Location-leaking through Network Traffic in Mobile Augmented Reality Applications

Gabriel Meyer-Lee[‡], Jiacheng Shang[†] and Jie Wu[†]
[†]Center for Network Computing, Temple University, Philadelphia, PA, 19121
[‡]Engineering Department, Swarthmore College, Swarthmore, PA, 19081
gmeyerlee@gmail.com

***Abstract*—Mobile Augmented Reality (AR) applications allow the user to interact with virtual objects positioned within the real world via a smart phone, tablet or smart glasses. As the popularity of these applications grows, recent researchers have identified several security and privacy issues pertaining to the collection and storage of sensitive data from device sensors. Location-based AR applications typically not only collect user location data, but transmit it to a remote server in order to download nearby virtual content. In this paper we show that the pattern of network traffic generated by this process alone can be used to infer the user's location. We demonstrate a side-channel attack against a widely available Mobile AR application inspired by Website Fingerprinting methods. Through the strategic placement of virtual content and prerecording of the network traffic produced by interacting with this content, we are able to identify the location of a user within the target area with an accuracy of 94%.This finding reveals a previously unexplored vulnerability in the implementation of Mobile AR applications and we offer several recommendations to mitigate this threat.**

***Index Terms*—Augmented Reality, mobile applications, data privacy.**

I. Introduction

Augmented Reality (AR) describes a technology which convincingly positions virtual objects within the real world and allows the user to interact with them. Currently, AR is dominated by mobile AR, which encompasses all AR applications displayed through mobile devices, including specially designed AR glasses as well as smart phones and other handheld devices[2].

The AR industry is expected to reach $85-$90 billion within 5 years, and already includes technologies built for education, entertainment, tourism, and virtual modeling. Games are currently the most popular application for AR and will likely remain so for some time[5]. The most popular of these games is Niantic's Pokemon Go.

Pokemon Go is not only an example of mobile AR, but geolocation-based mobile AR. This particular form of AR does not only attempt to position virtual objects within a real environment, but also grants the virtual objects an actual geographic location. In its most complete form, this type of AR can be used to create a persistent, entirely virtual layer to the world, which exists on top of the physical world.

Mobile AR, especially location-based mobile AR, is becoming more powerful by the day. The rapid expansion of cloud-computing capabilities, computer vision software, and the steady improvement of network connections, both increases in public WiFi and the impending mobile 5G, continually support greater and greater possibilities for these apps.

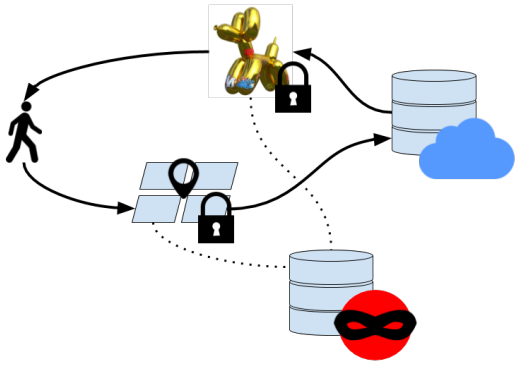

Figure 1. Diagram depicting the attack model for a network-traffic-analysis-based attack on a location-based Mobile Augmented Reality application

The AR industry is very young and so is the academic study of AR. Much of the current body of literature is very theoretical or concerned with developing the necessary technology to support AR or VR. Considerably less literature deals with the ramifications of AR technology actually existing. In particular, only recently has significant work been published addressing the security of AR technology. The heavy use most current AR apps make of device sensors means that AR applications represent a significant concentration of sensitive data.

Early discussion of AR security/privacy did occur 2000's Designing Augmented Reality Environments (DARE) conference. Mackay [11] discusses security and privacy concerns, but with regard to augmented paper, which is no longer considered an AR technology. On the other hand, Friedman and H. Kahn Jr [7] are remarkably prescient in their assessment of the risks of AR. Although their paper is short, they identify several key concerns, such as the increased difficulty of maintaining privacy while using AR technology and the possibility of deceiving or harming a user through attacks on the output content, which would be more thoroughly discussed by Roesner, Kohno, and Molnar [15] 14 years later. This work, while very thorough and much more reflective of the current AR paradigm than previous work, remains very theoretical. On one hand, their discussion of the challenges of interactions between independent AR systems is very forward-

978-1-5386-6808-5/18/$31.00 2018 IEEE

thinking and will hopefully influence the future development of AR software. On the other hand, the very-real current risks associated with the heavy sensor use and data-managing of AR mobile apps are given equal weight to more speculative concerns, like the need for a consistent shortcut to return to reality.

Guzman, Thilakarathna, and Seneviratne [8] show that up until 2014, less than 1% of papers on AR/VR discussed security and privacy concerns. Many of the privacy/security concerns addressed here were already brought up by Roesner, Kohno, and Molnar [15], however this more recent survey provides a comprehensive technical overview of the current work in regards to these concerns. This includes work in input sanitization, privacy-preserving data aggregation, and physiological-based user authentication, among other fields. The authors of this survey identify one particular area of defensive AR/MR-related research which is underdeveloped: data protection. In addition to noting that very little work had been done on data protection systems which targeted AR, the authors pointed out that a high-concentration of high-use AR technologies could easily put a strain on current networking infrastructure.

However, the heavy use of the network infrastructure itself poses a risk for data protection. AR applications have a natural vulnerability to side-channel attacks, given their particular reliance on taking in sensory data and communicating with a server to process that data. Specifically, AR apps display a vulnerability to network traffic analysis, a widely researched form of side-channel attack. This vulnerability has not yet been experimentally explored and the extent of the threat posed by side-channel attacks on AR applications is largely unknown.

In this paper we demonstrate conclusively that patterns in the network traffic data of an Mobile AR application can leak a user's physical location. We illustrate this threat through an example attack using data recorded from WallaMe, a publicly available AR application. The available virtual content observed creates a fixed set of distinct locations. Our attack uses a Convolutional Neural Network to classify the network traffic as belonging to one of these locations with an accuracy of approximately 94% demonstrating, for the first time, that physical location data is recoverable from Mobile AR network traffic patterns.

II. Related Work

a) Network Traffic Analysis: The literature on network traffic analysis reveals the extent to which patterns in network traffic can expose the information of unsuspecting users. Chen et al. [3] display the potential risks of side-channel attacks and point out that these vulnerabilities are the byproduct of common web design practices. These three vulnerabilities the authors identify, low entropy inputs, stateful communications, and significant traffic distinction are all also vulnerabilities of contemporary AR applications.

Website Fingerprinting (WF) [1] is an attack based on network traffic analysis in which the attacker reveals the identity of the web pages the victim is visiting, even if the victim is using an anonymity network like Tor. This is typically accomplished by extracting features from the observable network traffic and matching the user's traffic to prerecorded patterns corresponding to specific websites. Although there are many sophisticated algorithms to defend against WF, they typically increase either delay or bandwidth, which are already issues for AR apps.

Fingerprinting holds an even greater danger for AR. While WF can analyze web browser traffic to reveal the web sites you've visited, AR Fingerprinting could analyze AR browser traffic to reveal the actual physical locations you've visited.

b) Mobile AR Privacy/Security: As outlined above, most of the existing literature on AR privacy/security focuses on issues with the security or privacy of the displayed content itself. These dangers can include injecting malicious content into the display, comprising display privacy via shoulder surfing, or compromising the privacy of oneself or others via the camera feed. There is very little work addressing network attacks in a Mobile AR domain. A notable blog post by Colceriu [4] released shortly after Pokemon Go's jump in popularity demonstrated some network vulnerabilities of Pokemon Go. Ren et al. [13] demonstrate a general network traffic analysis attack against mobile apps. This is not particular to AR applications but mobile AR applications are just as capable of leaking personally identifiable information as any other AR application. McPherson, Jana, and Shmatikov [12] investigate several possible security concerns with Mobile AR browsers, including a brief discussion of potential network based attacks.

III. Overview of the Attack

The demonstrative attack utilizes a Mobile AR app, WallaMe, available through the Google Play store and iTunes App store as an platform for recording network traffic data. This app is essentially designed to be a digital graffiti platform, allowing users to draw digital art on physical walls and share their "walls" with their friends. We chose this app as a platform for the attack because it allows its users to place virtual content of a size of their choice at any location. This allows an attacker to deploy content in a specific pattern to facilitate localization of the victim. Additionally, the content is visible within a radius approximately equal to a single city block, allowing us to create patterns of manageable sizes for the purposes of our experiment.

A. Attacker Model

In this attack scenario, the attacker is able to passively monitor the victim's network traffic as they move through an area of roughly eight city blocks. This area presents a challenge not normally encountered in network traffic analysis: the moving victim will likely not be continuously connected to a single wireless access point. We have identified two possible attack models that could support monitoring of a moving victim, network sniffing at a university or urban center and a spyware application installed on the victim's device.

a) Network Sniffing: Network sniffing is the traditional model for an attack based on network traffic analysis. In this context, the mobile nature of the victim would require them to be moving within a university campus or urban center which was entirely blanketed by wireless networks. If the attacker has compromised the security of a university campus' wireless network, monitoring network traffic would be straightforward, but in order to utilize network sniffing to monitor the network traffic the attacker would need to deploy sniffing devices throughout the area to cover the many available access points as demonstrated by Kotz and Essien [10]. The attacker would also need to implement a highly efficient method to isolate and aggregate the victim's network traffic. This method would allow the attacker to view a highly detailed record of the victim's traffic and also utilize information about the routing of their network traffic to assist in localization.

b) Spyware on Device: Without WiFi coverage over the area of the attack, this attack would require spyware installed on the victim's mobile phone, with the most practical form being an over-permissioned application. This is a very practical attack vector as Felt et al. [6] showed that a majority of Android users do not pay attention to or comprehend Android permissions. The necessary Android permission is innocuously termed "usage access," which is less likely to raise suspicion in users than directly requesting location access. In addition, the Android permissions are not fine-grained enough to support users limiting the access of an application to the data usage of specific applications. This means that any application which is granted usage access, for example, an application to monitor cellular network data usage for users on a limited-data cellular plan, is able to access the data necessary to perform this attack. One advantage of this attack implementation is that it will work effectively for both WiFi and cellular network data usage. The main drawback of this method is that it does not allow packet level analysis of the network traffic. The attacker will only be able to monitor the quantity of data downloaded and uploaded by the target AR application over time, unable to distinguish individual packets or the source API of the downloaded data.

Our attack is based on the latter of the two possible described methods, as we believe it to represent a more realistic implementation of the attack. Over-permissioned mobile applications are a widely recognized current security concern and represent a more practical implementation of the attack than network sniffing. The network sniffing method does represent a more effective method of localizing a security-aware victim, but we judge the overall threat of the spyware method to be greater, as the setup costs of the network sniffing method are prohibitive of mass implementation. The "usage" permission on Android allows our over-permissioned app to track the cumulative data usage of WallaMe (or any other app on the phone). We sample this cumulative data usage at 1 Hz producing a signal over time of download and upload data. We utilize the patterns found in this download data alone to locate the user.

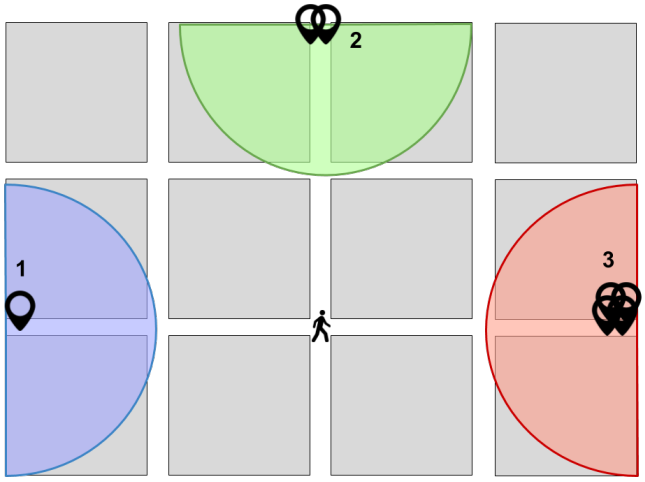

Figure 2. Diagram depicting Scenario 1 virtual object deployment

B. Victim Model

Additionally, in this attack scenario, our victim is actively searching for virtual content. As soon as some virtual content (a wall) on WallaMe becomes visible to them, they will select each available wall once in any order. This is necessary as the app does not download the full image until the wall has been selected. This behavior is fully within normal user behavior for the app, although real user behavior is unlikely to be as consistent. This model of user behavior is chosen in order to support a generalization to AR applications with automatically download virtual content within a set radius.

C. Scenario 1: Non Overlapping, Identical content

The layout of the attack consists of 7 identical images, each placed on its own wall, one placed two blocks to the west of the victim's starting location, two placed two blocks north, and four placed two blocks east. This pattern will allow the attacker to know once the victim has traveled one block in any of these three directions. This layout supports an attack in which the attacker is intending to track a moving victim through an urban environment. Assuming that the victim is traveling forward only, by deploying the virtual content in the three available directions for the victim, the attacker is able to infer the direction the victim has moved and redeploy the virtual content to continue tracking. The locations representing these three possible directions are labeled in Figure 2.

For the purposes of collecting data, our tests involved walking one block in each of these three directions and then returning. This is depicted in Figure 3 which depicts a single data collection trial, with the locations within the visibility radii of the three image groups labeled. The location of the network traffic data was labeled manually. GPS locations coinciding with the network traffic data were recorded during the experiment for the purposes of confirming the validity of the labels.

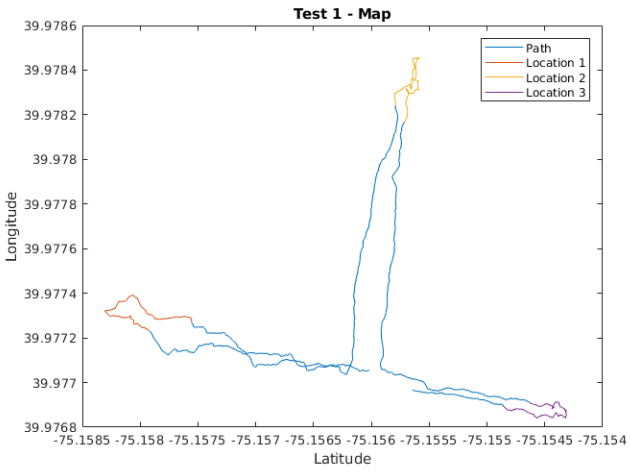

Figure 3. Data collection path with detectable locations labeled

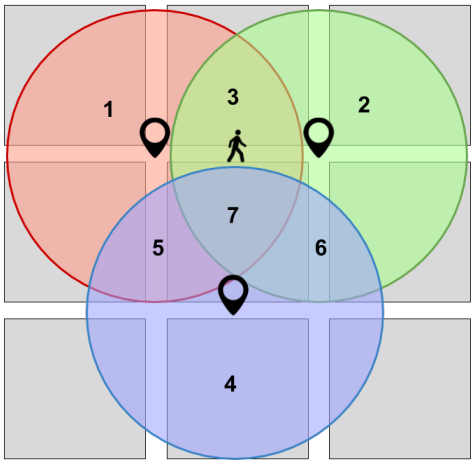

Figure 4. Diagram depicting Scenario 2 virtual object deployment

D. Scenario 2: Overlapping, distinct content

In the second attack scenario the attacker deploys 3 separate images instead of 7 duplicates of the same image. These images are specially chosen to have distinct download sizes within the WallaMe app. These images are deployed at an approximately one block distance from each other to create overlap between the visible areas of the individual images. This creates a total of 7 different distinguishable areas within a roughly two block radius of the epicenter. These 7 areas are shown labeled in Figure 4.

This attack is focused solely on localizing the victim within an urban area, although it also has the capability to track the victim's location. This scenario is not as heavily dependent on the prescribed victim behavior model. Because the distinct areas are each defined by a single uniquely-sized wall within WallaMe, whenever the victim opens a wall, there location can be tied to the visibility radius of that wall. We will, however, still assume for the purposes of our analysis that the victim

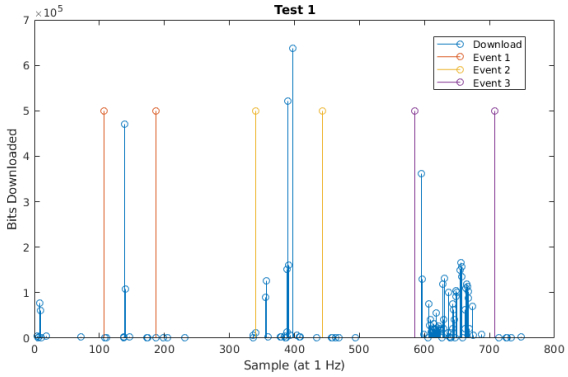

Figure 5. Data download over time of sample trial

is following the behavior model, as this allows us to utilize the victim's opening of multiple walls within the overlap of the visibility radii to establish a finer location for the victim, as well as supporting the generalization of this method to AR application which automatically download nearby virtual content.

As with Scenario 1, we manually recorded experimental data by visiting each of these possible locations in turn, although unlike Scenario 1, multiple different traversals of the 7 locations were performed and recorded.

IV. Analysis and Results

Due to networks conditions varying through time and location, the downloads of the visible images did not form perfectly regular patterns. Each image requires a consistent amount of data to be downloaded, but this data is not downloaded within a consistent period creating significant variation in the network patterns. Our experiment utilizes only the quantity of data downloaded over time by the user's WallaMe application as the user's interactions with virtual content are characterized most explicitly by spikes in download data.

Figure 5 shows how the download events are split into separate components of unpredictable size. The data describing the downloads of the single, double, and quadruple images are labeled as Events 1, 2, and 3. In order to verify the download sizes followed the expected pattern, we use a a windowing technique to aggregate the download samples.

Figure 6 depicts the resulting signal after convolution with a Hamming Window. This operation consistently reveals three discrete downloads of increasing size, although the local maxima visible as a result of this convolution operation are not at consistent heights. As a result, this windowing operation is not enough on its own to support the extraction of location from the recorded network traffic data. The variation present suggests that a machine learning algorithm would be an effective method for inferring the user's location.

A. Machine Learning Algorithm Requirements

In order to design an algorithm to identify the user's location based on the recorded network traffic data, several constraints

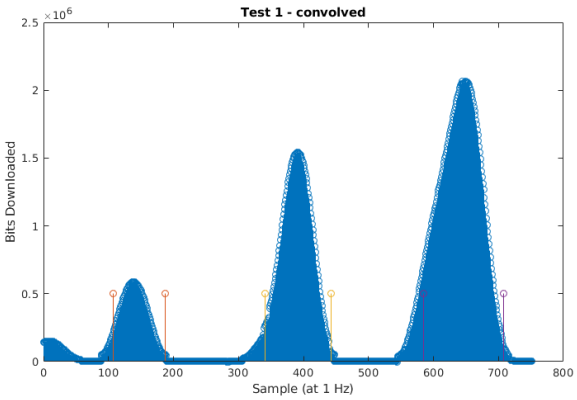

Figure 6. Windowed data download over time of sample trial

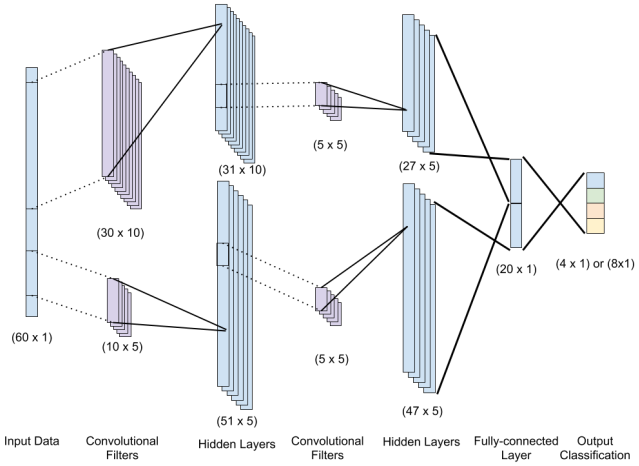

Figure 7. CNN structure

are imposed to assure the viability of this method in a real world attack.

- The algorithm should be designed so that it could provide near real time location updates in an online attack. This meant that a fixed size amount of network data had to be specified as the input to the algorithm.
- The algorithm must not rely on a sequential pattern of input information to localize the user. While this contextual information could certainly be used to improve localization accuracy in an attack, the use of a single, preset sequence of locations in Scenario 1 meant that the user's location could be much easier to infer from its position in the sequence than from the network traffic as intended. This meant that structures with some sort of memory of their input, like recurrent neural networks, would be unsuitable for this task.

B. Neural Network

Machine learning algorithms are widely used in the field of network traffic analysis for Website Fingerprinting tasks. Past state-of-the-art WF algorithms have utilized Support Vector Machines (SVMs), k-Nearest Neighbors (k-NN), and random forests. Only recently has the use of neural networks been demonstrated for WF [14]. The traditional methods are largely unsuitable for the data we have collected as they rely on manually engineered feature vectors extracted from traces. Our lack of proper traces prevents us from implementing these methods, as the continuous nature of our data complicates feature extraction. Features are typically based on incoming and outgoing packet lengths and quantities, information which cannot be recovered from our data.

1) Convolutional Neural Network: With the above constraints in mind, we choose to use a convolutional neural network (CNN) as our classifier. The design of this neural network is inspired by the use of CNNs for image classification tasks. CNNs have been shown to be capable of learning translation-invariant representations of images [9]. This property is desirable for our task as the download event for the virtual content can occur anywhere within our fixed length frame. The input to the CNN is a one minute long frame of WallaMe download data. An appropriate dataset is synthesized by using a moving 60 sample frame with a stride of 1 on the input data, labeling the frame with the appropriate location if the frame included a manually labeled download event. In addition to the translation-invariance property, we decided to use a CNN because of the success of the above windowing process in creating clear groupings in the recorded data. The 1D convolution operation, with the appropriate weights, is capable of creating a very similar output to the windowing process. Another advantage of CNNs is that they have less-trainable parameters than fully-connected networks and so can preform better with less training data.

2) Network Design: The neural network structure utilizes two parallel networks which are ultimately joined to before the final layer to perform a single classification task. This structure is partially inspired by the Inception module [16], which features multiple parallel convolutional layers with different filter sizes to allow the network to optimize the filter size itself. The first layer features independent 1D convolutions, one with a window size of 30 samples and 10 channels and one with a window size of 10 samples and 5 channels. We use a large filter size to allow the reproduction of the output shown in Figure 6 while the smaller size allows the network to observe dense spikes in download traffic. These layers are each followed by their own additional convolutional layer with a window size of 5 and 5 channels to allow the observation of patterns larger than either of the previous window sizes. These secondary convolutional layers are then each followed by their own fully-connected layer, with 10 output units each. These layers are concatenated and followed by a final fully-connected layer, whose output reflects the either 4 or 8 possible location classes. This structure is shown in Figure 7.

3) Training the CNN: Each of the recorded traversals of the possible detectable locations are split into numerous 60-sample frames as described above. For each traversal, this produces approximately 1000 null samples (reflecting the user not

entering any of the locations) and 50 samples corresponding to each location. This uneven representation in the training data, if uncorrected, would lead to a majority of the error in our network's predictions coinciding with the download events. To remedy this, we perform data augmentation [17], copying the current data for each location and applying Gaussian noise until there are roughly equivalent amounts of samples corresponding to each location and the null location.

10 traversals are initially used as training data and 3 traversals as a validation dataset in order to experimentally determine the validity of the presented network architecture. Following this, these 13 traversals are combined as training data and 4 yet-unseen traversals are used as test data to measure the accuracy of the network. The training data is shuffled and split into mini-batches of 32 samples. The classifier is trained on this dataset for 200 epochs. During the validation phase, the accuracy on the validation data set is recorded after each epoch. During the testing phase, the overall accuracy is only recorded once for the all of the test traversals together. The CNN's weights are adjusted using the Adam optimizer with a learning rate of .001 for Scenario 1 and a learning rate of .0002 for Scenario 2, which is lower to account for the greater quantity of data and output classes.

C. Results

Across the 4676 frames in the Scenario 1 test dataset, our CNN classifies 93.8% accurately and of the 4268 frames in the Scenario 2 test dataset our model classifies 87.6% correctly. These errors, however, are notably not evenly distributed throughout the predicted labels. By examining the distribution of error, we can show that our classifier was, in fact, more accurate than these figures suggest.

1) Scenario 1 Error Analysis: As shown in the Figure 8, the most accurately predicted classes are the null location and location 1. This is as expected, as the the null location is generally represented by very little network traffic and the first location is indicated by a single isolated download. Locations 2 and 3 are trickier as they each represent several sequential downloads. The mutual confusion is likely due to the limitations of the network, which cause the spikes in download data to be similar in size regardless of the total amount being downloaded. The confusion between both locations 2 and 3 and the null location is likely due to the gaps present in the training data for those locations. As locations 2 and 3 are characterized by individuals downloads separated by varying amounts of time, there are occasional gaps where the data is labeled as belonging to location 2 or location 3 but consists of very little network traffic. Copying the data corresponding to these locations has increased the representation of these gaps in the training data.

2) Scenario 2 Error Analysis: Unfortunately, in the Scenario 2 test predictions one class is not most likely to be predicted correctly. Location 5 download data is most likely to be predicted as belonging to location 3, as shown in Figure 9. This confusion is understandable, as location 5 and location 3 are similarly composed combinations of two downloads,

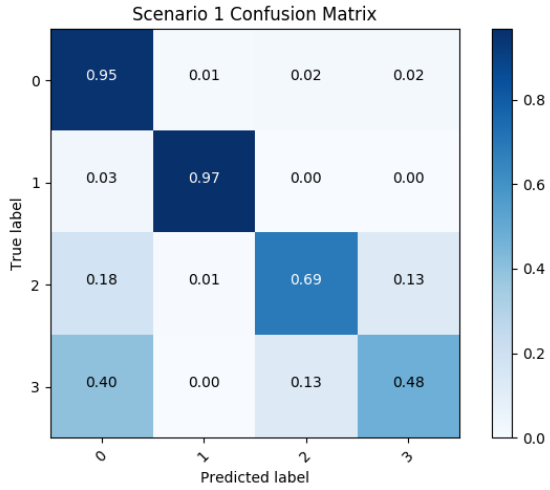

Figure 8. Normalized confusion matrix showing predictions on Scenario 1

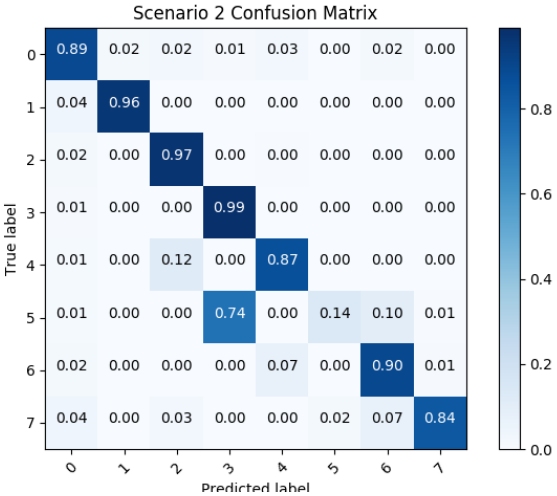

Figure 9. Normalized confusion Matrix showing predictions on Scenario 2

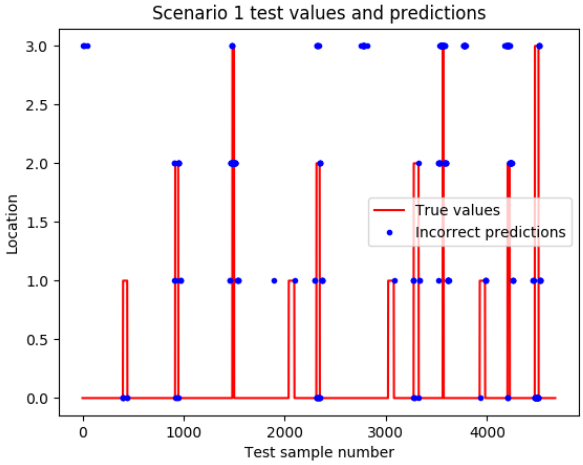

Figure 10. Scenario 1 test error plotted against true values

including the image at location 1 along with the image at location 4 or 2 respectively. The patterns representing these downloads are likely to be similar, as the confusion shown between locations 2 and 4 suggests that the download size of the location 4 image is not as distinctly larger than that of location 2 as would be ideal. Similarly, location 6 is occasionally confused with the larger of its two components images, the one found at location 4, and location 7 was occasionally confused with location 6, with which it shares its two largest components (the images at locations 4 and 2).

3) Moving Frame Error: There is a specific pattern in the observed error of the predictions on the test set which holds across both experimental scenarios and can actually ascribed to valid, robust performance by the CNN. That pattern consists of false positives, which within the context of this experiment we define as null location frames incorrectly classified as belonging to one of the detectable locations. Specifically, these false positives surround correctly predicted locations. The predicted value of the false positive generally increases towards the value of the nearby true positive the closer the frame is to the location in time of the true positive. This pattern can be viewed in the Scenario 1 test data in Figure 10, where it can be clearly seen that every occurrence in the test dataset of download data corresponding to locations 2 or 3 is surrounded by several instances of a null location frame being incorrectly predicted as location 1.

We call this error "moving frame error" as it is the natural result of trying to detect a combination of sequential downloads using a moving frame. Our labeling method only attaches a nonzero location label to a frame if it fully covers or is fully covered by the hand labeled region of the download event. A frame which partially covers a download event may include only a part of the sequence of downloads. In the case of Scenario 1, a frame near the location 2 labeled area may include one of the two sequential downloads which is actually exactly equivalent to the download data representing location 1. In this case, the CNN correctly identifies the presence of a single download and assigns the frame a label of 1, although the label for this frame will be 0, as the frame has not yet reached the double download which characterizes location 2. It is also possible that the frame may include enough of the double download to correctly identify that the user is in location 2, but may not cover the manually labeled region of location 2 download event sufficiently enough to be labeled as location 2.

This moving frame error, therefore does not necessarily constitute an actual error of the neural network, rather an error enforced by the arbitrary strictness of the labeling scheme. The moving frame error actually contains useful information itself. Figure 10 shows multiple instances of a null location being falsely identified as location 3. Unlike the actual instances of location 3, none of these false identifications show moving frame error, and so can easily be filtered out on this basis.

The moving frame error is actually responsible for a majority of the observed error. This is not reflected in the confusion matrices simply because the vast majority of the frames in the test dataset represent correctly identified null samples, so the influence of false positives appears to be relatively low. By accepting that the moving frame error actually represents correct operation of the neural network and is useful in the localization task, the effective accuracy of our predictive CNN is actually much higher than the raw accuracy, as shown in Table 1.

	Scenario 1	Scenario 2
Raw accuracy	93.8%	87.6%
Error due to moving frame	56.3%	58.2%
Accuracy excl moving frame	97.3%	94.8%

Table 1: Adjusting accuracy to exclude moving frame error.

V. Mitigation

Defenses against this attack could be implemented by the developer of the application, the AR technology itself, or even by the user.

The only real option for the user to mitigate such an attack would require the user to have knowledge of the mechanics of the attack and avoid the behavior targeted by the attack, which, in our case, is opening all of the available walls. Mobile AR apps typically have several features which are not location based and produce network traffic, such as interfaces that allow sharing with social media. Frequent use of these features may diminish the viability of recovering the location from network traffic.

The developers of the AR application have more and more powerful options for mitigating this form of attack than the users. Packet-padding is a classic defense against network traffic analysis attacks. If the app developer is able to pad the location-based downloads to a fixed size, this could prevent the inference of location from network traffic. This method may create too much overhead for use in an AR application so an alternative defense is asynchronous downloads of nearby digital objects. Most digital content tied to a specific location should be downloaded prior to when it must actually be served up to the user. Mobile AR apps typically have a set radius of in which content is visible to the user. The suggested method is to define a much larger area in which virtual content is downloaded probabilistically not en masse. The probability of a piece of content being downloaded should be dependent on its distance from the user, the user's heading, its size, and current network conditions. This method will allow AR apps comparable or improved performance, while obscuring the users location through the random downloads.

The developer of the AR technology has the ability to specify the access to data for applications running on the device. Well designed access rules, or user permissions, could prevent this attack being implemented via spyware installed on the device.

VI. Future Work

This paper merely establishes the possibility of this attack as a proof-of-concept. The level of threat posed by such an attack is not thoroughly assessed. This assessment would require

larger scale experimentation to determine the scalability of this attack model. Additionally, the potential for location-leaking in a more coarse sense must be considered. The potential of Mobile AR application network traffic to reveal which city or neighborhood the victim could be considered through this large scale experimentation.

A passive form of this attack could be developed for AR apps that do not allow users to place their own geolocated virtual content of arbitrary sizes. This attack could record data downloaded while traveling to different physical locations within the app to learn a fingerprinting algorithm which could tie the network traffic produced by existing location-based digital content to the location of that content. This could be used to attack a wider range of AR applications, including Pokemon Go.

VII. Conclusion

While our localization method for WallaMe comes with several qualifications, the 93.8% raw localization accuracy and 97.3% adjusted location accuracy achieved for Scenario 1 clearly that network traffic analysis techniques can be used to tie network traffic patterns to physical locations. The 94.8% adjusted accuracy found for Scenario 2 shows that this method can absolutely be used to establish automated localization of a WallaMe user within a fixed area. As AR technology spreads and increases in popularity, this threat that this attack poses will only increase. This threat can be mitigated, however, through responsible design by developers of location-based AR applications. These developer should build in measures to decorrelate the network traffic from the physical location of their users, such as download padding or probabilistic downloads of nearby AR content. Developers of AR technology must consider the network traffic of the device to be sensitive data, in addition to the personally identifiable information communicated via that network, and provide reasonable restrictions to the access of this data.

Acknowledgments

This research was supported in part by NSF grants CNS 1757533, CNS 1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, and IIP 1439672.

References

[1] Xiang Cai et al. “Touching from a Distance: Website Fingerprinting Attacks and Defenses”. In: *Proc. of the 2012 ACM Conference on Computer and Communications Security*. CCS. 2012, pp. 605–616.

[2] D. Chatzopoulos et al. “Mobile Augmented Reality Survey: From Where We Are to Where We Go”. In: *IEEE Access* 5 (2017), pp. 6917–6950.

[3] S. Chen et al. “Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow”. In: *2010 IEEE Symposium on Security and Privacy*. May 2010, pp. 191–206.

[4] Alina Colceriu. *Catching Pokemon GO in Your Network*. 2016. URL: https://www.ixiacom.com/company/blog/catching-pokemon-go-your-network.

[5] Digi-Capital. *Ubiquitous $90 billion AR to dominate focused $15 billion VR by 2022*. 2018. URL: https://www.digi-capital.com/news/2018/01/ubiquitous-90-billion-ar-to-dominate-focused-15-billion-vr-by-2022/ (visited on 07/27/2018).

[6] Adrienne Porter Felt et al. *Android Permissions: User Attention, Comprehension, and Behavior*. Tech. rep. UCB/EECS-2012-26. EECS Department, University of California, Berkeley, Feb. 2012. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-26.html.

[7] Batya Friedman and Peter H. Kahn Jr. “New directions: a value-sensitive design approach to augmented reality.” In: *Proceedings of DARE 2000 on Designing augmented reality environments*. Elsinore, Denmark: ACM, Jan. 2000, pp. 163–164.

[8] Jaybie A. de Guzman, Kanchana Thilakarathna, and Aruna Seneviratne. “Security and Privacy Approaches in Mixed Reality: A Literature Survey”. In: *CoRR* abs/1802.05797 (2018). arXiv: 1802.05797. URL: http://arxiv.org/abs/1802.05797.

[9] E. Kauderer-Abrams. “Quantifying Translation-Invariance in Convolutional Neural Networks”. In: *ArXiv e-prints* (Dec. 2018). arXiv: 1801.01450.

[10] David Kotz and Kobby Essien. *Characterizing usage of a campus-wide wireless network*. Tech. rep. Dartmouth, 2002.

[11] Wendy E. Mackay. “Augmented Reality: Dangerous Liaisons or the Best of Both Worlds?” In: *Proceedings of DARE 2000 on Designing Augmented Reality Environments*. DARE ’00. 2000, pp. 170–171.

[12] Richard McPherson, Suman Jana, and Vitaly Shmatikov. “No Escape From Reality: Security and Privacy of Augmented Reality Browsers”. In: *Proc. of the 24th Intl. Conference on World Wide Web*. WWW ’15. 2015, pp. 743–753.

[13] Jingjing Ren et al. “ReCon: Revealing and Controlling PII Leaks in Mobile Network Traffic”. In: *Proc. of the 14th Annual Intl. Conference on Mobile Systems, Applications, and Services*. MobiSys ’16. 2016, pp. 361–374.

[14] Vera Rimmer et al. “Automated Feature Extraction for Website Fingerprinting through Deep Learning”. In: *Proc. of 25th Annual Network and Distributed System Security Symposium*. NDSS. 2018.

[15] Franziska Roesner, Tadayoshi Kohno, and David Molnar. “Security and Privacy for Augmented Reality Systems”. In: *Commun. ACM* 57.4 (Apr. 2014), pp. 88–96. ISSN: 0001-0782.

[16] Christian Szegedy et al. “Going Deeper with Convolutions”. In: *CoRR* (2014). arXiv: 1409.4842. URL: http://arxiv.org/abs/1409.4842.

[17] Sebastien C. Wong et al. “Understanding data augmentation for classification: when to warp?” In: *CoRR* abs/1609.08764 (2016). URL: http://arxiv.org/abs/1609.08764.