Detecting MS Initiated Signaling DDoS Attacks in 3G/4G Wireless Networks

Aman Gupta*, Tanmay Verma*, Soshant Bali, Sanjit Kaul
IIIT Delhi, India

***Abstract*—The hierarchical architecture of present day cellular data networks implies that a large number of base stations depend on a small number of core network elements for essential services (including Internet connectivity). If a mobile botnet launches a distributed signaling attack on one or more core network elements (e.g., gateway), a large number of subscribers would experience service degradation. In this work, we propose a new detector that examines a subset of IP packets transmitted by a mobile station (MS) to determine its infection status. Service providers can install this detector anywhere in the data path, i.e., MS, Base Station (BS), gateway, etc., to detect and quarantine infected terminals. The proposed algorithm was trained using one week of IP packet traces generated by 62 different smartphones. Results indicate that this method can detect most types of signaling attacks with more than 0.9 detection probability and less than 0.1 false alarm probability.**

I. Introduction

Present day cellular wireless data networks (like LTE, HSDPA, EVDO etc.) have evolved from their voice only ancestors (like GSM, IS-95, etc.) and as a result the legacy hierarchical architecture can still be seen in today's 3G/4G networks. In this architecture, a large number (thousands) of base stations (BS) are controlled by a few elements like radio network controllers (RNC) and core network gateways (like PDSN, ASN-GW, GGSN, P-GW, etc.). For example, one PDSN and five RNCs may provide EVDO connectivity to the entire San Francisco Bay Area, that has several hundred base stations. This hierarchy implies that an attacker can degrade service to a large city by attacking just one (or a few) core network elements, like gateway or RNC [1], [7].

Whenever a mobile station (MS) authenticates with the network, requests an IP address, reserves or relinquishes the traffic channel, performs handoff to another BS, requests QoS flows, etc., it exchanges several signaling messages with the core network gateway. In addition, all Internet traffic destined to (or arriving from) the MS traverses the core network gateway. Most gateways use specialized network processors or application specific integrated circuit (ASIC) chips to forward IP data traffic (fast path). However, the signaling traffic is handled by the gateway's CPU (slow path) because the amount of computational logic required to handle these requests is significant [12]. The same CPU is typically also the central manager that controls other gateway functions, like master-slave state synchronization, network processor forwarding table programming, slave health monitoring, etc [12]. As a result, a temporary increase in signaling message volume can overload the gateway's CPU and paralyse it by delaying essential time-critical management tasks. This delay may cause serious problems in the gateway, which may eventually lead to a temporary Internet outage in a large geographic region [1], [7], [11]. To avoid this scenario, Gateways are typically equipped with signaling rate limiters that drop signaling messages when their arrival rate is very high [12]. Even though this protects the CPU, many signaling messages are dropped, resulting in user-perceived impairments. For example, when a mobile subscriber's handoff requests are dropped by the rate limiter, it loses Internet connectivity when it moves closer to the new BS.

A number of signaling messages are exchanged between the mobile station (MS) and the access network (AN) whenever a traffic channel is established (STANDBY to READY state transition) or released (READY to STANDBY state transition). For example, in UMTS networks, MS and AN process at least 20 signaling messages whenever the channel is established or released [1]. The MS requests a traffic channel whenever its send buffer has one or more packets queued for transmission, and releases it when no packets are sent or received for some time (timeout of T sec). One can envision a malicious application that sends an IP packet once every $T + \delta$ ($\delta < 1$ sec) just to increase the control signaling volume. If this behavior is the characteristic of a self propagating worm that spreads to a large number of mobile hosts, this botnet can launch a signaling attack on the core network gateway (at the command of a botmaster) and degrade connectivity in a large city [14], [15]. A naive attack detector would be one that checks the signaling rate of each MS and triggers an alarm if this rate is above some fixed threshold for a long enough period of time (e.g., using CUSUM detector [1]). However, since signaling rate is a function of usage and operating system (Android v/s IPhone v/s Symbian) [13], this simple fixed-for-all threshold detector would probably misclassify a heavy user as an attacker. Alternatively, a more sophisticated detector can examine IP packets to infer the presence of a malicious signaling attack application. We propose one such detector in this paper which, for scalability reasons, examines header information of only a fraction of all IP packets generated by a MS to infer the presence of a signaling attack application. This detector can be installed anywhere in the data path, e.g., the mobile handset, base station, gateway, deep packet inspection (DPI) system, etc. The proposed features compare

978-1-4673-5494-3/13/$31.00

*The authors contributed equally to this work

fields in IP packet headers to determine if the packets are unusually similar (or unusually dissimilar) to each other. If packet similarity appears very uncharacteristic of a normal MS, then the MS that generated this traffic is classified as an attacker.

The classifier was trained using one week long IP traffic traces collected from 62 different smartphones (23 Android, 6 Nokia, 5 iPhone, 2 Windows Mobile, 5 Samsung Bada 1 Blackberry, 20 unidentified) that belong to undergraduate university students. These traces were used to form labeled training samples of normal MS. For malicious MS, since our search for an existing signaling attack application did not yield any results, we were compelled to generate our own attacks. We generated 7 different types of attacks to train and test our system and the results were very encouraging in most cases. The detector when trained using a particular type of attack was tested using both the same type of attack and a different type of attack for which it had no prior training. The observed performance was fairly good in both cases.

II. Related Work

Signaling DoS attacks for 3G/4G networks have been discussed in [1], [3], [4], [5], [6], [7], [10]. Unlike the MS initiated signaling DOS attack proposed in our paper, the attack in [1] is launched from a host in the Internet. The attack program sends small sized IP packets from the Internet host to a large number of MS, causing many of them to transition from STANDBY to READY mode. This can dramatically increase the core network signaling load resulting in network wide service degradation. However, this attack cannot be launched in many (103 out of 180 [7][8]) 3G/4G networks due to the use of Network Address Translation (NAT). A NAT proxy only allows incoming packets that arrive in response to a recent outgoing packet. Unsolicited incoming packets are rejected because the NAT proxy does not have any destination port to private IP mappings for such packets. Although the primary reason for deploying NAT is the shortage of unused IPv4 addresses, an accidental outcome is automatic protection from network initiated signaling DoS attacks [7]. In systems that are not protected with NAT, an Internet based attacker must obtain IP addresses and locations of mobiles to orchestrate a signaling DoS attack. This problem was addressed in [6] and it was shown that it is possible to identify a sufficient number of IPs to launch an attack at a particular location.

The attack proposed in [7] is also network initiated, just like the one discussed in [1], wherein the attacker uses up all Temporary Flow Identifiers (TFI) of base stations under attack to deny service to mobiles in their coverage area. It was shown that an attacker needs just 200 Kbps bandwidth to degrade service in an area the size of Manhattan. The same attack method can also be used to deny service by congesting the PRACH channel [7]. This attack was also discussed in [3] along with several other types of signaling DoS attacks that are possible in cellular networks. A similar attack, which can also be launched from an Internet host, can deny service to legitimate users by congesting the paging channel [10].

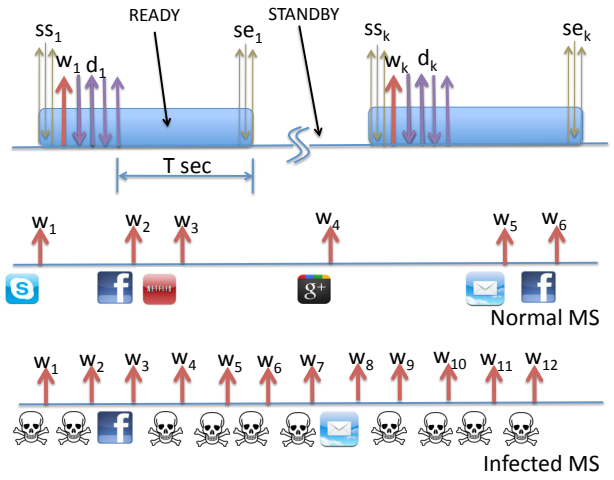

Fig. 1. First timeline shows wake-up packets (w_i) in red color. Wake-up packets are packets that cause MS to transition from STANDBY to READY mode. Signaling messages (ss_i and se_i) are generated whenever MS transitions from STANDBY to READY mode (or vice versa). Wake-up packets are immediately followed by normal IP data packets (d_i). Second timeline illustrates wake-up packets generated by a mix of applications installed on a typical MS. Third timeline illustrates increased wake-up activity due to the presence of a rogue application.

III. Detection Methodology

We define a wake-up packet as an IP packet whose arrival triggers the mobile station to transition from STANDBY mode to READY mode (see Figure 1). The MS is in STANDBY mode in the time period immediately preceding the arrival of any wake-up packet (by definition). Every wake-up packet is preceded by a time period of at least T sec (STANDBY timeout) when no IP packets are exchanged. A wake-up packet is either a downlink (Internet to MS) or uplink (MS to Internet) IP packet. A mobile based worm increases uplink wake-up packet initiated signaling, whereas, a Internet based port-scan increases downlink wake-up packet initiated signaling. Since we are only interested in detecting MS based worms (or rogue applications), downlink wake-up packets were ignored in this study. We examine selected features of uplink wake-up packets, like their frequency, destination IP, destination port, length, etc., to detect signaling DOS attacks. Our detection algorithm is based on the hypothesis that wake-up packets generated by a rogue application have a peculiarity that differentiates them from normal wake-up packets generated by a mix of applications installed on a typical MS.

Given a IP packet data set, which includes a series of n consecutive uplink wake-up packets generated by a particular MS, our objective is to classify this MS as normal or malicious. The list of features and a method for training the classifier are discussed below. The values of all the features listed below are determined from a data set of n consecutive wake-up packets and their responses.

A. Feature vector

1) Normalized destination IP entropy: If destination IP of wake-up packets is a random variable, then it is expected that the uncertainty associated with destination IPs chosen by an attacker will differ from that of a typical MS. For example, when an attacker sends ping wake-up packets to a fixed destination every few seconds, uncertainty associated with destination IPs is expected to be much lower than that of regular wake-up traffic. Alternatively, if the attacker pings random destination IPs, the uncertainty is expected to be unusually high. Let k be the number of unique destination IPs, and p_i the probability of observing IP i, then the normalized destination IP entropy is calculated as follows.

$$H_0 = \frac{H}{H_{max}} = \frac{-\sum_{i=1}^{k} p_i \ln p_i}{\ln k} \quad (1)$$

2) Normalized destination port entropy: The value of this feature is also obtained using Equation 1, wherein, k is the number of unique destination ports and p_i is the probability of observing destination port i in the wake-up packet trace. It is expected that the uncertainty of destination ports in the attack traffic will either be unusually high or unusually low. Destination port entropy is used as a feature here because it quantifies this uncertainty.

3) Normalized source port entropy: Similarly, the value of this feature is also obtained from Equation 1. The same argument on quantifying the uncertainty holds here as well.

4) Normalized packet length entropy: The value of this feature is obtained in the same way as features 1, 2 and 3 and for the same reasons as well.

5) Normalized wake-up rate: An attack application is expected to increase the MS wake-up rate and an increased rate for a prolonged period of time is indicative of the presence of a malicious application. The wake-up rate is calculated by finding the (exponential) average rate of n consecutive wake-up packets transmitted by the same MS. This wake-up rate is normalized by dividing it by the maximum possible wake-up rate of $1/T$.

6) Variance of inter wake-up times: If an attack application generates IP packets with a constant inter-arrival times, the resultant unusual periodicity of wake-up packets can be detected using variance of inter wake-up times. Normal wake-up traffic is not periodic and has higher variance than periodic attack traffic.

7) Response-request ratio: If the attack application sends wake-up requests to random IP addresses, all of these destination hosts are unlikely to respond, thus shrinking the average response-request ratio to an usually low value. Typical applications on the other hand communicate with legitimate hosts that respond to these requests.

B. Supervised Learning

The training data was extracted from one week long trace of IP packets generated by a set of 62 smartphones. It was assumed that none of these smartphones were infected with a connection state signaling worm/application. We believe that this is a valid assumption because to the best of our knowledge such an application does not exist. Training data for the two classes, i.e., normal MS and Malicious MS, was obtained as follows.

1) Normal MS: Given a window size n (e.g., $n = 50$), a set of n consecutive wake-up packets transmitted by a smartphone were used to form one labeled sample feature vector. If a smartphone transmitted less than n wake-up packets during the one week measurement period, then all of these samples were ignored. However, if a smartphone transmitted at least $m \times n$ wake-up packets (m is an integer) during the measurement period, then m labeled sample feature vectors were extracted for this MS.

2) Malicious MS: Since attack traffic samples are not available, we wrote our own attack application and collected pcap traces for 6 different types of attacks. The attack application was installed on a Linux server and all packets exchanged by this application were captured in the pcap file. The attack pcap was then merged with the one week long pcap of packets exchanged by an actual smartphone. When merging, the IP address of Linux server (attacker) was replaced with that of the smartphone, and timestamps in the attack trace were changed so that the attack traffic was interleaved with smartphone traffic. This merging was necessary to emulate the scenario where a rogue application resides on the MS, and the MS sends attack traffic in addition to the normal traffic generated by a regular mix of applications. Wake-up packets were then extracted from the merged pcap by locating IP packets that immediately follow a silence period longer than 5 seconds (STANDBY timeout value T is assumed to be 5 seconds). Each of the 6 attack traces were then merged with each of the 62 MS traces, resulting in 6×62 wake-up traces. All traces that had less than n wake-up packets were then removed. If a trace had more than $m \times n$ wake-up packets (m is an integer) then m labeled feature vectors were extracted from the trace.

Let b be the number of feature vectors that are labeled "normal MS" (b is a function of n and the data size). Since we generated a large number of attack packets, number of feature vectors labeled "malicious MS" were typically larger than b. Exactly b feature vectors were randomly selected from the set of all feature vectors labeled "malicious MS" (to avoid bias). This way, $2b$ labeled samples were available to train and test the support vector machine (SVM) classifier.

We decided to use SVM because it is a discriminating, inherently non-linear (when kernel is applied) approach and there is no need to make assumption of data parameters (distribution parameters) [9]. Moreover, since SVM solves a convex optimization problem, the decision boundary is consistent and there is no danger of converging to local minima, as is with other classifiers such as Neural networks. Supervised learning is used in this work as proof of concept and the performance of SVM may further improve with better parameter search.

TABLE I
BASIC ATTACK TYPES

Attack	Prot.	D. IP	D. Port	S. Port	Len.	Time
A1	R	R	R	R	R	R1
A2	TCP	S	F	S	0	R2
A3	ICMP	F	-	-	F	F(60s)
A4	TCP	F	R	F	0	R2
A5	UDP	S	F	R	R	F(100s)

IV. RESULTS

A. Data Sets

Measurement data set D_N includes one week of mobile phone traffic generated by 62 smartphones. We installed a traffic sniffer at the University edge router and captured all packets sent or received by these 62 mobiles phones over a period of one week. Note that the sniffer did not capture any packets when the students who own these smartphones were off campus (e.g. at night). To account for this, all inter wake-up times longer than 8 hours were removed from the analysis. However, when the students were on campus, we believe that all traffic generated by their smartphones was captured in our traces. In India, 3G plans are still very expensive and most students do not enable 3G on their phones. Instead the students prefer to stay connected to the University WiFi network when they are on campus.

We generated 5 different types of attacks, $\{A1, A2, ..., A5\}$ (see Table I for details), using a custom made attack application. Each attack pcap was then merged with data set D_N to yield the data set D_{Ai} ($i \in \{1, 2, ..., 5\}$). Each of the 5 merged data sets D_{Ai}, has 62 attack traffic samples, one from each user. The details of the 5 different attacks are listed in Table I. Here, R represents random, S is short for set, and F represents fixed. As an example, in attack 1, the malicious application sent TCP, UDP or ICMP packets that had random destination IPs, destination ports, source ports, and length. Packet inter-arrival times were obtained from a two state markov chain, the two states being "Low Rate" and "High Rate". For attacks with inter-arrival times labeled R2, when the system is in "Low Rate" state, inter-arrival times are uniformly distributed between 90 sec and 110 sec (50 sec and 70 sec for R1), and when the system is in "High Rate" state, inter-arrival times are uniformly distributed between 5 sec and 15 sec (5 sec and 15 sec for R1). When in a state, the probability of staying in that state is 0.95 and the probability of transitioning to the other state is 0.05. As another example, attack 2 was a TCP SYN attack, where the TCP packets had fixed source and destination ports, the payload length was 0, and inter-arrival times were random (from markov chain model). The destination IPs were randomly drawn from a set of ten responsive IPs (e.g., google.com). Attack 5 consisted of IP packets with destination UDP port 53 (DNS) sent to one of the commonly used DNS servers (e.g., google DNS).

In another type of attack ($A6$), our malicious program established a TCP connection with a web server and retrieved the webpage using linux wget. The inter wget time was determined using the two state markov chain model described above. In "high rate" state the inter-arrival times were uniform between 5 sec and 15 sec, and in "low rate" state the inter-arrival times were uniform between 50 sec and 60 sec. Web server was randomly selected from a list of ten well known websites (e.g., facebook.com). The resultant attack pcap was then merged with the 62 users' pcap to form data set D_{A6}.

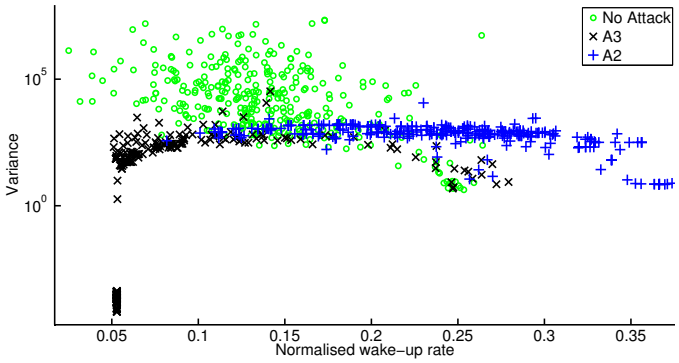

Fig. 2. Features wake-up rate and variance of inter wake-up times for normal MS traffic and for attacks A2 and A3 ($n = 50$ for all values in this graph)

B. Feature Vector

Figures 2, 3 and 4 illustrate the values of various features for several different types of attacks. The "No Attack" data shown in Figure 2 is from the data set D_N, i.e., original trace without any attacks. Both variance and rate span over a large range of values, as expected, because different users exhibit different and random usage behavior. Most of the attack A3 samples are clumped in the lower left corner of the graph. The low values of variance and fixed value of rate can be explained by the fact that inter-packet time was a constant value (60 sec) in attack A3. Some of the samples from attack A3 have higher variance because attack packets were mixed with normal user traffic. Clearly none of these higher variance samples should have a wake-up rate smaller than the attack rate of 1/60. In attack A2, wake-up rate spans a large range of values because the value of inter packet time was determined using a two state markov chain. Similarly, the "No Attack" data spans a large range of destination IP and destination port entropy values, as seen in Figure 3. Since attack A4 uses fixed destination IP and random destination port, normalized IP entropy is low and normalized port entropy is high. On the other hand, attack A1 has high values for both entropies because both destination IP and destination port were random. Similarly, Figure 4 illustrates the features source port entropy and packet length entropy. Attack A1 has high values of packet length entropy and source port entropy because both values are random. On the other hand, since both source port and packet length are fixed in attack A4, both entropies have values smaller than one.

C. Performance Evaluation

Given the normal MS data set D_N, the first step was to extract wake-up packets for each of the 62 MS. Let b_i be

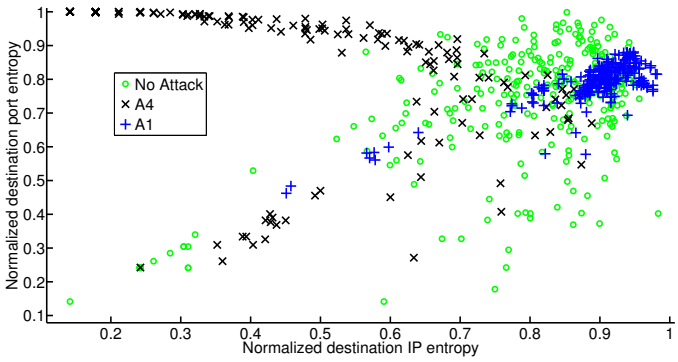

Fig. 3. Features normalized destination IP entropy and normalized destination port entropy of wake-up packets for normal MS traffic and for attacks A1 and A4 ($n = 50$ for all values in this graph)

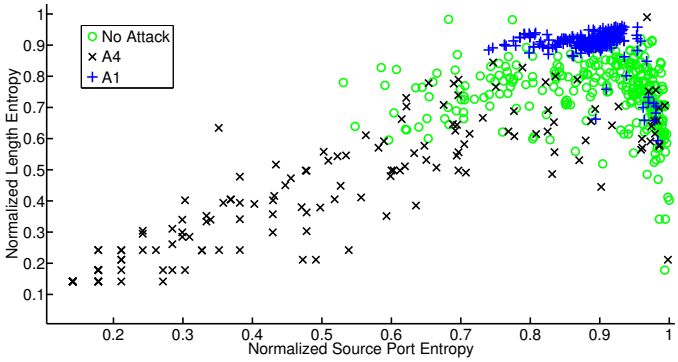

Fig. 4. Features normalized source port entropy and normalized length entropy of wake-up packets for normal MS traffic and for attacks A1 and A4 ($n = 50$ for all values in this graph)

the total number of segments from user i, where segment is defined as a set of n consecutive wake-up packets. There are a total of b_s segments in the data set, where, $b_s = \sum_{i=1}^{62} b_i$. These b_s segments are used to calculate b_s feature vectors labeled "Normal MS". Similarly, for any attack data set (e.g., D_{A1}), each user i has a_i segments, and there are a total of a_s segments. These a_s segments are used to calculate a_s feature vectors labeled "Malicious MS". In our data sets, a_s was always greater than b_s, and to remove the training bias, b_s segments were randomly selected from the a_s attack segments and used for training and classification. K-fold cross validation was then used, wherein, $2b_s - k$ out of the $2b_s$ labeled feature vectors were used to train the classifier, and the remaining k were used to test the classification accuracy. It is clear from the evaluation results shown in Figure 5 that the detector correctly identifies more than 90% of the attack segments as attacks, in all of the six different attack types. Moreover, less than 10% of the normal traffic segments are misclassified as attack segments. The probability of detection of attacks A4 and A6 is slightly better than all other attacks. This can be explained by the fact that A4 uses fixed source ports and A6 uses fixed destination port. Probability of detection of attack 5 is slightly worse than all other attacks, probably because of the low transmission rate.

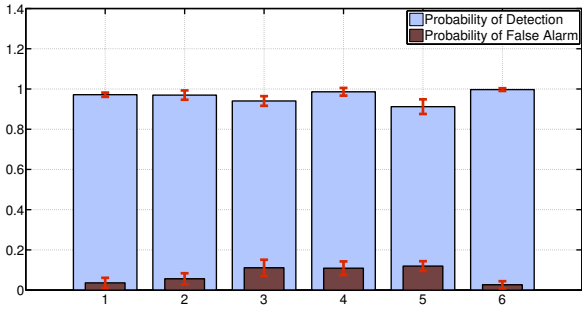

Fig. 5. Probability of detection and false alarm for various attacks. Window size $n = 50$ and $k = 5$ (k-fold cross validation)

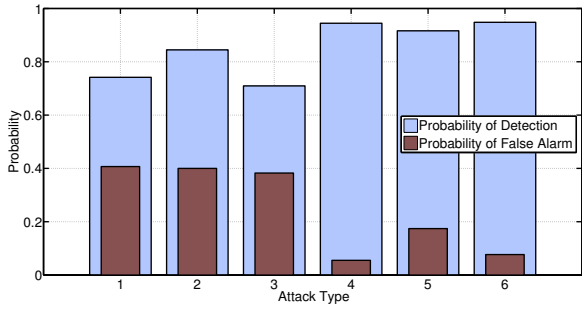

Fig. 6. Detector performance when the classifier has not been trained to detect the attack under evaluation (n was fixed at 50 samples)

TABLE II
PERFORMANCE AS A FUNCTION OF ATTACK RATE FOR ATTACK A4

Rate	[5,15].	[35,45]	[75,85]	[115,125]	[135,145]
Mean P_D	1	1	1	1	1
Std. P_D	0	0	0	0	0
Mean P_F	0.0028	0.0024	0.0032	0.0029	0.0029
Std. P_F	0.0063	0.0054	0.0072	0.0066	0.0064

One of the criticisms of the above approach is that the classifier was already trained for the attack that it detected. However, it may not always be possible to anticipate and train the classifier in advance for all types of attacks. We conducted an experiment to understand the performance when the detector hasn't been trained for the attack under evaluation. When testing the performance w.r.t. a particular attack, the detector was trained using all (five) other attacks except the attack under evaluation. It is clear from the results in Figure 6 that the performance drops significantly when the classifier has no prior training about the attack. However, attacks A4 and A6 were detected with very good performance without any prior training. This may be because both these attacks are very similar to each other and a classifier trained with just one of the two attacks could work well with both attacks.

We conducted another experiment to understand the effect of packet rate on detector performance. Table II illustrates detector performance as a function of the attack rate. We generated five new attacks of Type A4 and changed the packet rate in each attack. The inter-arrival times were uniformly distributed with low and high limits listed in Table II. Both detection probability and false alarm probability were found to

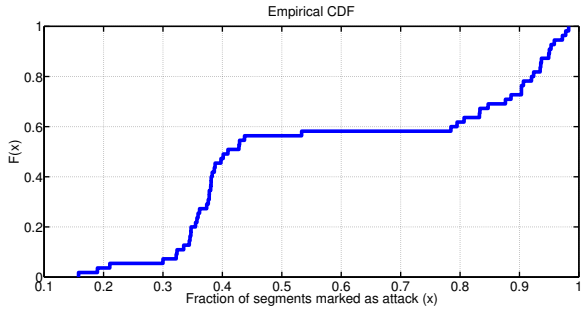

Fig. 7. CDF of the fraction of segments correctly marked as attack

be independent of the attack rate. This performance is slightly better than the one reported in Figure 5 because inter-arrival times had higher variation (markov chain) in the latter case.

In another experiment, the classifier was trained using labeled samples from 50% of the users. Data segments from each of the remaining users were then tested for the presence of an attack. Let attack user i have a total of a_i segments, then if x% of these a_i segments were correctly marked as an attack, then we note the number x against user i. This procedure was repeated multiple times to obtain the CDF shown in Figure 7. It is clear that more that 80% of segments were correctly identified as an attack 40% of the time, and more than 30% of the of the segments were correctly identified 90% of the time.

Since most of the attacks in our initial attack set were detected with reasonably good accuracy, we designed a more sophisticated attack that attempts to evade detection by replaying previously transmitted packets. Since the attack traffic looks exactly like legitimate traffic, it would be difficult for the system to detect this attack. The malicious application first captures all packets transmitted by its host MS for some period of time (e.g., one hour). It then starts replaying the captured packets in the background effectively doubling the MS initiated wake-up rate. As new user packets are generated, the malicious application keeps appending these packet to its replay-able pcap file, and replays them at an appropriate playback time. We generated this attack by replaying data files generated by each of the 62 different MS. The classifier was trained using the 6 attacks A1 through A6, with the hope that it would be able to detect the packet replay attack. However, our detector failed miserably, with a detection probability of 0.538 and false alarm probability of 0.492.

V. Conclusion

The main contribution of this work is a new method for detecting MS initiated signaling DoS attacks from IP packet traces. This method examines characteristics of wake-up IP packets to infer the presence of a malicious application. The classifier was trained and tested using 6 different types of attacks and the performance was found to be very good for all 6 attacks. The classifier was even tested with attacks for which it had no prior training and again the performance was satisfactory. Finally, the classifier was tested with a packet replay attack, and the detector failed to detect this type of attack.

Service providers can protect their network by installing a system based on this classifier anywhere in the data path, i.e., MS, base station, gateway, etc. The classifier can identify and report the infected mobile stations in near real-time. The infected MSs can be quarantined using several methods, e.g., Mobile Equipment ID (MEID) blocking in Home Subscriber Server (HSS), Mobility Management Entity (MME) initiated network detach [16], etc.

Acknowledgment

The authors would like to thank Shweta Sachan and Anoop Singh for facilitating this project; and Kuldeep Yadav, Samarth Bharadwaj, Pandarasamy Arjunan, for their input and comments.

References

[1] Patrick P. C. Lee, Tian Bu, Thoma Woo, On the Detection of Signaling DOS Attacks on 3G Wireless Networks, IEEE INFOCOM, 2007

[2] K. Wang and S. Stolfo, Anomalous payload-based network intrusion detection, Proceedings of Recent Advance in Intrusion Detection (RAID), Sept. 2004.

[3] Fabio Ricciato, Angelo Coluccia, Alessandro DAlconzo, A review of DoS attack models for 3G cellular networks from a system-design perspective, Computer Communications, 33 (2010) 551558.

[4] Patrick Traynor, Michael Lin, Machigar Ongtang, Vikhyath Rao, Trent Jaeger, Patrick McDaniel and Thomas La Porta, On Cellular Botnets: Measuring the Impact of Malicious Devices on a Cellular Network Core, CCS09, November 913, 2009, Chicago, Illinois, USA.

[5] Zhizhong Wu, Xuehai Zhou, Feng Yang, Defending against DoS Attacks on 3G Cellular Networks Via Randomization Method, 20IO International Coriference on Educational and Information Technology (ICEIT 2010)

[6] Zhiyun Qian, Zhaoguang Wang, Qiang Xu, Z. Morley Mao, Ming Zhang, Yi-Min Wang, You Can Run, but You Cant Hide: Exposing Network Location for Targeted DoS Attacks in Cellular Networks, In Proceedings of 17th Annual Network and Distributed System Security Symposium (NDSS) 2012, San Diego, CA

[7] Patrick Traynor, Patrick McDaniel and Thomas La Porta, On Attack Causality in Internet-Connected Cellular Networks, In Proc. of USENIX Security, 2007.

[8] Zhaoguang Wang, Zhiyun Qian, Qiang Xu, Z. Morley Mao, Ming Zhang, An Untold Story of Middleboxes in Cellular Networks, SIGCOMM11, August 1519, 2011, Toronto, Ontario, Canada.

[9] Chang, Chih-Chung and Lin, Chih-Jen, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology, Vol. 2, No. 3, 2011, pp. 27:1–27:27

[10] J. Serror, H. Zang, and J. C. Bolot, Impact of Paging Channel Overloads or Attacks on a Cellular Network, In WiSe, 2006.

[11] Patrick P. C. Lee, Tian Bu, and Thomas Woo, "On the Detection of Signaling DoS Attacks on 3G/WiMax Wireless Networks," Computer Networks, Elsevier, 53(15), pp. 2601-2616, October 2009.

[12] Tellabs Smartcore 9100 WIMAX gateway, http://www.tellabs.com/solutions/mobilepacketcore/

[13] Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, Deborah Estrin, "Diversity in Smartphone Usage," MobiSys10, June 1518, 2010, San Francisco, California, USA.

[14] Traynor, Patrick and Lin, Michael and Ongtang, Machigar and Rao, Vikhyath and Jaeger, Trent and McDaniel, Patrick and La Porta, Thomas, On cellular botnets: measuring the impact of malicious devices on a cellular network core, Proceedings of the 16th ACM conference on Computer and communications security, CCS 2009, Chicago, Illinois, USA.

[15] Collin Mulliner and Jean-Pierre Seifert, Rise of the iBots: 0wning a telco network, 5th International Conference on Malicious and Unwanted Software (MALWARE), 19-20 Oct. 2010, Nancy, Lorraine

[16] 3GPP Technical Specification 23.401, General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access (Release 11)