detected accurately
- F1-score
  - The harmonic mean of precision and recall

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

- AUC
  - The probability that this classifier will rank a randomly chosen malicious user higher than a randomly selected legitimate user

Evaluation on Different Neural Networks



- We run different neural network models on the event type sequence (Type seq.) and the event interval sequence (Interval seq.), respectively, and compare their performance
- PLSTM performs the best for both the sequences

Comparison of Different Attention Mechanisms

Models	Precision	Recall	F1-score	AUC
PLSTM with the combined time series seq.	0.915	0.825	0.868	0.900
Parallel PLSTM	0.928	0.883	0.905	0.930
Parallel PLSTM + AttentionLoc	0.928	0.887	0.906	0.931
Parallel PLSTM + AttentionConcat	0.924	0.892	0.907	0.934

- The parallel design produces higher F1-score and AUC value than taking the combined time series sequence as the input
  - The result confirms the necessity to using a parallel design
- The advantage of the attention methods shows the importance of considering the relations between the two event sequences

Performance Evaluation on Different Feature Subsets

Feature sets	Precision	Recall	F1-score
GitSec	0.951	0.892	0.920
- Event features	0.928	0.854	0.889
-Account features	0.945	0.889	0.916
-Statistical features	0.943	0.887	0.914

- Starting from GitSec, we delete one feature subset at a time
- The performance decreases the most after deleting the event features

Performance Evaluation on Different Feature Subsets

Feature sets	Precision	Recall	F1-score
Random Guess	0.248	0.496	0.330
+ Event Features	0.940	0.880	0.909
+Account features	0.594	0.678	0.633
+Statistical features	0.923	0.846	0.883

- Starting from a random guess classifier, we add one feature subset at a time
- Adding the Event feature subset could increase the F1-score the most

Comparison with Existing Malicious Account Detection Approaches

Approach	Precision	Recall	F1-score
GitSec	0.950	0.892	0.920
DeepScan [1]	0.932	0.844	0.886
Al-Qurishi et al. [2]	0.870	0.804	0.836
Viswanath et al. [3]	0.479	0.937	0.634

- DeepScan: single activity sequence design using an LSTM network
- Al-Qurishi et al.: random Forest classifier with the feature preprocessed by PCA
- Viswanath et al.: using PCA to process the raw user data and separate the residual space from the normal space, and distinguish malicious users from the features in the residual space

[1] Gong et al. IEEE Communications Magazine, 2018
[2] Al-Qurishi et al. IEEE Transactions on Industrial Informatics, 2018
[3] Viswanath et al. Proc. of USENIX Security, 2014

Future Work

- Evaluate GitSec with the data of other online developer communities
- Collaborate with some developer communities to take back-end user activities into consideration
  - By integrating the clickstream information and the entire social graph into our solution

Thank you!

https://chenyang03.wordpress.com/