BotFlowMon
Learning-Based, Content-Agnostic Identification of Social Bot Traffic Flows

Yebo Feng, Jun Li, Lei Jiao
University of Oregon
{yebof, lijun, jiao}@cs.uoregon.edu

Xintao Wu
University of Arkansas
xintaowu@uark.edu
Online Social Network with Social Bots

- Online social networks (OSNs) are increasingly threatened by software-controlled social bots.
  - They impersonate real OSN users.
  - Majority of OSN malicious activities come from social bots (80% - 90%).
- They can, for example:
  - infiltrate an OSN
  - launch spam campaigns
  - spread fraud information
  - collect private data, and
  - perform financial fraud.
Social Bot Examples

• Post bot
• Chatbot
• Amplification bot
• OSN crawler bot
• Hybrid bot
Post Bot

- Automatically post spam tweets, Facebook posts, etc.
- Can contain malicious URLs or texts
- The most common social bots

Examples are from medium.com - Analyzing a bot attack on a news site in the United States.
Chatbot

- Automatically converse with regular users

Hello, I am a bot from University of Oregon, your digital companion.
Is there something you want to talk about?
Amplification bot

- Amplify OSN accounts via fake followers, or
- Amplify content by, for example, artificial retweets and likes

Examples are from the searching results of ebay.com.
OSN Crawler Bot

• Page crawler
  • read the HTML files of OSN users
• API crawler
  • become friends of OSN users and fetch their information via API calls

API sample code:

```javascript
api.GetUser(user)
api.GetReplies()
api.GetUserTimeline(user)
api.GetHomeTimeline()
api.GetStatus(status_id)
api.GetStatuses(status_id)
api.GetFriends(user)
api.GetFollowers()
api.GetFeatured()
```
Outline

• Related work
• BotFlowMon Overview
• Five Modules of BotFlowMon
• Evaluations
• Conclusions
Related Work

• **Content-based approaches**
  • Rely on post syntax, content, account activity, post linguistic features, etc.
  • Only executable by OSN providers
  • Could incur severe privacy concerns

• **Topology-based approaches**
  • Use topology structure of an online social network

• **Crowdsourcing-based approaches**
  • Also only executable by OSN providers
  • Could incur moderate privacy concerns
  • Ask participants to judge whether an account is a bot or not
  • Incur a long running time, a high cost, and privacy risk
Outline

- Related work
- **BotFlowMon Overview**
- Five modules of BotFlowMon
- Evaluations
- Conclusions
BotFlowMon: Content-Agnostic Social Bot Detection

- BotFlowMon can process traffic flow data to distinguish social bot traffic flows from real OSN user flows.
- Any network service provider can deploy BotFlowMon
- Only need metadata of traffic flows
  - No IP packet payload data is needed
  - Fast and scalable
  - Privacy-preserving
BotFlowMon has two modes:
- training mode: which uses labeled NetFlow data to derive a classification model.
- detection mode: which uses the classification model to detect social bot flows from the input traffic flows.

With five modules:
- Preprocessing
- Flow aggregation
- Transaction fingerprint generation
- Transaction subdivision
- Machine learning & classification
BotFlowMon Architecture

Preprocessing

... OSN flow extraction ...

Flow Aggregation

OSN flows for different IPs

Transaction Fingerprint Generation

OSN flows for transactions

Transaction Subdivision

Transaction fingerprints

Machine Learning & Classification

Action fingerprints

Any External Defense System

Data labeled for training

Data unlabeled for inference

Detected social bot flows
BotFlowMon Architecture

Preprocessing

Flow Aggregation

Transaction Fingerprint Generation

Transaction Subdivision

Machine Learning & Classification

Any External Defense System

NetFlow Data

BGP Data

Flows

OSN IP prefixes

... OSN flow extraction ...

OSN flows for different IPs

OSN flows for transactions

Transaction fingerprints

Action fingerprints

Detected social bot flows

Data labeled for training

Data unlabeled for inference
BotFlowMon Architecture

- Preprocessing
  - NetFlow Data
  - BGP Data
  - Flows
  - OSN IP prefixes

- OSN flow extraction
- OSN flows for different IPs

- Flow Aggregation
- Transaction Fingerprint Generation
- Transaction Subdivision
- Machine Learning & Classification

- Any External Defense System
- Detected social bot flows

Data labeled for training
Data unlabeled for inference
BotFlowMon Architecture

Preprocessing

Flow Aggregation

OSN flow extraction

OSN flows for different IPs

Transaction Fingerprint Generation

OSN flows for transactions

Transaction fingerprints

Transaction Subdivision

Action fingerprints

Machine Learning & Classification

Any External Defense System

NetFlow Data

BGP Data

Flows

OSN IP prefixes

Detected social bot flows

Data labeled for training

Data unlabeled for inference
BotFlowMon Architecture

Preprocessing

OSN flow extraction

Flow Aggregation

OSN flows for different IPs

Transaction Fingerprint Generation

OSN flows for transactions

Transaction Subdivision

Transaction fingerprints

Machine Learning & Classification

Action fingerprints

Any External Defense System

Detected social bot flows

Data labeled for training

Data unlabeled for inference
NetFlow Data Format

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%ts</td>
<td>Start time – first seen</td>
<td>%das</td>
<td>Destination AS number</td>
</tr>
<tr>
<td>%te</td>
<td>End time – last seen</td>
<td>%in</td>
<td>Input interface number</td>
</tr>
<tr>
<td>%td</td>
<td>Duration</td>
<td>%out</td>
<td>Output interface number</td>
</tr>
<tr>
<td>%pr</td>
<td>Protocol</td>
<td>%pkt</td>
<td>Number of packets</td>
</tr>
<tr>
<td>%sa</td>
<td>Source address</td>
<td>%byt</td>
<td>Number of bytes</td>
</tr>
<tr>
<td>%da</td>
<td>Destination address</td>
<td>%fl</td>
<td>Number of flows</td>
</tr>
<tr>
<td>%sap</td>
<td>Source address port</td>
<td>%flg</td>
<td>TCP flag</td>
</tr>
<tr>
<td>%dap</td>
<td>Destination address port</td>
<td>%tos</td>
<td>Type of service</td>
</tr>
<tr>
<td>%sp</td>
<td>Source port</td>
<td>%bps</td>
<td>Bits per second</td>
</tr>
<tr>
<td>%dp</td>
<td>Destination port</td>
<td>%pps</td>
<td>Packets per second</td>
</tr>
<tr>
<td>%sas</td>
<td>Source AS number</td>
<td>%bpp</td>
<td>Bytes per packet</td>
</tr>
</tbody>
</table>

- The information to leverage from NetFlow data is simple and straightforward.
- Content-agnostic, highly summarized information from packet headers.
Transactions & Actions

• BotFlowMon introduces two key concepts to study OSN traffic flows:
  • transactions
  • actions
• BotFlowMon aggregate flows into transactions
• Every transaction is composed of actions
• It can classify actions into bot actions and real user actions, and then classify the transactions based on how their actions are classified.
Outline

• Related work
• BotFlowMon Overview
• Five Modules of BotFlowMon
• Evaluations
• Conclusions
Preprocessing

- Raw NetFlow data is noisy and messy.
- This module:
  - denoise traffic flows,
  - extract OSN related traffic flows,
  - filter out flows with irrelevant protocols,
  - group flows by IP addresses, and
  - sort flows by timestamps.
Flow Aggregation

Problem:
• No sufficient information from individual NetFlow records to detect social bot flows
• Both social bot and real OSN user behaviors are conducted at the application level.

Aggregate adjacent flows into a transaction:
• More representative of application activities
• with more information to inspect collective OSN behaviors of flows
Flow Aggregation

1. Divide each flow’s duration into time bins of equal length, and define a flow point for each time bin.
2. Use modified DBSCAN to group flow points into clusters.
3. Inspect the time window of each cluster. The flows that fall within this window will belong to this transaction.
Problem:
• A transaction is a set of NetFlow records.
• Not regularized or normalized for comparison

Transaction fingerprint – a data fusion technique
• Derives an $f \times N$ matrix from every transaction
• Use this matrix as the fingerprint of the transaction
• Directly comparable and easy to visualize
Transaction Fingerprint Generation

Transaction Fingerprint:

- $f \times N$ matrix
  - $N$ is the number of time bins of equal length within the time window of the transaction
  - $f$ is the number of features over each time bin
- E.g. $6 \times 200$
  - Incoming/outgoing bps, pps, ToS
  - Visualizable
Transaction Fingerprint Example 1

Human user

Chatbot
Open the browser and load the page

Post 2 tweets that contain images

Reload the page

Post one tweet that contains image
Transaction Subdivision

Problem:
- There can be countless types of transactions.
- Each transaction can be of an arbitrary duration.
- Size of training data is limited.

Subdivide a transaction into actions:
- Easier to differentiate bot actions from real user actions
- Reduce required training data size
- Increase training speed & detection accuracy
Transaction Subdivision

Subdivision Algorithm:
- A new density-valley-based clustering algorithm
  - Parameter $r$ (duration threshold)
  - Find out all the density valley points
  - Choose the valley points with enough contrast with surroundings as subdivision moments of a transaction
- Output: a set of action fingerprints

\[ \text{Den}(p) = \sum_f \frac{\text{Byte}_f}{\text{Duration}} \]
• A post bot’s transaction is subdivided into five actions in this case.
• Each action now has a more outstanding pattern than the original transaction fingerprint.
A transaction by a real user that is composed of two actions:

- one was opening an OSN site, and
- the other was scrolling down the page of the OSN site.
BotFlowMon uses Multilayer Perceptron and Conventional Neural Network as its training approaches.

- **Input:** a set of action fingerprints.
- **intermediary output:** labeled action fingerprints.
- **Then,** use action fingerprints to vote for their transaction fingerprint’s label.
- **Final output:** transaction fingerprints’ labels.
Outline

• Related work
• BotFlowMon Overview
• Five Modules of BotFlowMon
• Evaluations
• Conclusions
Bot Simulations

- Leverage existing open-sourced bot programs and frameworks
  - E.g., Botmaster, ChatterBot, PhantomBot
- Develop home-grown bot programs
- Five types of bots are simulated
  - post bot, chatbot, amplification bot, OSN crawler bot, hybrid bot
Data Collection

• NetFlow data collected from University of Oregon campus network
  • Data from realistic scenarios for analysis and verification
  • 507GBs are collected
• NetFlow data collected from experimental computers and routers
  • has superior flexibility and conveniences for simulation, data collection, and experiments
  • 28GBs are collected
The subdivision algorithm’s purity scores with different r values.

Optimal results: when r is in the range of 18 to 23.

Doesn’t need to precisely partition all the transactions. It is designed to make data more friendly to the machine learning process.
Transactions and Actions

1. Randomly select 100 bot and 100 real user transactions
2. Conduct subdivision
3. Record the number of resulted actions and average duration

- Bot transactions tend to have more actions
- Bot actions have shorter durations
Detection Accuracy

- **6 × 200 transaction fingerprint**
  - Incoming/outgoing bps, pps, ToS
  - CNN Accuracy: 0.9361
    - Precision: 0.9887
    - Recall: 0.9067
    - F1 score: 0.9459

- CNN is slightly better than MLP
- The subdivision increases the accuracy significantly
Detection Accuracy

- Remove features from ToS field
  - ToS can be modified by third parties
  - Test the universality
- Accuracies almost remain the same
- Totally usable without ToS

<table>
<thead>
<tr>
<th>Model</th>
<th>Without subdivision</th>
<th>With subdivision</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP – 6*200</td>
<td>0.727</td>
<td>0.901</td>
</tr>
<tr>
<td>MLP – 4*200</td>
<td>0.717</td>
<td>0.912</td>
</tr>
<tr>
<td>CNN – 6*200</td>
<td>0.749</td>
<td>0.936</td>
</tr>
<tr>
<td>CNN – 4*200</td>
<td>0.730</td>
<td>0.923</td>
</tr>
</tbody>
</table>
Limitations

- Not all the social bots are malicious
  - The boundary between “good” bots and “bad” bots can be blurry.
  - Distinguishing social bots with malicious intentions from those that are innocent is hard to achieve without payload data.
- May not be able to detect zero-day social bots
  - Learning-based approach, fully depends on training dataset
Outline

• Related work
• BotFlowMon Overview
• Five Modules of BotFlowMon
• Evaluations
• Conclusions
Conclusions

Social bots are becoming far more sophisticated and threatening than before.

Our contributions:
- **BotFlowMon**: flow-level social bot traffic identification
  - tackles big networking data to identify the traffic of OSN bots
  - content-agnostic, privacy-preserving and efficient
  - Easy to deploy by both OSN providers and ISPs
- Several new techniques and algorithms
  - an aggregation technique that derives transactions
  - a data fusion technique that extracts features from transactions and actions
  - a density-valley-based clustering algorithm
THANK YOU!

This material is based upon work supported by the National Science Foundation under Grant No.1564348. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.