CNT 5410: Computer and Network Security
Final Report: Modeling Network Effects of Continuous Location Sharing Application Abuse

Bernardo Broeto Ponti Medeiros
b.broetopontimed@ufl.edu

December 6, 2024

1 Introduction

Intimate partner violence (IPV) is a pervasive problem; over one third of women (37.3%) and men (30.9%) in the United States experience some form of IPV over their lifetime [12]. Victims find themselves at particularly high risk of digital privacy and security violations at the hands of their abusers, even after they have begun attempting to sever ties [8]. Indeed, many abusers take advantage of shared technologies to perpetrate abuse and further restrict victim autonomy [6]. Though abusers are generally *UI-bound adversaries* who interact with victims' devices through standard user interfaces, the intimate nature of the relationship often means abusers have unfettered access to personal info, facilitating harassment, unwanted contact, and other negative behaviors [6, 8].

Victims are vulnerable to abuse through continuous location sharing (CLS) applications, which provide users the ability to check other users' locations with few constraints; many of these apps require users to "opt out" of location sharing, allowing abusers to continue spying on unwitting victims even after contact has been broken.

A focus on CLS applications is particularly salient given the increasing prominence of location sharing services; a recent poll found that two in ten (16%) of U.S. adults have location sharing activated on their phones at all times, with this figure increasing to four in five (79%) for those who have location sharing active some of the time [7]. Furthermore, existing research links IPV with post-relationship stalking, making clear that CLS applications are likely to be of enduring relevance for survivors across a range of stages of escaping abusive relationships [11, 2, 8].

This project engages in a technical analysis of Apple's 'Find My Friends' application to deepen understanding of how location data transmission occurs, and uses this information to construct a machine learning model that uses traffic analysis techniques to identify application use patterns consistent with abusive surveillance behaviors.

2 Background & Related Work

Despite the above-discussed vulnerabilities, the privacy-violating potential of CLS applications – which allows them to serve as an increasingly prevalent vector for IPV – is understudied in the literature. There is a notable gap in the literature concerning both the potential for CLS applications to contribute to abusive dynamics, and the extent to which the problems with CLS applications differ from similar privacy and security hurdles faced by IPV victim survivors (e.g., unauthorized account access by the abuser, installation of stalkerware, etc.) [6, 8].

Current studies concerned with the IPV implications of location monitoring applications focus on analyzing spyware and dual-use applications that can be used for spying purposes, with CLS applications – which in many cases require the victim to first share their location – not being considered a distinct category despite their unique properties [3]. Chatterjee et al. and other similar studies acknowledge CLS applications as a potential threat in IPV contexts, their analysis is broadly focuses on detecting spyware and dual-use applications on distribution platforms, rather than examining the flow of information between users.

Much of existing research that examines tech-facilitated abuse does so from a clinical standpoint, human-centered – the potential for technical strategies such as the one proposed in this project to help victims is underexplored [5].

Modern network traffic analysis methods use various kinds of packet metadata to train machine learning models capable of identifying suspicious or malicious behavior on the network [4]. Machine learning methods have been used to accurately classify browsers, applications, and user operating systems based on encrypted network traffic, illustrating the extensive information provided by network data [9].

However, the application of traffic classification to identify potential *UI-bound adversaries* using CLS applications has not been explored in existing literature. Existing research focuses on either using traffic classification to identify protocols related to specific applications, or to identify 'classes' of applications (i.e., malware) [4, 9, 13].

3 Approach

3.1 Objective

This project investigates proactive strategies for combating CLS application abuse, with the goal of building a binary machine learning classifier that can determine whether a user is having their location checked in a malicious capacity (i.e., whether they are being surveilled by an abuser).

More specifically, building upon the empirical experiments reported on in my Mid-Semester Project Report – which found that the 'Find My Friends' application server does in fact send packets to both the querying and queried device whenever location is checked – this project creates models that can identify those packets associated with a location check for .pcapng files captured from both the querying and queried devices.[1]

This project opts to focus specifically on Apple's Find My Friends application because other applications are likely to yield less promising results. Preliminary investigations indicated that that Google Maps's location sharing functionality updates each user's location every three minutes, and that Snapchat's Snap Maps feature refreshes location only when a user has the application open. If location-checked devices do not receive packets when checked, using network traffic generated by these applications to identify check frequency will not be possible.

3.2 Methodology

3.2.1 Packet Capture

Wireshark was used in conjunction with Apple's Remote Virtual Interface tool to capture packet traces for iOS devices that were checking the locations of others and having their locations checked [1].[2] Several .pcapng files were manually generated for model training and testing. All captures are approximate 10 real-time minutes in length.

- Scenario 1: 'Surveilled' device traffic, while not connected to the internet (all subsequent captures will be connected to the internet).
- Scenario 2: 'Surveilled' device traffic without having its location queried.
- Scenario 3: 'Surveilled' device traffic, while having its location 'precisely' queried once per minute.
- Scenario 4: 'Surveilled' device traffic, while having its location 'imprecisely' queried once per minute.
- Scenario 5: 'Surveilling' device traffic, while making a precise location query to the 'surveilled' device once per minute.
- Scenario 6: 'Surveilled' device traffic, while having its location 'precisely' queried twice per minute.
- Scenario 7: 'Surveilled' device traffic, while having its location 'precisely' queried once every other minute.

3.2.2 Dataset Annotation

A semi-supervised annotation script was written in Python to annotate the packets in most the .pcapng files described above (with the exceptions of scenarios 1 and 2) as either 'malicious' or 'non-malicious', using the pyshark Python library to parse packets in .pcapng files.[3] The script looks only at packets containing IP and TCP information with source or destination port '553' and source or destination IP prefix '17.57.144.0/24', since this is the port and IP prefix that has been established to be utilized by the Find My Friends protocol in refreshing device locations.

For each packet in the .pcapng file, source IP, destination IP, source port, destination port, packet length, TCP sequence number, header length, and TCP flags (SYN, ACK, FIN, RST, PSH, URG) are all extracted from the packet for potential use as model features. An 'inter-arrival time' value (i.e., the interval between this packet and the last packet) is also calculated by subtracting the current packet's timestamp from the previous packet's timestamp; a 'rolling average' value – the value length of the last 10 packets – is also calculated.

The script flags packets as malicious based on an 'interval' input parameter which defines the expected number of seconds between malicious queries (corresponding to the frequency with which the 'surveilling' device is querying in the scenarios discussed in the previous section), and a 'burst_interval' input parameter which determines for how

[1] I have chosen not to report on these empirical experiments so as to avoid redundancy and excessive length; see Mid-Semester Project Report for additional details on how packets are sent, and how this information flow was confirmed.

[2] Throughout all captures, the same iPhone 16 Pro was used as the 'surveilling' device, and the same iPhone XR was used as the 'surveilled' device.

[3] pyshark: https://github.com/KimiNewt/pyshark

many seconds after the current packet a packet can still be considered malicious before we must wait the interval again. This is to say, at 0 seconds, and then again every 'interval' seconds, any packets on the designated port and IP address are flagged as malicious for the next 'burst_interval' seconds.

3.2.3 Model Implementation and Testing

The sci-kit learn implementations of three traditional machine learning models were tested in this project: Logistic Regression, Random Forest, and Support Vector Classification (SVC) [10].These three models were chosen since they typically perform well for classification tasks and to ensure coverage of a wide berth of model types – Logistic Regression is simple and highly interpretable, Random Forest is robust to overfitting and typically does not require much tuning, and SVC is known to be effective for high-dimensional spaces.

Not all of the extracted packet features discussed in the previous section were ultimately used as model features. Formal, extensive feature engineering was beyond the scope of this project and remains a task for future research, but the feature engineering I was able to perform resulted in the selection of the following features: header length, inter-arrival time, rolling average length, TCP sequence number. Binary representations of TCP flag values were extensively tested as features, but did not seem to meaningfully improve or deteriorate model performance.

Some features were also explicitly excluded from consideration in order to avoid training the model on the wrong information. For instance, relative packet timestamp was generated for calculation of inter-arrival time, but not included as a feature so as to ensure that models did not train themselves to identify the interval between packets, which is static in the training file but not necessarily in real-world applications.

The training procedure was as follows: a training dataset as annotated above was separated into a training set and a validation set with an 80/20 split. For all models, a hyperparameter grid search was performed on the training set using sci-kit learn's GridSearchCV method. Figure 1 shows the grid search parameters for all three models. Recall scoring was used for all models, since all the models generally have higher precision than recall across the board, and prioritizing true positives is most rational given that the positive class is underrepresented in the dataset and it is the one we are interested in consistently finding. 5-fold cross-validation was applied for RandomForest and LogisticRegression, but 2-fold cross-validation was used for SVC due to device CPU limitations.[4] The best model for all three types was then chosen to have its performance validated with the validation set. A precision-recall curve (discussed in the following section) was also generated for each model to test the impact of changing the classification threshold on model performance.

x

Figure 1: Grid Search Parameters for Each Model

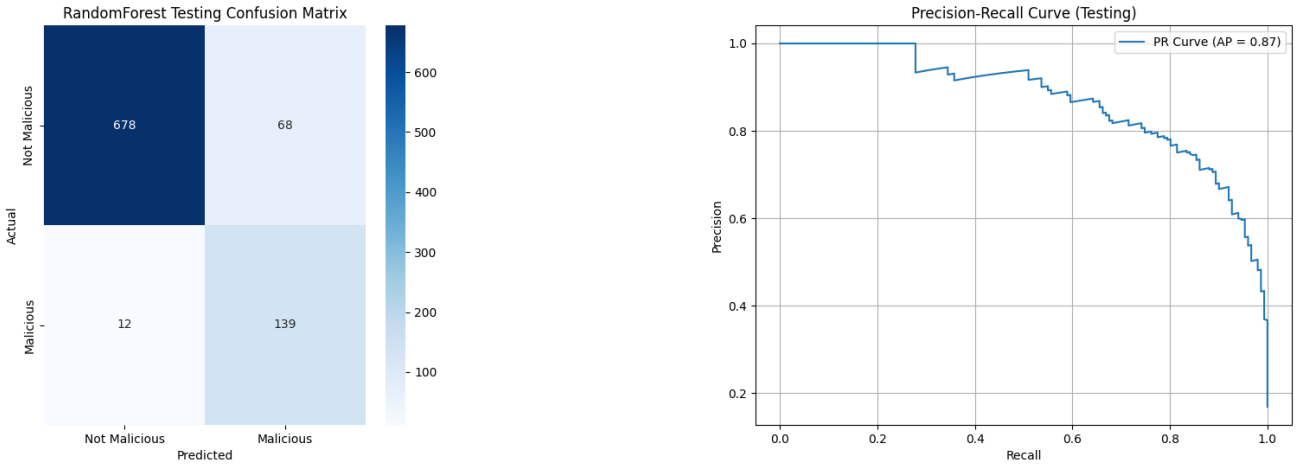
Model	**Hyperparameter**	**Values**
Random Forest	Number of Estimators	{50, 100, 200}
	Maximum Depth	{10, 20, None}
	Min Samples Split	{2, 5, 10}
	Class Weight	{'balanced', None}
Logistic Regression	C	{0.01, 0.1, 1, 10}
	Solver	{'liblinear', 'saga'}
	Class Weight	{'balanced', None}
Support Vector Classifier	C	{0.01, 0.1, 1, 10}
	Class Weight	{'balanced', None}

Models were trained with the annotated datasets from .pcapng files captured from scenarios 3, 4, and 5 (i.e., the scenarios with 1 minute intervals) as discussed in the previous section, while the scenario 6 and scenario 7 annotated datasets were used for an additional layer of testing after validation, to test the performance of the models on tasks with different location query intervals. For each model, the precision-recall curve was generated the testing dataset, to specifically evaluate the model's effectiveness on different surveillance intervals at different classification thresholds.

[4]Device CPU limitations also meant that different 'kernel' and 'gamma' values could not be tested for the SVC model, introducing an artificial constraint on its performance as compared to the other two tested model types.

4 Results

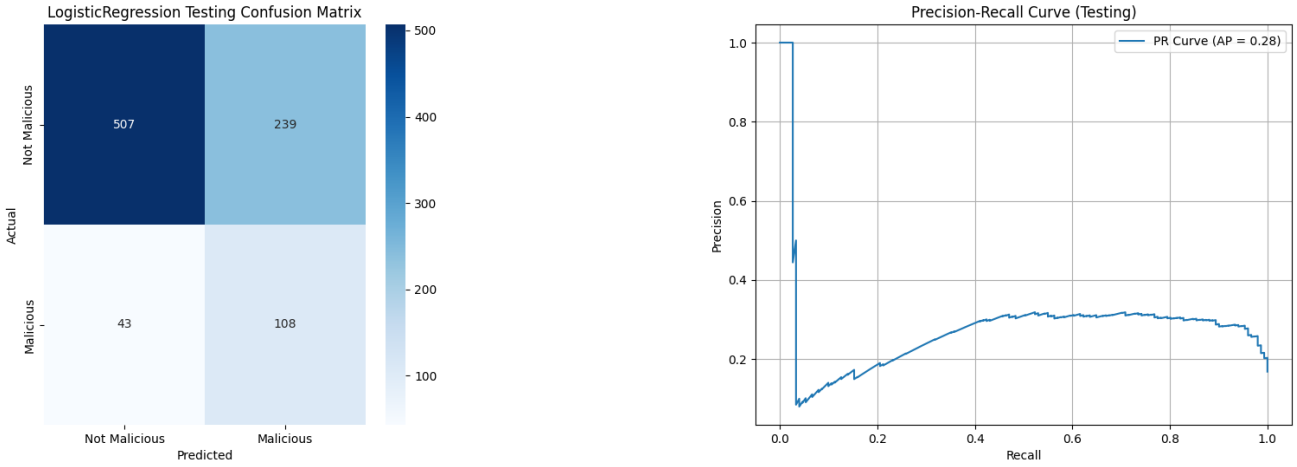
Though several permutations of training & testing scenario were tested throughout the evaluation process, I have opted to only include graphical representations of model performance for the best-performing model (i.e., for each model type, confusion matrices and precision-recall curves are only shown for the training/testing set combination), as shown in Figures 2, 3, and 4.



(a) Confusion Matrix

(b) Precision-Recall Curve

Figure 2: Random Forest model performance, trained on 'scenario 4' and tested on 'scenario 7'.



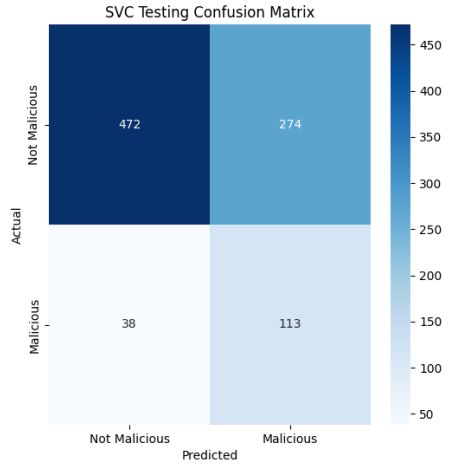
(a) Confusion Matrix

(b) Precision-Recall Curve

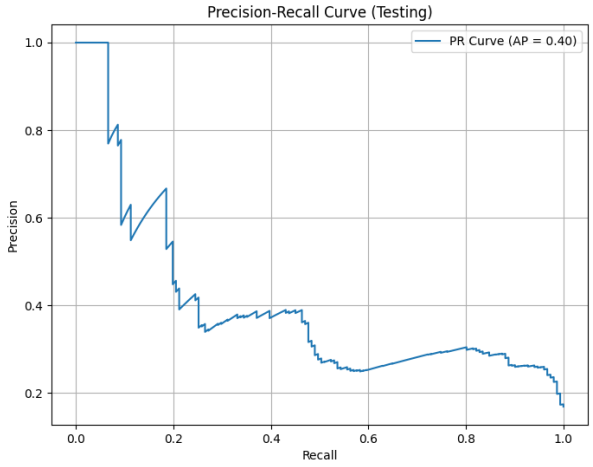
Figure 3: Logistic Regression model performance, trained on 'scenario 3' and tested on 'scenario 7'.

It is clear, from looking at the confusion matrices in Figures 2a, 3a, and 4a that the Random Forest model outperforms the Logistic Regression and SVC models – in particular, Random Forest is markedly less prone to false negative (i.e., mistakenly classifying a malicious packet as non-malicious) when compared to the other two models; Random Forest is also less prone to false positive, but the margin of difference is smaller in this case.

Results observed in the precision-recall curves (Figures 2b, 3b, and 4b) reflect a similar pattern. With an average precision (AP) value of 0.87, the Random Forest precision-recall curve once again illustrates that this is the best-performing model, with relatively high precision and recall scores at all threshold values. In contrast, Logistic Regression has an AP value of 0.28, and SVC a value of 0.40 – though both of these values are lower than the Random Forest value, SVC's higher value and more natural-looking curve indicate that, over varying threshold values, the SVC model performs better than the Logistic Regression model – this is an interesting finding, as the



(a) Confusion Matrix



(b) Precision-Recall Curve

Figure 4: Support Vector Classification model performance, trained on 'scenario 3' and tested on 'scenario 7'.

Logistic Regression model's confusion matrix is slightly better than the SVC model's, and provides reason to further explore tuning of these and other models for this classification task.

5 Conclusions

5.1 Findings

The findings presented in this project provide reason to believe that construction of a model that uses packet header features to identify packets associated with the Find My Friends application is feasible. As expected, of the tested models Random Forest outperforms other model types by a wide margin. Nevertheless, both Logistic Regression and especially SVC both perform well enough to provide reason to believe that exploration of other model types to this classification problem has potential to be fruitful.

The confusion matrices and PR curves presented in the previous section show that, with additional tuning, any of the presented models could perform quite well for the classification task in question. Even with limited feature engineering, all of the models are quite good at correctly classifying malicious packets as malicious – this, then, seems to indicate that it is in fact viable to use network traffic analysis techniques to classify packets that are (or are not) associated with privacy-violating accesses of a CLS application. There is, of course, significant room for improvement – particularly in reducing the rate of false positives across all models.

5.2 Limitations & Future Research

The findings presented in this project, while preliminary, are positive, and provide reason to believe in the suitability of machine learning for this classification task. Nevertheless, there are methodological limitations that have constrained this project and are worth discussing as avenues to build upon this project in the future.

The most apparent limitation of this project the annotation script used for dataset labeling, which is rudimentary – it relies on a pre-determined static interval to label packets as malicious, and there is no guarantee that the correct packets on the Apple Push Notification Service server are being flagged by the script (i.e., the server is used by other Apple applications in addition to Find My Friends, and it can be difficult for a script to parse which packets are the 'malicious' ones). The annotation script relies on the assumption that the first packet on the IP/port combination will be malicious (since, in the scenarios, a location check was always performed at the very beginning of the capture) and that packets observed after the interval has passed are malicious as well, but this assumption may not be robust.

Additionally, testing the performance of other potentially fruitful machine learning models – such as k-Nearest Neighbors, Naive Bayes, and XGBoost – is important but will remain a topic to be explored by future research due to research and time constraints, as will the application of large language models to this classification task [10]. Furthermore, SVC testing was significantly limited by device CPU constraints – additional kernel and gamma value combinations should be tested in the future to more accurately assess SVC's performance for this classification task.

More robust feature engineering should also be considered as an avenue for future research, as extensive feature engineering was beyond the scope of this project due to time and resource constraints.

It is also worth noting that this project has limited itself to analyzing packet features available in headers. Application of deep packet inspection techniques, which could be used to extract payload information from packets — while more time- and resource-intensive – would almost certainly result in better models. Nevertheless, this remains a topic for future research.

Finally, this project has not engaged with CLS application protocols beyond that of Find My Friends – therefore, I am not equipped to make a judgment on whether the techniques I have explored can be used to identify malicious behavior on other CLS applications. Future research should be sure to engage with this question, as understanding the landscape of these applications will be crucial for understanding the scalable applicability of the technique this project has explored.

References

[1] Recording a Packet Trace.

[2] Louis Bailey, Joanne Hulley, Tim Gomersall, Gill Kirkman, Graham Gibbs, and Adele D. Jones. The Networking of Abuse: Intimate Partner Violence and the Use of Social Technologies. *Criminal Justice and Behavior*, 51(2):266–285, February 2024. Publisher: SAGE Publications Inc.

[3] Rahul Chatterjee, Periwinkle Doerfler, Hadas Orgad, Sam Havron, Jackeline Palmer, Diana Freed, Karen Levy, Nicola Dell, Damon McCoy, and Thomas Ristenpart. The Spyware Used in Intimate Partner Violence. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 441–458, May 2018. ISSN: 2375-1207.

[4] Michael J. de Lucia and Chase Cotton. Detection of Encrypted Malicious Network Traffic using Machine Learning. In *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, pages 1–6, November 2019. ISSN: 2155-7586.

[5] Diana Freed, Sam Havron, Emily Tseng, Andrea Gallardo, Rahul Chatterjee, Thomas Ristenpart, and Nicola Dell. "Is my phone hacked?" Analyzing Clinical Computer Security Interventions with Survivors of Intimate Partner Violence. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–24, November 2019.

[6] Diana Freed, Jackeline Palmer, Diana Minchala, Karen Levy, Thomas Ristenpart, and Nicola Dell. "A Stalker's Paradise": How Intimate Partner Abusers Exploit Technology. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, Montreal QC Canada, April 2018. ACM.

[7] Andrew Laningham. Are Location Sharing Features More Than a Convenient Tool?, August 2022.

[8] Tara Matthews, Kathleen O'Leary, Anna Turner, Manya Sleeper, Jill Palzkill Woelfer, Martin Shelton, Cori Manthorne, Elizabeth F. Churchill, and Sunny Consolvo. Stories from Survivors: Privacy & Security Practices when Coping with Intimate Partner Abuse. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2189–2201, Denver Colorado USA, May 2017. ACM.

[9] Jonathan Muehlstein, Yehonatan Zion, Maor Bahumi, Itay Kirshenboim, Ran Dubin, Amit Dvir, and Ofir Pele. Analyzing HTTPS encrypted traffic to identify user's operating system, browser and application. In *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6, January 2017. ISSN: 2331-9860.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] Svenja Senkans, Troy E. McEwan, and James R. P. Ogloff. Assessing the Link Between Intimate Partner Violence and Postrelationship Stalking: A Gender-Inclusive Study. *Journal of Interpersonal Violence*, 36(1-2):NP772–NP802, January 2021. Publisher: SAGE Publications Inc.

[12] Sharon G. Smith, Kathleen C. Basile, Leah K. Gilbert, Melissa T. Merrick, Nimesh Patel, Margie Walling, and Anurag Jain. National Intimate Partner and Sexual Violence Survey (NISVS) : 2010-2012 state report. 2017.

[13] Sadegh Torabi, Amine Boukhtouta, Chadi Assi, and Mourad Debbabi. Detecting Internet Abuse by Analyzing Passive DNS Traffic: A Survey of Implemented Systems. *IEEE Communications Surveys & Tutorials*, 20(4):3389–3415, 2018. Conference Name: IEEE Communications Surveys & Tutorials.