ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (291-295), Month: April - June 2015, Available at: www.researchpublish.com

# Traffic Diagnosis for Encrypted Messaging Services: Whatsapp, Apple Imessage and Many More

[1]Sumanth Reddy Devi Reddy, [2]Prof Tarik El Taeib

***Abstract:*** **Instant messaging services have gone from a niche application used on desktop computers to the most prevalent form of communication in the world, due in large part to the growth of Internet enabled phones and tablets. Messaging services, like Apple iMessage, Telegram, WhatsApp, and Viber, handle tens of billions of messages each day from an international user base of over one billion people. Given the volume of messages traversing these services and ongoing concerns over widespread eavesdropping of Internet communications, it is not surprising that privacy has been an important topic for both the users and service providers. To protect user privacy, these messaging services offer transport layer encryption technologies to protect messages in transit, and some services, like iMessage and Telegram, offer end-to-end encryption to ensure that not even the providers themselves can eavesdrop on the messages. As previous experience with Voice-over-IP and HTTP tunnels has shown us, however, the use of state-of-theart encryption technologies is no guarantee of privacy for the underlying message content. In this paper, we analyze the network traffic of popular encrypted messaging services. (1) To understand the breadth and depth of their information leakage, (2) Determine if attacks are generalizable across services and (3) Calculate the potential costs of protecting against this leakage.**

***Keywords:*** **communication, encryption, services, privacy, messages, Internet.**

## I. INTRODUCTION

Instant messaging services are quickly becoming the most dominant form of communication among consumers around the world. Apple iMessage, for example, handles over 2 billion message each day, while WhatsApp claims 16 billion messages from 400 million international users. To protect user privacy, these services typically implement end-to-end and transport layer encryption, which are meant to make eavesdropping infeasible even for the service providers themselves. In this paper, however, we show that it is possible for an eavesdropper to learn information about user actions, the language of messages, and even the length of those messages with greater than 96% accuracy despite the use of stateof-the-art encryption technologies simply by observing the sizes of encrypted packet. While our evaluation focuses on Apple iMessage, the attacks are completely generic and we show how they can be applied to many popular messaging services, including WhatsApp, Viber, and Telegram.

We focus our daignosis on the Apple iMessage service and show that it is possible to reveal information about the Attack Method Accuracy.

| ATTACK | METHOD | ACCURACY |
|---|---|---|
| Operating System | Naive Bayes | 100% |
| User Action | Lookup Table | 96% |
| Language | Naive Bayes | 98% |
| Message Length | Linear Regression | 6.27 chars. |

Above table gives you Summary of attack results for Apple iMessage.

Device operating system, fine-grained user actions, the language of the messages, and even the approximate message length with accuracy exceeding 96%, as shown in the summary provided in Table 1. In addition, we demonstrate that these attacks are applicable to many other popular messaging services, such as WhatsApp, Viber, and Telegram, because they target deterministic relationships between user actions and the resultant encrypted packets that exist regardless of the underlying encryption methods or protocols used. Our analysis of countermeasures shows that the attacks can be completely mitigated by adding random padding to the messages, but at a cost of over 300% overhead, which translates to at least a terabyte of extra data per day for the service providers. Overall, these attacks could impact over a billion users across the globe and the high level of accuracy that we demonstrate in our experiments means that they represent realistic threats to privacy, particularly given recent revelations about widespread metadata collection by government agencies.

**iMessage Overview:**

iMessage uses the Apple Push Notification Service (APNS) to deliver text messages and attachments to users. When the device is first registered with Apple, a client certificate is created and stored on the device. Every time the device is connected to the Internet, a persistent APNS connection is made to Apple over TCP port 5223. The connection appears to be a standard TLS tunnel protecting the APNS messages. From here, the persistent APNS connection is used to send and receive both control messages and user content for the iMessage service. If the user has not recently interacted with the sender or recipient of a message, then the client initiates a new TLS connection with Apple on TCP port 443 and receives key information for the opposite party. Unlike earlier TLS connections, this one is authenticated using the client certificate generated during the registration process. Once the keys are established, there are five user actions that are observable through the APNS and TLS connections made by the iMessage service. These actions include: (1) start typing (2) stop typing (3) send text (4) send attachment (5) read receipt.

All of the user actions mentioned follow the protocol flow shown in Figure 1, except for sending an attachment. The protocol flow for attachments is quite similar except that the attachment itself is stored in the Microsoft Azure cloud storage system before it is retrieved, rather than being sent directly through Apple. Over the course of our analysis, we observed some interesting deviations from this standard protocol. For instance, when TCP port 5223 is blocked, the APNS message stream shifts to using TCP port 443. Similarly, cellular-enabled iOS devices use port 5223 while connected to the cellular network, but switch to port 443 when WiFi is used. Moreover, if the iOS device began its connection using the cellular network, that connection will remain active even if the device is subsequently connected to a wireless access point. It is important to note that payload sizes and general APNS protocol behaviors remain exactly the same regardless of if port 5223 or 443 are used, and therefore any attacks on the standard APNS scenarios are equally applicable in both cases.

## II. ANALYSIS

Here, we investigate information leakage about devices, users, and messages by analyzing the relationship between packet sizes within the persistent APNS connection used by iMessage and user actions. For each of these categories of leakage, we first provide a general analysis of the data to discover trends or distinguishing features, then evaluate classification strategies capable of exploiting those features.

**Data Methodology:**

To evaluate our classifiers, we collected data for each of the five observable user actions (start, stop, text, attachment, read) by using scripting techniques that drove the actual iMessage user interfaces on OSX and iOS devices. Specifically, we used Applescript to natively type text, paste images, and send/read messages on a Macbook Pro running OSX 10.9.1, and a combination of VNC remote control software and Applescript to control the same actions on a jailbroken iPhone 4 (iOS 6.1.4). For each user action, we collected 250 packet capture examples on both devices and in both directions of communications (i.e., to/from Apple) for a total of 5,000 samples. In addition, we also collected small samples of data using devices running iOS 5, iOS 7, and OSX Mountain Lion to verify the observed trends. Throughout the remainder of the paper, we simply refer to attachments as “image” messages. Although the Tatoeba dataset does not contain typical text message shorthand, it is generated through a community of non-expert users (i.e., crowd-sourced) and so actually contains several informal phrases that are not found in a typical language translation corpus. Each experiment in this section used 10-fold cross validation testing, where the data for each instance in the test was constructed by sampling TCP payload lengths and packet directions (i.e., to/from Apple) from the relevant subset of the packet capture files. The only preprocessing that was performed on the data was to remove duplicate packets that occur as a result of TCP retransmissions and those packets without TCP payloads. Performance of our classifiers is report with respect to overall accuracy, which is calculated as the sum of the true positives and true negatives over the total number of samples evaluated. Where appropriate, we also use confusion matrices that show how each of the test instances was classified and use absolute error to measure the predictive error in our regression analysis.

**Operating System:**

The scatterplot of iMessage packet sizes shows how iOS appears to more efficiently compress the plaintext, while OSX occupies a much larger space. These two classes of data are clearly separable, but the figure also shows five unique bands of plaintext/ciphertext relationship, which hints at leakage of finer-grained information about the individual messages. Additionally, when we break down the distributions based on their direction (to/from Apple), we see that there is a deterministic relationship between the two. That is, as messages pass through Apple, 112 bytes of data are removed from OSX messages and 64 bytes are removed from iOS messages. Aside from the ability to fingerprint the OS version, the deterministic nature of these changes indicates that it is also possible to correlate and trace communications as it passes through Apple on the way to its destination.

To identify the OS of observed devices, we use a binomial naıve Bayes classifier from the Weka machine learning library with one class for each of the four possible OS, direction combinations. The classifier operates on a binary feature vector of packet length, direction pairs, where the value for a given dimension is set to “true” if that pair was observed and “false” otherwise. To determine the number of packet observations necessary for accurate classification, we run 10-fold cross-validation experiments where the 1,024 instances used for each experiment are created with N = 1, 2, . . . , 50 packets sampled from the appropriate subset of the dataset for each OS, observation point class. The results indicate that we are able to accurately classify the OS with 100% accuracy after observing only five packets regardless of the operating system. A cursory analysis of iOS 5 and 7 indicates that they also produce messages with lengths that are unique from both the OSX and iOS 6.1.4 device, which indicates that this type of device fingerprinting could be refined to reveal specific version information when the size of the APNS messages changes between OS versions. 3.3 User Actions. Recall from our earlier discussion that there are five highlevel user actions that we can observe: start, stop, text, attachment (image), and read. Most classes have two distinctive packet lengths – one for when the message is sent to Apple and one when it is received from Apple.

The only classes that overlap substantially are the read receipt and start messages in the iOS data going to Apple. The stability and deterministic nature of the payload lengths in most classes makes the use of probabilistic classifiers unnecessary. Instead of using heavyweight machine learning methods, we create a hash-based lookup table using each observed length in the training data as a key and store the associated class labels. In addition to creating classes for the five standard message types derived from user actions, we also create a class for the payload lengths of identified control packets. When a new packet arrives, we check the lookup table to retrieve the class label(s) for its payload length. If only one label is found, the packet is labeled as that message type. In the case where two class labels are returned, we choose the class where that payload length occurs most frequently in the training data. In an effort to focus our evaluation, we assume that the OS has already been accurately classified such that we have four separate message-type classifiers, one for each combination of OS and direction. Each of the classifiers is evaluated using 10-fold cross validation with instances drawn from the respective subsets of the dataset, for a total of 1,250 instances per classifier. The accuracy is surprisingly good for both iOS and OSX given such a simple classification strategy. As it turns out, all message types can be classified with accuracy exceeding 99%, except for iOS read messages that are easily confused with start messages.

**Message Attributes:**

The final experiment in our analysis of information leakage examines if it is possible to learn more detailed information about the contents of messages, such as their language or plaintext length. The foundation for this experiment is built upon the observation that shows several distinct clusters when comparing plaintext message length to payload length. While the clusters are most prevalent in the OSX data, the iOS data also has a similar set of clusters (albeit more compressed).

When we separate this data into its constituent languages, the reason for these clusters becomes clear. Essentially, each cluster represents a unique character set used in the language (e.g., ASCII, Unicode). For languages that use only a single character set, like English (ASCII), Russian (Unicode), or Chinese (Unicode), there is only one cluster approximating a linear relationship between plaintext and payload lengths, with a “stair step” effect at AES block boundaries.

The other three languages all use some mix of ASCII and Unicode characters, resulting in an ASCII cluster with better plaintext/payload length ratios, and Unicode cluster that requires more payload bytes to encode the plaintext message. These graphs also help to answer our question about the possibility of guessing the message lengths, which is supported by the approximately linear relationship that appears. To test our ability to classify these languages, we use the Weka multinomial naive Bayes classifier, with raw counts of each length, (packet) direction pair observed so that we can take full advantage of the subtle differences in the distribution. As with previous experiments, we assume that earlier classification stages for OS and message type were 100% accurate in order to focus specifically on this area of leakage. The results from 10-fold cross validation on 1,024 instances generated from N = 1, 2, . . . , 50 text message packets. Classification of languages in OSX data is noticeably better than iOS, as we might have expected due to compression. On the OSX data, we achieve an accuracy of over 95% after 50 packets are observed. When applied to the iOS data, on the other hand, accuracy barely surpasses 80% at the same number of packets. However, by the time we sample 100 packets all languages are achieving classification accuracies of at least 92% regardless of the dataset.

Given that language classification can be achieved with high accuracy after a reasonable number of observations, we now move on to determining how well we can predict message lengths within those languages. For this task, we apply a simple linear regression model using the payload length as the explanatory variable and the message length as the dependent variable. The models are fitted to the training data using least squares estimation. Again, we performed 10-fold cross validation with 1,024 instances and calculated the resultant absolute error. In general, the values are small – an error of between 2 and 11 characters – when we consider that the sentences in the language dataset range from two characters to several hundred, with an average error of 6.27 characters. Those languages with multiple clusters, like French and Spanish, fared the worst since the linear regression model could not handle the bimodal behavior of the distribution for the multiple character sets. For completeness, we also applied a regression model to the image transfers to and from the Microsoft Azure cloud storage system. The regression model was extremely accurate for the attachments, with an absolute error of less than 10 bytes.

## III. BEYOND IMESSAGE

Thus far, we have focused our attacks exclusively on Apple iMessage, however we note that they rely only on the user’s interaction with the messaging service and a deterministic relationship between those actions and packet sizes. In effect, the attacks target fundamental operations that are common to all messaging services, to examine the leakage of user actions and message information in the WhatsApp, Viber, and Telegram messaging services. Just as with Apple iMessage these three messaging services clearly allow us to differentiate fine-grained activities by examining individual packet sizes. Moreover, when we examine the relationship between plaintext message lengths and ciphertext length. First, it shows that the same general strategies used to infer user actions, languages, and message lengths can be used across many of the most popular messaging services regardless of their individual choices in data encoding, protocols, and encryption. Second, it is clear that WhatsApp and Viber provide even weaker protection against information leakage than iMessage, since there are exact oneto-one relationships between packet sizes and plaintext message lengths. Specifically, Apple iMessage data showed a “stair step” pattern due to the AES block sizes used, which naturally quantizes the output space and adds uncertainty to message length predictions, while Viber and WhatsApp allow us to exactly predict message length. Telegram, with its use of end-to-end encryption technology, appears to be very similar to iMessage in terms of its payload length distributions. Therefore, we can expect the accuracy of the attacks will be at least as good as what was demonstrated on Apple iMessage traffic.

To mitigate against such privacy failures, it is possible to apply standard padding-based countermeasures. Apple iMessage and Telegram already implement a weak form of countermeasure through packet sizes quantized at AES block boundaries. A much more effective approach, however, would be to add random padding independently to each packet up to the maximum observed packet length for each service, thereby destroying any relationship to user actions. When implemented on our Apple iMessage data, the random padding methodology reduced all of our attacks to an accuracy of 0% at the cost of 613 bytes (328%) of overhead per message for iOS and 596 bytes (302%) for OSX. Although the absolute increase in size is rather small, we must consider that services like iMessage handle upwards of 2 billion messages every day, which translates to an additional terabyte of network traffic daily. For the more popular WhatsApp service, a similar increase would incur at least 4 terabytes of overhead. Other countermeasure methods, such as traffic morphing, may actually provide a more palatable trade-off between overhead and privacy. Overall, the attacks that we have demonstrated raise a number of very important questions about the level of privacy that users can expect from these services. While the exact plaintext content cannot (yet) be revealed, rich metadata can be learned about a user and their social network. In the wake of recent reports of widespread metadata gathering by government agencies and given the unusually broad impact of these attacks on an international user base, it seems reasonable to assume that these types of attacks are a realistic threat that should be taken seriously by messaging services.

## REFERENCES

[1] Spencer Ackerman and James Ball. Optic Nerve: Millions of Yahoo Webcam Images Intercepted by GCHQ. http://www.theguardian.com/world/2014/feb/27/gchqnsa-webcam-images-internet-yahoo, February 2014.

[2] Inc.Apple.iOSSecurity.http://images.apple.com/iphone/business/docs/iOS_Security_Feb14.pdf, February 2014.

[3] Marjorie Cohn. NSA Metadata Collection: Fourth Amendment Violation. http://www.huffingtonpost.com/marjorie-cohn/nsametadata-collection-f_b_4611211.html, January 2014.

[4] K.P. Dyer, S.E. Coull, T. Ristenpart, and T. Shrimpton. Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In Proceedings of the 33rd IEEE Symposium on Security and Privacy, pages 332–346, May 2012.

[5] Michael Frister and Martin Kreichgauer. PushProxy: A Man-in-the-Middle Proxy for iOS and OS X Device Push Connections. https://github.com/meeee/pushproxy, May 2013.

[6] Dan Goodin. Can Apple Read Your iMessages? Ars Deciphers End-to-End Crypto Claims. http://arstechnica.com/ security/2013/06/can-appleread-your-imessages-ars-deciphers-end-to-endcrypto-claims/, June 2013.

[7] Matthew Green. Can Apple read your iMessages? http://blog.cryptographyengineering.com/2013/06/canapple-read-your-imessages.html, June 2013.

[8] Andy Greenberg. Apple Claims It Encrypts iMessages And Facetime So That Even It Can’t Decipher Them. http://www.forbes.com/sites/andygreenberg/2013/06/ 17/apple-claims-it-encrypts-imessages-and-facetimeso-that-even-it-cant-read-them, June 2013.

[9] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA Data Mining Software: An Update. SIGKDD Explorations, 11(1), 2009.

[10] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naive-Bayes Classifier. In Proceedings of the ACM Workshop on Cloud Computing Security, pages 31–42, November 2009.