Modeling and Monitoring Wi-Fi Calling Traffic in Enterprise Networks Using Machine Learning

Sharat Chandra Madanapalli*, Arunan Sivanathan*, Hassan Habibi Gharakheili*, Vijay Sivaraman*,
Santosh Patil† and Byju Pularikkal†
*University of New South Wales, Sydney, Australia, †Cisco Systems Inc., USA
Emails: {sharat.madanapalli@student., a.sivanathan@, h.habibi@, vijay@}unsw.edu.au
{santpati, byjupg}@cisco.com

***Abstract*—Many enterprise campuses have poor signal coverage indoors from one or more mobile operators, and thus are increasingly embracing carrier Wi-Fi calling services, allowing their users to make and receive mobile phone calls over the enterprise Wi-Fi connection. Mobile carriers employ IPSec tunnels to secure user calls and messages that traverse untrusted enterprise networks and possibly the public Internet. These encrypted connections from user handsets are seen as potential security threats in enterprise networks. In this paper, we develop a machine learning-based system for monitoring encrypted traffic of IPSec tunnels on the network to distinguish Wi-Fi calling traffic from anomalies. Our contributions are as follows: (1) We analyze traffic traces consisting of carrier Wi-Fi calls made over four mobile networks to highlight network behavioral characteristics of this enterprise application. We develop a set of models using one-class and multi-class classification algorithms to determine if Wi-Fi calling application is present on the IPSec tunnel (if so, to classify its state), otherwise generate a notification to block the non Wi-Fi calling flow, and (2) We evaluate the efficacy of our system in detecting real calls and their states (initiation, heartbeat, and actual call) as well as raising true alarms in case of anomalous traffic.**

I. Introduction

Carrier Wi-Fi calling, one of the voice-over-Wi-Fi applications, was initially launched by a handful of mobile operators in the US and Europe and is being offered by a growing number of Tier-1 operators around the world [1]. According to Cisco, it will take 53% of mobile IP voice service usage by 2020, a more than three-fold increase from 2015 [2]. This technology was primarily designed for consumers to compensate poor cellular coverage inside their homes but also benefits enterprises with wireless-shielded buildings. It is a standard protocol that allows users to make regular mobile phone calls (and also send texts and other media) over Wi-Fi networks with no need for an additional application – just from phones native dialer. For mobile network operators, offload of mobile voice traffic onto Wi-Fi access networks is beneficial, since they can offer high-quality voice calls and SMS at a lower price, thereby combating over-the-top applications like Skype and WhatsApp.

Mobile carriers do not trust third-party-owned Wi-Fi access networks (and the Internet) to carry important voice calls, and therefore they establish secure tunnels (IPSec) between user devices and their core packet gateway to protect the Wi-Fi-calling traffic. However, the use of IPSec tunnels creates a security concern for enterprises who want to allow the Wi-Fi-calling service within their organization, since their firewall policy needs to unblock IPSec flows with no ability to detect the application carrying the encrypted traffic.

This paper describes our solution for modeling network behavior of Wi-Fi calling traffic that enables enterprise network operators to automatically monitor encrypted UDP flows and detect anomalies in real-time. We begin by analyzing real traffic traces of Wi-Fi calls from four mobile network operators and highlight the key characteristics of Wi-Fi calling flows on the network. We then train one-class classifiers as well as a multi-class classifier that are collectively able to distinguish Wi-Fi-calling flows from anomalous ones. Finally, we prototype our system and validate it with real traffic.

II. Related Work

Mobile data offload onto Wi-Fi access networks is well understood and practiced by industry and academia [3], [4]. However, mobile voice offload has become prevalent only in recent years when phone manufacturers started to natively support Wi-Fi calling [5]. Analysis of network traffic to identify applications has been an active research area for decades. Specifically, voice over IPSec has been studied in [6] where the authors quantified the performance of the application in terms of bandwidth usage and transmission delay. Similar to our work, authors of [7] aim to detect VoIP packets over IPSec tunnels to forward them with the highest priority (*i.e.,* for quality of service). Their proposed method is quite simple and only checks the packet size within a fixed range. We, instead, compute 8 attributes from time-series profile of each IPSec flow and develop four ML models to capture the normal behavior of Wi-Fi calling application.

In terms of security, work in [8] surveys threats and vulnerabilities of VoIP technology. However, security of Wi-Fi calling application has not been well studied until recently [9], [10]. Authors of [9] demonstrate various attacks on end-user devices where IPSec keys can be extracted from SIM cards. Work in [10] conducts a comprehensive study on vulnerabilities of the Wi-Fi calling service over major mobile operators in the US, and highlight several privacy risks (*e.g.,* inferring user identity, call statistics, and device information) for users of this application. Our work primarily aims to automatically ensure that only Wi-Fi calling traffic is exchanged with trusted mobile operators over IPSec tunnels from enterprise networks – anomalous flows are blocked in real-time.

III. Modeling Wi-Fi Calling Application

A. Network Behavior of Wi-Fi Calling

A Wi-Fi calling session starts with IPSec tunnel establishment, followed by *initiation* phase wherein it exchanges device details like phone number, and then stays in a heartbeat (*i.e., keep-alive*) phase until a *call* is made [10]. Let us now look into a real trace of a real Wi-Fi calling session and its various phases in Fig. 1. The Wi-Fi call was made over a major US-based carrier network using an LG Android device – we observed similar network profiles on iOS devices across various carrier networks.

IPSec Tunnel Establishment. When a mobile device (with Wi-Fi calling feature enabled) connects to a Wi-Fi network, it first fetches a server IP corresponding to carrier's Wi-Fi calling endpoint by sending a DNS query. The device next establishes a secure IPSec tunnel with the server using the IKE and ISAKMP protocols [11] and encapsulates the data in the tunnel using ESP protocol over UDP [12]. Note that we can not solely rely on DNS information to ensure that the application is Wi-Fi calling or not. For example, in case of a DNS spoofing attack, the man-in-the-middle attacker can reply to the legitimate DNS query, causing the device to establish the tunnel to a malicious server. Therefore, it is needed to detect the application using the profile of traffic exchanged over the tunnel which is a non-trivial exercise.

Initiation. Following tunnel establishment, the Wi-Fi Calling application exchanges device specific information (*e.g.,* phone number) with the carrier's server. This typically takes about 7 to 10 seconds and results in the first peak highlighted in red (Fig. 1). This phase consists of ESP packets (typically over 1000 bytes each) transferred at a rate of 200-400 Kbps.

Keep-Alive. Following the initiation, the application enters into a Keep-Alive phase wherein one packet (of 60 bytes) is sent every 20 seconds from the server to the device only for keeping the NAT mappings alive (as described in detail in [12]). Also, every couple of minutes, a pair of ISAKMP request/response (Informational type) packets of size 122 bytes are exchanged. The application continues to be in this phase, until a call is made.

Call. When a user makes a call, the device send/receives call data using the IPSec tunnel established over Wi-Fi. It can be seen in Fig. 1 that data packets flow bidirectionally. The amount of data transferred depends on the conversation, *i.e.,* in this example, the user who made the call does not seem to talk much and hence less upload traffic. Further, in comparison to the initiation, the traffic rates is lower and seems to have an upper cap of around 100 Kbps. The packets exchanged are typically 174 bytes. Note that upon termination of the call, the application switches back to the Keep-Alive phase.

Although a representative Wi-Fi calling session occurs this way, we have also observed certain minor differences. For example, the IPSec connection might get terminated and re-established periodically (commonly observed in iOS devices). We think it is probably because the phone disconnects the session to save battery during idle periods (*i.e.,* keep-alive phase). We note that only one IPSec connection is active at any point in time. Further, we observed that sometimes during tunnel re-establishment the server IP might change (but it can be captured by an updated DNS response) due to dynamics of cloud-based services. Additionally, we have observed that the UDP port used on the client is selected at random in Android devices, but is set to 4500 (identical to the server port) in iOS devices. Nonetheless, across the 4 major providers and the two operating systems, we have observed the traffic profile of each phase to be almost identical and thus helping us build a general model to detect and monitor the Wi-Fi calling.

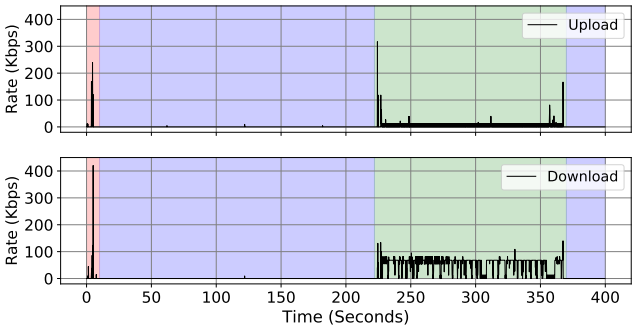


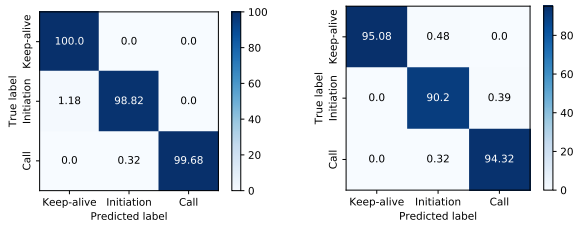
Fig. 1. Network profile of Wi-Fi calling traffic.

B. Phase Classification and Anomaly Detection

We now explain our method to detect Wi-Fi calling sessions using the network activity data. As explained in previous section a Wi-Fi calling flow can be in one of three phases: Initiation, Keep-Alive or Call. Thus, given an IPSec flow established for Wi-Fi calling, we need to detect and monitor these phases in real-time. We also need to identify IPSec flows which are not established for Wi-Fi calling, considered as anomalies in our use-case. To perform these tasks, we first break the traffic profile (*i.e.,* time-series signal) into fixed-length windows and extract appropriate features from its activity. These features are used to train models which perform two tasks: (a) classify phases of a legitimate Wi-Fi calling session, and (b) detect anomalous behavior by a non Wi-Fi calling session.

Feature Extraction. We have observed that Wi-Fi calling application exhibits certain characteristics: (a) transfers content at rate less than 500 Kbps, (b) is generally idle (*i.e.,* mostly in Keep-Alive phase), and (c) has a distinct patterns of packet-sizes. Using these observations we extracted a set of attributes for a 10-second window of each IPSec flow. We have chosen 10 seconds for our window size since the initiation phase typically takes 7-10 seconds to finish and to classify this phase we need a minimum of 10-second worth of data. Each 10-second window consists of byte counts and packet counts computed every 100ms, *i.e.,* a total of 100 data points.

For each window, we compute the following set of attributes in both upload and download directions: (1) average packet size for highlighting the initiation phase; (2) zero fraction (fraction of time no data is exchanged) for highlighting the Keep-Alive phase, (3) average, and (4) max transfer rate for highlighting the call phase. These attributes help us classify phases and detect anomalies as explained below.



(a) Random-Forest model. (b) Isolation-Forest models.

Fig. 2. Performance of: (a) Random-Forest, and (b) Isolation-Forest, models.

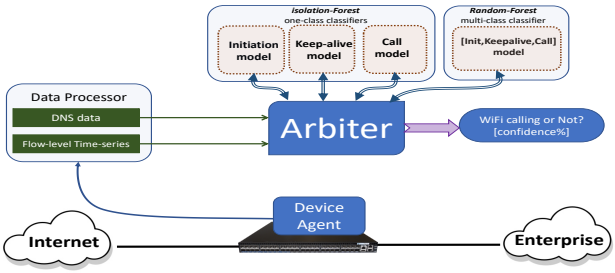


Fig. 3. System architecture of prototype for Wi-Fi calling monitoring.

Training Dataset. We collected 20 PCAP traces of Wi-Fi calling sessions from Cisco labs. We then built a dataset consisting of a total of 4,162 labeled instances, each corresponding to 10-second worth of traffic trace for phases Initiation (255 instances), Keep-Alive (3,574 instances), and Call (317 instances). Unsurprisingly, most of instances are keep-alive since the application tends to spend more time in this phase. Each labeled contains 8 attributes mentioned above.

Multi-Class Classification. We trained a Random Forest classifier (*i.e.,* decision tree-based learning) on our dataset using the scikit-learn library in Python – this model would generate a pair of outputs: a label (*i.e.,* Initiation, Keep-Alive, or Call) and a confidence-level. We used 80% of the data to train our model and the remaining 20% for testing it. We have achieved an accuracy of 99.7% in classifying phases. Fig. 2(a) shows the confusion matrix of our Random-Forest model. We can see that this model performs well in learning patterns to distinguish among the expected phases. We also need more precise and possibly sensitive models (*i.e.,* one-class classifier is explained next) for individual phases which together with the Random-Forest model enable us to detect anomalous flows.

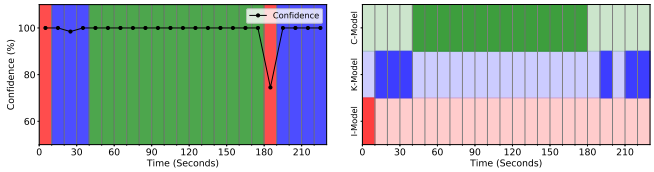
One-Class Classification. We built three models using Isolation Forest algorithm, each is specialized in expected behavior of one phase in a Wi-Fi calling flow – each model would generate a binary output (*i.e.,* positive if expected profile is detected in the instance, otherwise negative). Consequently, an anomaly (*i.e.,* a non Wi-Fi calling flow) would be detected if none of these models generate a positive output. To train the models, we passed 80% of the data corresponding to each phase and set the contamination rate (of Isolation forest algorithm) to 0.05. We tested each models by two datasets: (a) remaining 20% of instances from its corresponding phase, and (b) all instances from other two phases. For example, the initiation model was tested by 20% of Initiation instances (unseen by the model during training) and all instances of Call and Keep-Alive. Fig. 2(b) shows the confusion matrix of testing Isolation-forest models. For example, 95.08% of Keep-Alive instances are correctly detected by its intended model (top left cell), while less than 1% incorrectly detected by the Initiation model and none by the Call model. We also note that 4% of Keep-Alive instances are not detected by any of the three models, this measure is 9% and 5% for the Initiation and Call instances, respectively.

Combination of One-Class and Multi-Class. Note that Random Forest performs well in capturing decision boundaries and develops rules to differentiate the behavior of various classes. However, even if the input does not belong to any of trained classes (*i.e.,* anomalies), the model still predicts one of leaned classes – such predictions are typically accompanied by lower confidence values since there will be a disagreement among decision trees within the forest. This task of identifying anomalous data can be better done using Isolation Forest. They form tight bounds on attribute values of a benign class, and report anomalies if a small non-conformance is observed. However, these models tend to be very sensitive and may miss to output positive signal for benign instances. do not generalize well to differentiate among classes and perform the task of classification. Thus, in our system, we use both types of models to accurately monitor the carrier Wi-Fi calling application, *i.e.,* tracking its intended phases and also reporting non Wi-Fi calling IPSec traffic.

IV. Prototype and Experimentation

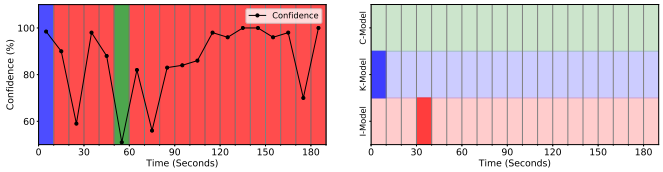
We prototyped our scheme in a small testbed, depicted in Fig. 3, which can be readily deployed in enterprise networks. In this architecture, "Device Agent" (running on a network switch) extracts flow level information from the raw packets passing through the switch. In our prototype, we have used Cisco's Joy [13] open-source software to perform this task. Joy extracts a time-trace of byte and packet counts from each flow and aggregates them over a time window, say 10 seconds, and sends an IPFIX packet to "Data Processor". The data processor decodes the IPFIX packet and performs the following tasks: (a) extracts DNS query name and the corresponding server IP (used to identify carrier Wi-Fi calling endpoints), (b) filters IPSec flows (*i.e.,* UDP 4500), and (b) aggregates raw byte-count and packet-count at resolution of 100ms to generate 100 data points. These inputs are passed on to "Arbiter" which performs multiple functions and finally outputs whether the application in the IPSec tunnel is Wi-Fi calling with a confidence value.

Before understanding the arbiter module and its functions, let us walk through how our models perform for real Wi-Fi calling and non Wi-Fi calling flows as shown in Figures 4 and 5. For the Random-Forest model (Fig. 4(a) and Fig. 5(a)), each column represents a time window of 10 seconds and its color shows the classified phase – red, blue, and green respectively correspond to Initiation, Keep-Alive and Call. For the Isolation-Forest models (Fig. 4(b) and Fig. 5(b)) color codes are identical to of Random-Forest and each row represents the output of each Isolation-Forest mode – a darker



(a) Random-Forest model. (b) Isolation-Forest models.

Fig. 4. Performance of models with a Wi-Fi calling flow.



(a) Random-Forest model. (b) Isolation-Forest models.

Fig. 5. Performance of models with a non Wi-Fi calling flow.

color indicates the positive signal from the model.

Considering a legitimate Wi-Fi call, the Random-Forest model, as shown in Fig. 4(a), consistently classifies with high confidence (*i.e.,* $>= 97\%$), the first instance as initiation, followed by three Keep-Alive instances, and 14 call instances until the second 180. The Isolation-Forest models also accurately detect the same phases in the first 18 epochs, as shown in Fig. 4(b). The next instance (*i.e.,* 180-190 second), however, gets classified as initiation by random forest with lower confidence and isolation forest models do not detect any phase. Right after that, the last four instances are accurately classified by random forest while isolation forest misses one instance (*i.e.,* 200-210 second). We investigated the misclassified instance between 180s and 190s and found that the call was active for the first 3 seconds followed by 7 seconds of Keep-Alive. This caused the models to be confused as they were trained on epochs containing just one phase. We acknowledge that such false alarms might occur for a single instance but instances surrounding this miss-classified instance should still be accurately classified (given that the traffic corresponds to a Wi-Fi call). For a non Wi-Fi calling application, we can see that random forest detects Keep-Alive first (which never occurs in Wi-Fi calls) and subsequently mostly classifies instances as initiation phase with a highly varying confidence ($< 85\%$). Further, the isolation forest models, shown in Fig. 5(b), do not detect any of the phases for most of the time except detecting keep-alive in the beginning and one initiation between seconds 30 and 40.

These observations from performance of our models on real traffic helped us design the functions of the Arbiter to accurately predict the phase of Wi-Fi calling flows, monitor them, and detect anomalies. The arbiter needs to perform the following functions: (a) to maintain a window of models' outputs to discount one-off miss-classifications and accurately report anomalies, (b) to combine outputs of multiple models to take a decision, and (c) to report events such as IPSec session detected, presence of and current phase of Wi-Fi calling application, and anomalous non-WiFi calling IPSec flows. The arbiter keeps track of each bidirectional IPSec flow and the corresponding model outputs for the last "k" epochs. Then it looks at these k epochs to decide whether the application is indeed Wi-Fi calling. For example, with k=3 (over last three epochs), if the isolation forests did not detect any of the phases and average confidence of random forest classifier is lower than a configurable threshold, say 80%, it deems the flow to be anomalous and sends an anomaly event notification. Otherwise, it is a normal Wi-Fi calling session. We additionally use the DNS information captured by the arbiter to decide the parameter k. If the domain queried is indeed legitimate, we set k to be a larger value say 5 to tolerate miss-classifications, if any, as DNS information suggests a legitimate Wi-Fi call. The phases are reported by using the classification output and confidence of random forest classifier as it specializes in that task. Whenever the arbiter detects a flow that is not present in its state, it sends a new IPSec flow event notification. It ages out flows based on inactivity using a timer and sends a notification for the IPSec flow terminated. With this system design, we were able to accurately identify Wi-Fi calls and raise anomalies within the first window of observation for non Wi-Fi calling IPsec sessions.

V. Conclusion

In this paper we have developed a solution for enterprise networks who want to allow Wi-Fi calling over encrypted IPSec tunnels. We have analyzed real traces of Wi-Fi calling traffic and identified key behavioral network patterns. We then developed ML-based models to classify three phases of a Wi-Fi calling flow and detect anomalous traffic exchanged inside an IPSec tunnel. Lastly, we prototyped our scheme in a testbed to show how benign Wi-Fi calling traffic can be automatically distinguished from anomalous IPSec flows.

References

[1] Apple. (2019) Wireless carrier support and features for iPhone. [Online]. Available: https://support.apple.com/en-au/HT203982

[2] Cisco. (2016) Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020. [Online]. Available: https://bit.ly/2UvEB73

[3] S. Dimatteo *et al.*, "Cellular Traffic Offloading Through WiFi Networks," in *Proc. IEEE MASS*, 2011.

[4] F. Rebecchi *et al.*, "Data Offloading Techniques in Cellular Networks: A Survey," *IEEE Communications Surveys Tutorials*, vol. 17, no. 2, pp. 580–603, Secondquarter 2015.

[5] B. Pularikkal *et al.*, "Carrier Wi-Fi Calling Deployment Considerations," Working Draft, IETF Secretariat, Internet-Draft, January 2017. [Online]. Available: https://www.ietf.org/archive/id/draft-pularikkal-opsawg-wifi-calling-03.txt

[6] R. Barbieri, D. Bruschi, and E. Rosti, "Voice over IPsec: Analysis and Solutions," in *Proc. ACSAC*, Washington, DC, USA, Dec 2002.

[7] T. Yildirim and P. Radcliffe, "VoIP traffic classification in IPSec tunnels," in *Proc. ICEIE*, Kyoto, Japan, Aug 2010.

[8] A. D. Keromytis, "A Comprehensive Survey of Voice over IP Security Research," *IEEE Communications Surveys Tutorials*, vol. 14, no. 2, pp. 514–537, Second 2012.

[9] S. Chalakkal *et al.*, "White paper: Practical Attacks on VoLTE and VoWiFi," ERNW Enno Rey Netzwerke, Tech. Rep., July 2017. [Online]. Available: https://bit.ly/2XZk882

[10] T. Xie *et al.*, "The Dark Side of Operational Wi-Fi Calling Services," in *Proc. IEEE CNS*, Beijing, China, May 2018.

[11] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen, "Internet key exchange protocol version 2 (ikev2)," Tech. Rep., 2014.

[12] A. Huttunen *et al.*, "UDP encapsulation of IPsec ESP packets," Tech. Rep., 2004.

[13] Cisco. (2019) Joy Tool. [Online]. Available: https://github.com/cisco/joy