SEVENTH FRAMEWORK PROGRAMME	**SEVENTH FRAMEWORK PROGRAMME** **Trustworthy ICT**

Project Title:

Enhanced Network Security for Seamless Service Provisioning in the Smart Mobile Ecosystem

Grant Agreement No: 317888, Specific Targeted Research Project (STREP)

DELIVERABLE

D4.2 Anomaly detection based on real-time exploitation of billing systems

Deliverable No.	**D4.2**		
Workpackage No.	**WP4**	Workpackage Title	**Anomaly detection using control plane data**
Task No.	**T4.2**	Task Title	**Anomaly detection based on real-time exploitation of billing systems in billed and prepaid environments**
Lead Beneficiary	**ICL**		
Dissemination Level	**PU**		
Nature of Deliverable	**R**		
Delivery Date	**31 October 2014**		
Status	**F**		
File name	**NEMESYS_Deliverable_D4.2.pdf**		
Project Start Date	**01 November 2012**		
Project Duration	**36 Months**		

Authors List

Author's Name	Partner	E-mail Address
Leading Author/Editor		
Omer H. Abdelrahman	ICL	o.abd06@imperial.ac.uk
Co-Authors		
Anastasios Drosou	CERTH	drosou@iti.gr
Erol Gelenbe	ICL	e.gelenbe@imperial.ac.uk
Gokce Gorbil	ICL	g.gorbil@imperial.ac.uk
Vasilios Mavroudis	CERTH	mavroudis@iti.gr

Reviewers List

Reviewer's Name	Partner	E-mail Address
Laurent Delosières	HIS	ldelosieres@hispasec.com
Madalina Baltatu	TIIT	madalina.baltatu@it.telecomitalia.it
Luciana Costa	TIIT	luciana.costa@it.telecomitalia.it

Contents

List of Figures

List of Tables

Abbreviations

CN	Core Network
CDR	Charging Data Record
DCH	Dedicated Channel
DNS	Domain Name System
DoS	Denial of Service
DPI	Deep Packet Inspection
EOF	Entropy Outlier Factor
FACH	Forward Access Channel
FP	False Positive
FPR	False Positive Rate
GGSN	Gateway GPRS Support Node
HTTP	Hypertext Transfer Protocol
IM	Instant Messaging
IP	Internet Protocol
KPI	Key Performance Indicator
LTE	Long Term Evolution
M2M	Machine-to-Machine
MME	Mobility Management Entity
MNO	Mobile Network Operator
NAT	Network Address Translation
PC	Personal Computer
PCH	Paging Channel
PGW	Packet Data Network Gateway
QoS	Quality of Service
RLC	Radio Link Control
RNC	Radio Network Controller
RNN	Random Neural Network
ROC	Receiver Operating Characteristic
RRC	Radio Resource Control
SGSN	Serving GPRS Support Node
SGW	Serving Gateway
SMS	Short Message Service
SMSC	Short Message Service Centre
TCP	Transmission Control Protocol
TP	True Positive

TPR	True Positive Rate
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
VoIP	Voice over IP

Abstract

Mobile malware and mobile network attacks are becoming a significant threat that accompanies the increasing popularity of smart phones and tablets. Thus in this deliverable we propose two anomaly detection algorithms that use traffic measurements and billing meta(data) in order to identify malicious or misbehaving mobile devices. The first algorithm is based on measuring various quantities describing the activity of a user, applying different statistical methods to compute features that capture both instantaneous and long term changes in behaviour, and using a random neural network to fuse the information gathered from the individual features in order to detect anomalies in real-time. The second approach uses graph based descriptors to model billing records, where vertices in the graph represent users and services, while edges correspond to communication events. Anomaly detection is then performed by extracting features from the graph, and applying a supervised learning technique to discriminate between normal and anomalous users. The proposed methods are evaluated on two datasets from our mobile network simulator, representing threats that affect mobile users and networks.

1 Introduction

Mobile malware is emerging as a significant threat due to the increasing popularity of smart phones and tablets, which now run fully-fledged operating systems on powerful hardware and feature multiple interfaces and sensors. Personal computers (PCs) are no longer the dominant computing platform, and in fact the global shipments of smart phones alone have exceeded those of desktop computers since 2011 [18,19]. Further, with the accelerated adoption of 4G technologies including mobile WiMAX and Long Term Evolution (LTE), cellular devices will become the primary means of broadband Internet access for many users around the world. Indeed, while 4G capable devices represent only 0.9% of all global mobile connections observed in the Internet during 2012, they already account for 14% of the mobile data traffic [2]. As more and more people move from PCs to handheld devices, cyber criminals are naturally shifting their attention to the mobile environment. This trend is fuelled by the availability of off-the-shelf malware creation tools [14], and the proliferation of mobile software marketplaces which enable the distribution of malicious applications to potentially millions of users [45]. Such mobile malware can attempt to snoop and steal saleable information, generate revenue by calling premium rate numbers, or perform other malicious activities that could directly impact the availability and security of cellular networks.

1.1 Motivation

Despite this growing challenge, most operators have been reactive rather than proactive towards these security threats [10] and investments in detection and mitigation techniques specific to mobile networks were mostly implemented when a problem occurred. However, there has been recently a growing interest among mobile carriers [12] and vendors [49] to develop network based anomaly detection systems which offer a number of advantages over client-side security solutions:

- *Footprint*: Network level analysis does not incur additional monitoring, processing, storage or communication overhead to mobile devices which, although may have sufficient computational power nowadays, are battery and bandwidth constrained.

- *Efficacy*: Most users are not aware of the growing security risks with mobile devices [11], while others may be reluctant to use traditional antivirus solutions because of their inherent power-hungry characteristics, negative impact on performance, and potential false positives caused by novel behaviour. Since large scale malware infections pose a significant threat to the availability and security of cellular networks, it is in the interest of operators to ensure that users are well protected even if they are not security conscious.

- *Agility and ease of deployment*: It can be modified easily without requiring users to install patches and updates, whereas an on-device solution must support a large number of mobile operating systems and hardware platforms.

- *Trustability*: Network based detection is not vulnerable to exploits that allow malware to circumvent client-side security, but could be bypassed by stealthy attackers concealing their activities to appear as normal users.

- *Coverage*: It provides a broad view of malicious activities within a carrier's network, enabling the detection of zero-day attacks that misuse detection technologies would not recognise. Indeed, while some attacks could be detected by examining per user behaviour, others become more visible when specific features are aggregated across multiple users (e.g., data exfiltration attempts by a single server). Furthermore, the operator's access to charging data records (CDR) can facilitate the detection of attacks that have direct impact on billing data. In this respect, statistics extracted from billing information offer a new dimension in the analysis of anomalies, but the approach is also limited in scope in the sense that it considers only features related to billing and communication via the cellular network while on-device security systems could benefit from monitoring other behavioural data such as system calls, power consumption, keystrokes, and communication over non-cellular interfaces.

Thus, analysis of mobile network traffic offers a complementary means for detecting both user and network targeted attacks, which could be used in conjunction with other NEMESYS client-side solutions such as the mobile honeypot [3, 4] and high interaction honeyclient [5], in order to improve detection performance. These tools which operators may deploy on their networks can constitute value-added services for communities of users, in the same way that banks use profiling to reduce credit card fraud.

There are, however, two main drawbacks of existing network-side security mechanisms which employ anomaly detection[1]. First, without proper and very specific training on

[1]Based on the experience of NEMESYS telecommunication partners.

the real network, existing tools produce a significantly high rate of false alarms, so much so that analysts usually prefer to switch them off. Second, the training process has to be repeated at least each time a change occurs in the network, e.g. a device is substituted or a new service is introduced, rendering the management cost of these solutions very high.

1.2 Objective of the Deliverable

In this deliverable, we develop anomaly detection mechanisms that mobile network operators (MNOs) may deploy in their networks in order protect their own infrastructure and defend users against malware. The approach is based on the analysis of anonymised CDR for both voice and Internet traffic, which are readily available to the operator and used for billing purposes. Such approach is easier to implement in a mobile network since it relies on the analysis of user plane (meta)data, in contrast to solutions that require changes to some of the network components and/or protocols. Furthermore, we address the aforementioned management overhead of network-side solutions by proposing a method to automate the configuration of the algorithms' parameters, for example based on key performance indicators generated by the network equipment.

1.3 Outline of the Deliverable

The rest of this deliverable is structured as follows. Section 2 provides background about the most common threats affecting mobile networks and users, and motivations for our CDR based algorithms. Section 3 develops a framework for real-time detection using random neural networks (RNN) [28,29], and presents experimental results validating the accuracy of the approach in identifying mobile devices that are contributing to signalling overloads. The results are complemented with a mathematical model that will allow us to optimise the performance of the signalling based detector of D4.1 [6], when combined with the present RNN algorithm as part of our future integration work. Section 4 describes an approach that utilises graph based descriptors to represent billing activities, from which features are extracted and used in order to identify anomalous users in two distinct scenarios: the above signalling storm, and SMS spam campaign. The datasets used in this deliverable have been generated by extending the mobile network simulator that we have developed within NEMESYS [6, 34], so as to include more realistic data plane models in addition to the signalling protocols of mobile networks. Finally, we provide our concluding remarks in section 5.

2 Background

This section presents an overview of the main threats against mobile users and services, and outlines the merits and limitations of existing anomaly detection approaches. The analysis is intended to motivate the attack models and algorithms that are developed later in the deliverable.

2.1 Network Level Threats and Mitigation

Mobile systems are challenged by mobile broadband requirements such as video streaming including 3D and playback, together with machine to machine (M2M) and vehicular communications. All these will demand a lower signalling overhead and better quality of service (QoS) to short payloads, at much higher traffic volumes and bandwidth requirements. The constant access to the cloud in order to offload, away from the mobile devices, the energy and computation critical applications creates yet another need for continuous and secure connectivity.

Unfortunately, mobile networks are vulnerable to signalling denial-of-service (DoS) attacks which overload the control plane through small but frequent communications that exploit vulnerabilities in the signalling procedures used, for example, in paging [59], transport of SMS [26], service requests [65] and radio resource control (RRC) [8,56]. Such attacks can be carried out either by compromising a large number of mobile devices, or from the Internet by targeting a hit list of mobile devices through carefully timed transmissions, and can seriously compromise the connections between a large set of mobiles and the services to which they are connected or are trying to connect.

Since security and uninterrupted connectivity in all mobile applications will become even more important, not just from the "nuisance" and QoS perspective, but also because it is expected that safety critical applications will transition to mobile devices and away from special purpose private networks or the commonly used fixed sensor networks, the whole issue of how network threats can be detected and mitigated will become even more important. Safety critical applications that will be accessed via mobile networks will include emergency management, smart metering, smart grid control, public transportation control systems (including railways), and electric vehicle charging networks.

M2M applications will in particular be quite vulnerable to such attacks since they

will not have the human-in-the-loop who can turn off a mobile when she sees something strange going on. Thus significant efforts need to be made to better understand the security liabilities and weaknesses of mobile connections and find new approaches to make them resilient and reliable in the face of malicious malware or malfunctioning applications.

2.1.1 RRC based Signalling Overload

In the context of UMTS networks, bandwidth is managed by the RRC protocol which associates a state machine with each user equipment (UE). There are typically four RRC states, in order of increasing energy consumption: IDLE, Paging Channel (cell_PCH), low bandwidth Forward Access Channel (cell_FACH), and high bandwidth Dedicated Channel (cell_DCH). We will refer hereafter to state cell_X as X. State promotions are triggered by uplink and downlink transmissions, and the move to FACH or DCH is determined by the size of the radio link control (RLC) buffer of the UE: if at any time the buffer exceeds a certain threshold in either direction, the state will be promoted to DCH. State demotions in states DCH, FACH and PCH are triggered by inactivity timers T_1, T_2 and T_3, respectively. LTE implements a simpler RRC state machine with two states: idle and connected. Frequent transitions between the RRC states can result in excessive signalling load, in order to allocate and deallocate radio resources, leading to performance degradations and outages. The RRC protocols in 3GPP standards are described in detail in D4.1 [6].

A number of recent studies have addressed the question of whether RRC signalling attacks are feasible in real networks, and three practical issues have been identified along with possible methods for dealing with them:

- *Inference of radio resource allocation policies*: Signalling DoS attacks require knowledge of the RRC policies used by the cellular operator, so that attack traffic could inflict substantial load on the control plane. Probing techniques for inferring the RRC state machine of operational networks have been developed in [13, 52].

- *Reachability and NAT traversal*: Mobile operators deploy firewalls and Network Address Translation (NAT) at network boundaries to protect their infrastructure and mobile users from unsolicited traffic from the Internet, by hiding the entire IP address space behind a single public IP address. This enables communication through edge routers only when a mobile initiates the data session, and any attempt by an external host to scan IP addresses inside the network will be unsuccessful. The authors in [53, 67] develop a mobile application and a probing technique allowing them to investigate middlebox policies used by 180 cellular carriers around

the world. Among other findings, the study reveals that 51% of the operators allow mobile devices to be probed from the Internet: some of them assign public IP addresses to mobile devices while others use private ones but allow IP spoofing or device-to-device probing within the network.

- *Exposing locations of mobile devices*: An attacker must be able to map IP addresses of mobile devices to a geographic area in order to launch a signalling DoS attack from the Internet. In [53], active probing is used to discover a combination of static and dynamic features, such as inactivity timer and minimum round trip time, enabling an attacker to identify a sufficient number of IP addresses in a particular location. It is also found that 80% of mobile devices keep their IP addresses for more than 4 hours, giving an attacker enough time to perform network measurements.

In practice, however, signalling DoS attacks have not been observed, which is likely due to lack of financial incentives for cyber criminals who would rather have the infrastructure functional in order to launch profitable attacks. Nevertheless the threat cannot be ignored as DoS attacks could be used, for example, to impair the competition or as a form of protest (hacktivism).

Signalling storms are similar to signalling DoS attacks, but they are mainly caused by misbehaving mobile applications that repeatedly establish and tear-down data connections [50] in order to transfer small amounts of data. Such "chatty" behaviour triggers repeated signalling to allocate and deallocate radio channels and other resources, and therefore has a negative impact on the control plane of the network [58]. There are a number of recent high profile cases, e.g. in Korea [25] and Japan [27], where large operators suffered major outages due to popular applications that constantly poll the network even when users are inactive. Ad based mobile monetisation is another culprit, shown to cause erratic spikes in signalling traffic [24]. Many mobile carriers have also reported [10] outages and performance issues caused by non-malicious but poorly designed applications, yet the majority of those affected followed a reactive approach to identify and mitigate the problem. It is expected that signalling storms will continue to pose challenges to operators, with the projected growth in mobile data [22] and the advent of M2M systems for which existing cellular networks are not optimised [1, 41, 60, 63]. Signalling storms could also occur as a byproduct of malicious activities that involve frequent communications. Indeed, a recent analysis of the traffic profiles of mobile subscribers in China [44] indicated a positive correlation between the frequency of resource-inefficient traffic and malicious activities in the network such as private data upload, billing fraud and TCP SYN flooding.

2.1.2 Existing Countermeasures

Signaling problems in mobile networks have a limited impact on the data plane and thus are difficult to detect using traditional intrusion detection systems which are effective against flooding type attacks. For Internet based attacks, a change detection algorithm using the cumulative sum method has been proposed in [42], where the signaling rate of each remote host is monitored and an alarm is triggered if this rate exceeds a fixed threshold. The use of a single threshold for all users, however, presents a trade-off between false positives and detection time, which can be difficult to optimise given the diversity of users' behaviour and consumption.

A supervised learning approach is used in [35] to detect mobile-initiated signalling attacks, whereby transmissions that trigger a radio access bearer setup procedure are monitored, and various features are extracted relating to destination IP and port numbers, packet size, and response-request ratio. In this deliverable, we develop algorithms to detect signalling anomalies using only billing-related information, thus simplifying their deployment in the network. Furthermore, the methods proposed in this deliverable can be integrated with the signalling based solution of D4.1 [6], so as to improve the overall detection performance of the system. The work in [39] considers the detection of SMS flooding attacks using low reply rate as the main indicator of malicious activities, which is likely to misclassify SMS accounts used for M2M communications, such as asset tracking and smart grid metering [48].

A general framework for anomaly detection is presented in [23], where time-series of one dimensional feature distributions are derived and change detection algorithms are applied to identify statistically significant deviations from past behaviour. While the method in [23] aims to identify large scale events by aggregating and analysing statistics from all mobile users, our algorithm in Section 3 follows a different approach whereby it is activated and configured based on key performance indicators (KPIs) (e.g. when signalling load exceeds a certain threshold), and as such it does not need to operate continuously. Furthermore, the aim of our method is to identify in real-time the users that are contributing to a problem rather than detect the problem itself. Nevertheless, we also develop in Section 4 a graph based algorithm that can be executed periodically in order to detect stealthy but non-critical malicious campaigns that may not affect KPIs in the mobile network.

2.2 Attacks Against Mobile Users

A recent report by Kaspersky Lab [46] revealed that the most frequently detected malware threats affecting Android operating system are (i) SMS Trojans which send mes-

sages without users' consent, (i) adware which displays unwanted advertisements, and (iii) root exploits which allow the installation of other malware or the device to become part of a botnet. While botnets are a well-known phenomenon in the wired Internet, the year 2012 saw the emergence of the first mobile botnets [47]. Mobile botnets pose interesting questions as to their capabilities and uses since smart mobile devices possess many abilities not present on a desktop computer. Such mobile botnets could be used to attack mobile users, e.g. SMS spam, which in turn may have a serious impact on the network functioning as described earlier. In the following, we discuss two approaches to deploying anomaly detection systems at the operator's network, and we review existing techniques that have been proposed in the literature.

2.2.1 Network based Detection

Recent work [43] has shown that mobile malware families are not different from their non-cellular counterparts, in the sense that they rely on the same Internet infrastructure to support their illicit operations, and share many behavioural characteristics such as host changes, growth patterns and so on. Thus traditional network based solutions which have been applied successfully in the wired domain could be also effective in detecting and mitigating mobile threats. The literature includes a number of proposals for enabling mobile operators to detect user targeted attacks:

- *DNS analysis*: Since malware typically uses DNS to retrieve IP addresses of servers, detecting and blacklisting suspicious domains can be a first step towards limiting the impact of malware [36, 43]. However, detection should not be based solely on historical data (i.e. known malicious domains), but also on behavioural characteristics that may differentiate normal and malicious traffic.

- *Content matching*: Uncommon header flags and syntactic matches in HTTP messages can be used as indicators of data exfiltration attempts [36], but this approach is not effective when end-to-end encryption is used, as it relies on extracting information from plain-text transmissions, and it also requires performing deep packet inspection (DPI) which may not be scalable.

- *CDR analysis*: One of the key characteristics of mobile communications pertains to the fact that the whole extent of exchanged traffic load is continuously monitored for billing and accounting purposes. Hence, it is expected that many malicious activities will have an evident impact on the CDR of the parties involved. In [48], communication patterns of SMS spammers are compared to those of legitimate mobile users and M2M connected appliances, showing evidence of spammer mobility, voice and data traffic resembling the behaviour of normal users, as well as

similarities between spammers and M2M communication profiles. Fuzzy-logic is used in [66] to detect SMS spamming botnets by exploiting differences in usage profiles, while in [68] SMS anomalies are detected through building normal social behaviour profiles for users, but the learning technique fails to identify transient accounts used only for malicious purposes. The work in [38] uses a bi-partite graph to represent voice calls from domestic to international numbers, and proposes a Markov clustering algorithm to detect and classify voice-related fraud; the analysis shows that different fraud activities, such as those carried by malicious applications with automated dialler or through social engineering tactics, exhibit distinct characteristics.

A somewhat different approach that can be used by operators to identify malware offers a trade-off between network-level analysis and on-device security: the former imposes zero-load on the device but limits its scope to cellular data, while the latter is able to utilise internal mobile events for detection but may be resource hungry. This *hybrid* approach uses a thin mobile client to extract relevant features from the device [17, 57] which are then offloaded to the network or the cloud for inspection. Although this approach offers heavy-weight security mechanisms to devices that may not otherwise have the processing power to run them, it still requires continuous monitoring, some processing and frequent communication with a remote service. Moreover, this approach can only protect those users that install a security application, and requires a large number of subscribers in order to identify large-scale events, while network based detection does not require the user to do anything as all detection is performed using data available to the network operator.

3 RNN based Online Anomaly Detection

In this section we present a random neural network (RNN) [28, 29] based algorithm for detecting mobile signalling anomalies in real-time using charging data records (CDR). The algorithm uses supervised learning to distinguish between normal and abnormal behaviour, and is able to identify quickly when a mobile device generates excessive control messages without directly monitoring the signalling plane. In contrast to signalling based solutions which would require modification to cellular network equipment or protocols, the algorithm is designed to run on the core network using standard monitoring tools. Our motivation behind the use of a supervised learning approach is that it is suited for detecting threats that are well-understood, which include signalling storms whose characteristics and root causes have been analysed thoroughly in D4.1 [6].

The rest of this section is structured as follows. In section 3.1 we give a brief summary of the RNN model as applied to our problem of distinguishing between normal and misbehaving mobile devices. Section 3.2 presents the core of the detection technique, including the decision making process, and provides a detailed description of the choice of input features, and the parameters that can influence the performance of the algorithm. In section 3.3, we evaluate our detection mechanism using data generated by the mobile network simulator developed within NEMESYS; we describe the user and attack models, and present some experimental results. Then section 3.4 presents a mathematical model for evaluating and optimising the performance of our algorithm when used in conjunction with the signalling based detector developed in D4.1. Finally, we summarise our findings in section 3.5.

3.1 Background

The RNN is a biologically inspired computational model, introduced by Gelenbe [28], in which neurons exchange signals in the form of spikes of unit amplitude. In RNN, positive and negative signals represent excitation and inhibition respectively, and are accumulated in neurons. Positive signals are cancelled by negative signals, and neurons may fire if their potential is positive. A signal may leave neuron i for neuron j as a positive signal with probability p_{ij}^{+}, as a negative signal with probability p_{ij}^{-}, or may depart from the network with probability d_i, where $\sum_j [p_{ij}^{+} + p_{ij}^{-}] + d_i = 1$. Thus, when

neuron i is excited, it fires excitatory and inhibitory signals to neuron j with rates:

$$w_{ij}^{+} = r_i p_{ij}^{+} \geq 0, \quad w_{ij}^{-} = r_i p_{ij}^{-} \geq 0,$$

where:

$$r_i = (1 - d_i)^{-1} \sum_j [w_{ij}^{+} + w_{ij}^{-}].$$

The steady-state probability that neuron i is excited is given by $q_i = \frac{N_i}{D_i}$ where:

$$N_i = \Lambda_i + \sum_j q_j w_{ji}^{+}, \quad D_i = \lambda_i + r_i + \sum_j q_j w_{ji}^{-}$$

with Λ_i and λ_i denoting the rates of exogenous excitatory and inhibitory signal inputs into neuron i, respectively.

A gradient descent supervised learning algorithm for the recurrent RNN has been developed in [29]. For a RNN with n neurons, the learning algorithm estimates the $n \times n$ weight matrices $\mathbf{W}^{+} = \{w_{ij}^{+}\}$ and $\mathbf{W}^{=}\{w_{ij}^{-}\}$ from a training set comprising K input-output pairs $(\mathbf{X}, \mathbf{Y})$. The set of successive inputs to the algorithm is $\mathbf{X} = (\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(K)})$, where $\mathbf{x}^{(k)} = (\mathbf{\Lambda}^{(k)}, \lambda^{(k)})$ are the pairs of exogenous excitatory and inhibitory signals entering each neuron from outside the network:

$$\mathbf{\Lambda}^{(k)} = (\Lambda_1^{(k)}, \cdots, \Lambda_n^{(k)}), \quad \lambda^{(k)} = (\lambda_1^{(k)}, \cdots, \lambda_n^{(k)}).$$

The successive desired outputs are $\mathbf{Y} = (\mathbf{y}^{(1)}, \cdots, \mathbf{y}^{(K)})$, where the k-th vector $\mathbf{y}^{(k)} = (y_1^{(k)}, \cdots, y_n^{(k)})$, whose elements $y_i^{(k)} \in [0, 1]$ correspond to the desired output values for each neuron. The training examples are presented to the network sequentially, and the weights are updated according to the gradient descent rule to minimise an error function:

$$E^{(k)} = \frac{1}{2} \sum_{i=1}^{n} a_i [q_i^{(k)} - y_i^{(k)}]^2, a_i \geq 0.$$

The update procedure requires a matrix inversion operation for each neuron pairs (i, j) and input k which can be done in time complexity $O(n^3)$, or $O(mn^2)$ if m-step relaxation method is used, and $O(n^2)$ for feed-forward networks. The RNN has been successfully applied to several engineering problems [64] including pattern recognition, classification and DoS attack detection [32, 51].

3.2 Detection of Signalling Anomalies

The RNN based anomaly detection algorithm monitors the activity of each user, and measures a set of expressive features that describe various characteristics of the user's behaviour. Time is divided into slots, each of duration Δ seconds, in which summary statistics such as the mean and standard deviation of several quantities related to the activity of the user are collected. The algorithm stores the most recent w set of measurements, and use them to compute the current values of the input features; i.e. the features for time slot τ are computed from measurements obtained for time slots $\tau, \tau-1, \cdots, \tau-(w-1)$ so that the observation window of the algorithm is $W = w\Delta$. We discuss in section 3.2.2 the selection of the algorithm's parameters and how they influence its performance.

Let $z[\tau]$ denotes a measured or calculated quantity for time slot i, then the i-th input feature $x_i[\tau]$ can be obtained by applying a statistical function f_i as follows:

$$x_i[\tau] = f_i(z[\tau], z[\tau-1], \cdots, z[\tau-w-1]).$$

Hence, by employing different operators f_i on different statistics z stored for the observation window of w slots, it is possible to capture both instantaneous (i.e. sudden) and long-term changes in the traffic profile of a user. In our work, we have used a number of simple statistical functions such as the mean and standard deviation of z across the entire window, and also an exponential moving average filter in which the current feature is computed as:

$$x_i[\tau] = \alpha x_i[\tau-1] + (1-\alpha)z[\tau],$$

where α is some constant $0 < \alpha < 1$ typically close to 1, with higher values discounting older observations faster. An important concept from information theory that we have also used in our approach is *entropy* which is a measure of the uncertainty or unpredictability in the data:

$$x_i[\tau] = -\sum_{t=\tau-w-1}^{\tau} p_{z[t]} \log p_{z[t]},$$

where $p_{z[t]}$ is the probability of observing data item $z[t]$ within the window, which can be estimated from the histogram of the data. Entropy is typically interpreted as the minimum number of bits required to encode the classification of a data item, thus a small entropy indicates deterministic behaviour which is often associated with signalling anomalies [27, 44, 54].

Once the input features for a slot have been computed, they are fused using a trained feed-forward RNN architecture such as the one presented in Fig. 3.1 to yield the final

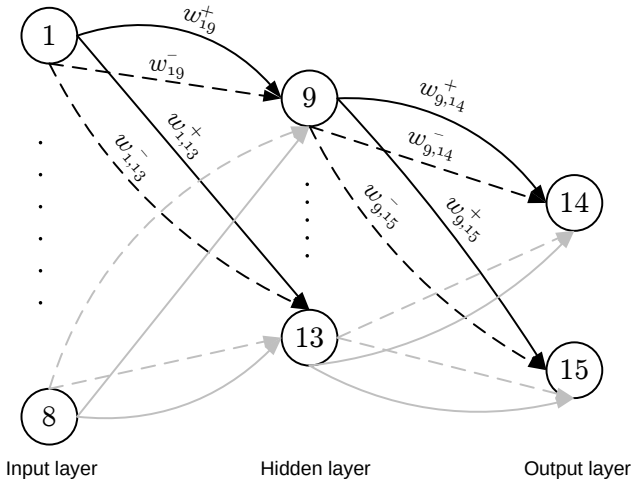


Figure 3.1: An example of the feed-forward RNN structure used for anomaly detection, with 8 input nodes, 5 hidden neurons and 2 output nodes corresponding to attack and normal traffic. The learning algorithm processes the input training patterns in sequence and updates the weights. The k-th training set consists of a feature vector $\mathbf{x}^{(k)} = (\Lambda_1^{(k)}, \cdots, \Lambda_8^{(k)})$ and its classification $\mathbf{y}^{(k)} = (y_{14}^{(k)}, y_{15}^{(k)})$ set to $(1, \epsilon)$ for attack and $(\epsilon, 1)$ for normal samples where $\epsilon \simeq 0$. All other exogenous signals are set to zero.

decision: the input neurons receive the features computed for the current time slot, and the output nodes correspond to the probabilities of the input pattern belonging to any of two traffic classes (i.e. attack or normal). The final decision about the traffic observed in the time slot is determined by the ratio of the two output nodes (i.e. q_{14}/q_{15}): attack if the ratio is greater than 1 and normal otherwise. We have used an implementation of the RNN provided in [7].

3.2.1 Feature Selection

The selection of useful and information bearing input features for any classification problem is one of the most important parts of the solution. In our approach, we used features that can capture the RRC signalling dynamics of a user based on raw IP packet traces collected from the mobile network core, namely the SGSN/GGSN in UMTS and SGW/PGW in LTE. Our aim is to select features that are easy to measure or calculate without high computational or storage cost, given the sheer size of the mobile net-

work, while at the same time reflect both the instantaneous behaviour and the longer term statistical properties of the traffic. We extract for each UE information related to inter-arrival times, lengths and destination IP addresses of packets. We do not assume knowledge of the application generating a packet nor its service type, which would require the use of a commercial deep packet inspection (DPI) tool, and would result in considerable overhead for real-time detection. The features that we have used in our detection mechanism are described below.

Inter-arrival Times

RRC signalling occurs whenever the UE sends or receives packets after an inactivity period that exceeds an RRC timer. Thus, the volume of traffic exchanged by a UE does not map directly into signalling load which is more influenced by the frequency of intermittent activities. To capture this interaction between the data and RRC control plane, we define a *burst* as a collection of packets whose inter-arrival times are less than δ seconds, where δ is smaller than the RRC timers, typically in the order of few seconds. Specifically, for a sequence of packets whose arrival instants are $\{t_1, t_2, \cdots\}$, we group all packets up to the n-th arrival into a single burst, where $n = \inf\{i : t_i - t_{i-1} > \delta\}$, and then proceed in a similar manner starting from the $(n+1)$-th packet arrival. Note that packets within a single burst are likely not to trigger any control plane messages, while inter-arrival times of bursts will be correlated to the actual signalling load generated by the UE. In this manner, we remove any bias regarding the volume of traffic sent or received by the UE, and focus more on the frequency of potentially resource-inefficient communications.

The features based on the times between bursts are then calculated as follows. The algorithm stores the mean and standard deviation of the inter-burst times in each slot then, using the most recent w values, it computes (i) entropy of the averages, (ii) moving average of the standard deviations, and (iii) moving average of an anomaly score for the averages computed based on the RRC timer T in the high bandwidth state. In particular, the anomaly score $\alpha(z[t])$ of the average inter-burst time in slot t is set to zero when $z[t] < T$, reflecting the fact that such shortly spaced bursts may not have generated many RRC transitions; it is high when $z[t]$ is slightly larger than T, indicating potentially resource-inefficient bursts; and it drops quickly when $z[t]$ is few seconds larger than T. We can obtain this effect using a gamma distribution:

$$\alpha(z[t]) = \frac{(z[t] - T - \epsilon)^{n-1} e^{-\frac{(z[t]-T-\epsilon)}{\theta}}}{\theta^n \Gamma(n)},$$

where $\Gamma(n)$ is the gamma function evaluated at n, ϵ is a small positive number and

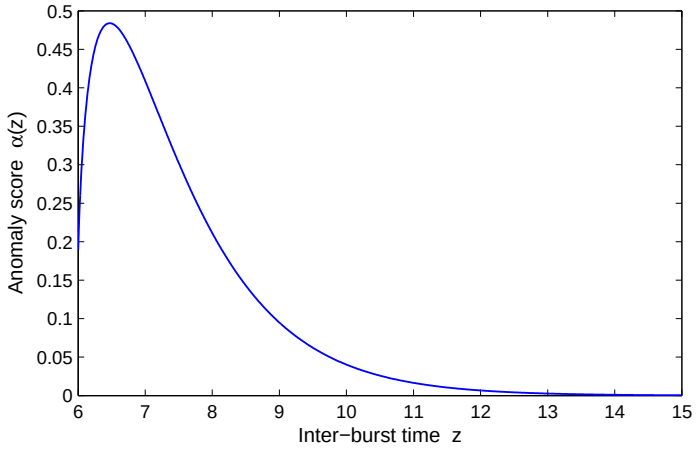


Figure 3.2: Anomaly score based on the gamma function, taking high values when the time between successive bursts is slightly larger than the RRC timer which in this case is $6s$.

n, θ are parameters of the gamma distribution chosen to adjust the decay of α as $z - T$ increases. The shape in Fig. 3.2 which satisfies the above requirements is obtained by setting $n = 1.5$ and $\theta = 1$.

Packet Size

If the data sent by a user has a probabilistic description, then it is expected that the packet size distribution for a normal device will be markedly different from that of a mobile device running a misbehaving application. For example, signalling storms can be caused by failures in over-the-top cloud services [54] or peer-to-peer networks used by VoIP applications [23]. In such cases, the client application will attempt to reconnect more frequently, causing significant increase in the number of TCP SYN packets sent by the user. This in turn changes the randomness or uncertainty of information associated with the size of packets, and can be used to identify misbehaving mobiles in the event of a signalling storm. Our algorithm computes the average size of packets sent by a UE within each slot, and evaluates a feature based on the entropy of the most recent w measurements.

Burst Rate

Another obvious characteristic of signalling storms is the sudden increase and sustained rate of potentially harmful bursts generated by a misbehaving user. Moving average of the burst rate per slot and entropy of the rates across the observation window are used as features in order to capture, respectively, the frequent and repetitive nature of nuisance transmissions. Furthermore, a misbehaving application may change the traffic profile of a user in terms of the ratio of received and sent bursts, as in the case of the outage induced storm described above where many SYN packets will not generate acknowledgments. Hence, we also use as a feature the mean of the response ratios within the window of w slots.

Destination Addresses

The number of destination IP addresses contacted by a normally functioning mobile device is expected to be significantly different from that of an attacker [35], whether the attack originates from the mobile network due to a misbehaving application, or from the Internet as in the case of unwanted traffic (e.g. scanning probes, spam, etc.) reaching the mobile network [55]. In the former, the number of destination IP addresses will be unusually small relative to the frequency of bursts, while in the latter this number is very high. Thus we calculate the percentage of *unique* destination IP addresses contacted within each time slot, and use the average of the most recent w values as a feature.

3.2.2 Selecting the Parameters of the Algorithm

In the following, we summarise the parameters for the RNN algorithm and discuss how they should be selected adaptively and how the choice of each parameter influences the performance of the anomaly detector:

- Slot size Δ: This defines the resolution of the algorithm and the frequency at which classification decisions are made. It should be long enough for the measured statistical information to be significant, but not too long to make the algorithm react slowly to attacks. In our experiments we set $\Delta = 1$ minute.
- Window size W: The window size $W = w\Delta$ determines the amount of historical information to be included in a classification decision. The choice of the window size presents a trade-off between speed of detection and false alarm rate, since a small window makes the algorithm more sensitive to sudden changes in the traffic profile of a user, which in turn increases both detection and false alarm rates. This trade-off can be optimised by adjusting W according to the level of congestion in

the control plane, with shorter windows for higher signalling loads to enable the algorithm to quickly identify misbehaving UEs. Note that information about the "health" of different network servers is typically available to the mobile network operator in the form of key performance indicators (KPIs) that can be fed to the algorithm to adjust the window size. Furthermore, based on these KPIs, the anomaly detector could be switched on only when the signalling load exceeds a certain threshold, thus eliminating the need to continually analyse users' traffic. The value of w used in our experimental results is 5, but we also experimented with other values which confirmed the aforementioned observations.

- Maximum packet inter-arrival time within a burst δ: This should be selected based on the RRC timers, so that potentially resource-inefficient transmissions can be tracked. In our simulations of a UMTS network, the timers in DCH and FACH states are set to, respectively, $T_1 = 6s$ and $T_2 = 12s$ based on [52]. We have evaluated different values of $\delta < \min(T_1, T_2)$ and the results indicate that it does not affect detection performance significantly, but that training time drops as δ is increased. The results presented in this deliverable are obtained with $\delta = 3s$.

3.3 Experiments and Results

In this section, we evaluate the performance of our CDR based anomaly detection algorithm. Towards this end, we have extended our simulator [6,34] to include more realistic data plane models in addition to the signalling protocols of mobile networks. We first present the traffic models that have been integrated into the simulations, including two attack models that represent both malicious and misbehaving UEs. Then we describe the results of applying our real-time algorithm on the dataset produced by the simulator.

While the impact of signalling storms on mobile networks has been analysed extensively in [6], the objective of the present simulation setup is to evaluate the performance of our algorithms in identifying signalling anomalies in users' profiles; thus a small scenario has been considered. In particular, we simulated 200 UEs in an area of 2x2 km^2 which is covered by 7 Node Bs connected to a single radio network controller (RNC). The core network (CN) consists of the SGSN and the GGSN which is connected to 37 Internet hosts acting as application servers, 5 of which for instant messaging, and 2 are contacted by the attacking UEs.

3.3.1 Model of the User

The user model consists of three popular mobile services that are active simultaneously in order to create more realistic user behaviour. The model can also support a diurnal pattern for UE behaviour, where the UE is active for a certain duration (e.g. between 14 and 16 hours) every 24 hours, and is inactive the rest of the time during which the user does not generate or respond to traffic. This pattern represents the day/night cycle of users, and it varies from one user to another based on a random distribution.

Web Browsing

The interactive web browsing behaviour is based on the self-similar traffic model described in D4.1 [6] and assumes Zipf-like distribution for web server popularity, which has been widely used in the literature since it was first suggested in [16].

Instant Messaging

Instant messaging (IM) applications are characterised by frequent, small data transmissions and a long tail distribution representing messages with media rich contents such as videos and photos. The IM application model consists of two distinct but related parts: message generator and responder. Each UE generates messages to chosen destinations, and also responds to received messages with a given probability. The message generator works based on sessions and waves. A session represents the duration that the user is actively generating messages, and consists of one or more waves where the messages are actually sent. At each wave, the user generates and sends one or more messages, the number and length of which are configurable with random distributions, to a single destination (mobile user) chosen at random. The time between waves within a session, the session duration and the time between user sessions are all given by random distributions. On the other hand, the UE responds to each received message with a given probability, and this response behaviour is independent of message generation, and can occur both inside and outside of the user's IM sessions.

The final destination of a message can be another mobile in the same network or a mobile in another network (not explicitly simulated); mobiles in different networks are represented by one or more servers in the simulation, which act on behalf of these users. Mobiles in the same network are explicitly simulated. Regardless of its final destination, each message passes through an Internet chat server, which forwards the message to its final destination, i.e. another mobile user. We simulate multiple chat servers representing popular chat applications and services such as WhatsApp, GTalk, Skype, etc., and currently assume that each message belongs to a chat application that is chosen uniformly

at random from the available applications. The simulation model supports more generic message-to-application assignment based on other random distributions.

Short Message Service

The SMS application is similar to the IM application in that it consists of a message generator and a responder, and also operates based on the same concept of sessions and waves. Different from the IM application, we assume a single intermediate server within the mobile network that handles all SMS messages for that network, i.e. the SMSC server. SMS messages are also different than IM messages in their types. Each SMS message is assigned a type at creation time, which can also be inferred from its destination address (i.e. phone number); an SMS message can be classified as in-network mobile, out-network mobile, premium, and other, e.g. non-premium SMS based services, based on its destination. In-network mobiles are naturally represented by the UEs explicitly simulated; we represent the out-network mobiles, premium numbers and other destinations by servers outside the simulated mobile network, with one or more servers representing each class. Therefore, the type of a sent or received SMS can be inferred from its source and destination addresses (numbers). The type of the SMS message the UE generates is chosen at random based on the parameters of the SMS application.

3.3.2 Attack Model

We consider *DCH attacks* where the attacker aims to overload the control plane by causing superfluous promotions to the high bandwidth DCH state. A similar FACH attack can be launched where the transition of interest is to the FACH state, which is more effective in overloading the core network components such as SGSN (UMTS) and MME (LTE), but it is generally more difficult to launch because it requires knowledge of the RRC buffer thresholds and measurement of user traffic volume.

We consider two types of attackers. The first is *aggressive* in the sense that a malicious device knows when an RRC state transition occurs, and launches the next attack once a demotion from DCH to FACH is detected. To perform the attack, we assume that the attacker has inferred the radio network configuration parameters, and is monitoring the user's activity in order to estimate when a transition occurs so as to trigger a new one immediately afterwards. However, there could be an error between the actual transition time and the estimated one, which we represent by an exponentially distributed random variable with mean $2s$. When the attacker "thinks" that a transition has occurred, it sends a high data rate traffic to one of its Internet servers in order to cause the buffer threshold to be reached and therefore result in a promotion to DCH. This model is used

mainly for *training* the supervised anomaly detection algorithm.

The second attack type is based on a *poorly designed* application that sends periodic messages whenever the user is inactive, with the transmission period set to be slightly larger than the DCH timer in order to increase the chances of triggering state transitions. This behaviour represents the case where an application uses a pull mechanism to fetch updates periodically, and the update period happens to "synchronise" with the RRC timer. However, unlike aggressive attackers, the misbehaving application cannot guarantee the generation of signalling traffic for each of its updates, since (i) the application only starts when local user activity stops but it cannot observe downlink traffic that may have restarted the DCH timer at the signalling server; and (ii) the data volume may not be large enough to trigger a promotion to DCH. In both cases, the periodic transmissions may become completely *out of sync* with the RRC state machine, therefore not generating any signalling traffic.

The two distinct attack models allow us to represent both malicious and benign behaviours that may lead to a storm, but the first is well distinguishable and separable from the behaviour of a normal user in terms of both temporal and traffic volume. On the other hand, the second attack model captures the signalling behaviour of legitimate applications that are much more similar to an "attack" rather than to a "normal" behaviour, but are difficult to detect from CDR dynamics. Thus we use this model to *test* the performance of our algorithm.

3.3.3 Results

The RNN algorithm provides at the end of a time slot the probabilities that the input features belong to an attack and normal behaviour (i.e., q_{14} and q_{15} in Fig. 3.1). The final decision about the traffic is then determined by the ratio of the two output nodes q_{14}/q_{15}: it is classified as attack if the ratio is greater than 1 and normal otherwise. Fig. 3.3 shows the classifier output (top) and the actual RRC state transitions (bottom) of a *misbehaving* UE as captured during a simulation run. It can be observed that when the malfunctioning application is active, the number of state transitions significantly increases, with most transitions occurring between the FACH and DCH states in this attack scenario. It is this back-and-forth transitioning behaviour that causes excessive signalling load in the mobile network, while the load on the data plane is mostly unaffected, rendering traditional flooding based security solutions unable to detect signalling storms. However, our anomaly detection mechanism is able to track very accurately the RRC state transitions of the UE, and to quickly identify when excessive signalling is being generated, despite the fact that it does not directly monitor these transitions but rather infers them from the CDR features that we have described. One can also observe

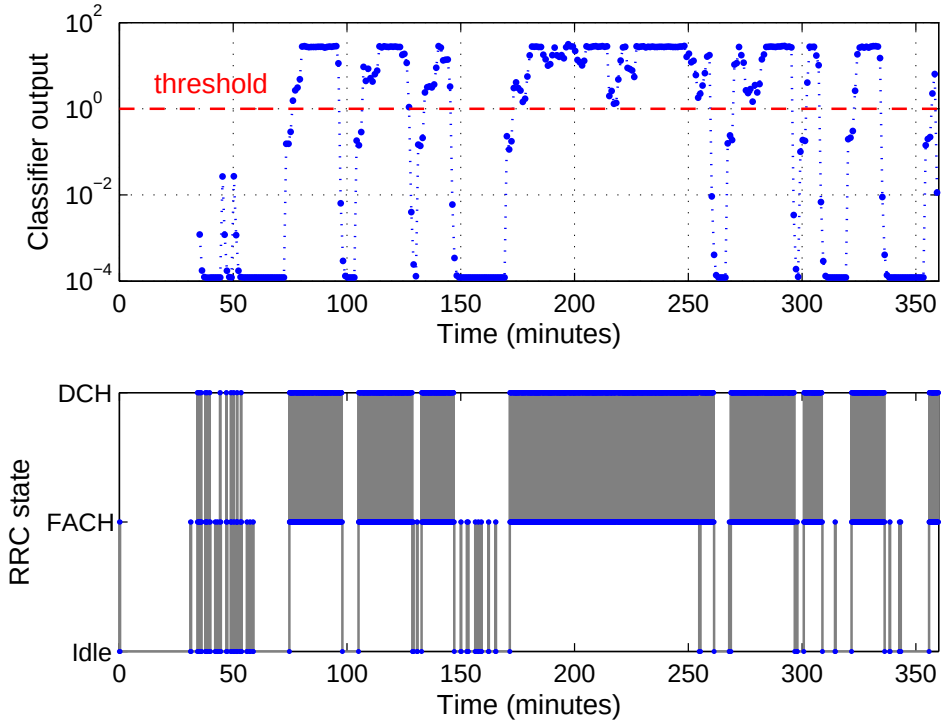


Figure 3.3: Classifier output (top) and RRC state transitions (bottom) for a misbehaving UE.

that the classifier's output sometimes drops close to 1 during an attack epoch, which is attributed to other normal applications generating traffic in those time instants, thus reducing the severity of the attack. As mentioned earlier, the detection speed and tolerance to signalling misbehaviour can be adjusted by modifying the size of the observation window, which in this scenario is set to 5 minutes.

Fig. 3.4 shows results when there is no attack, where the number of state transitions in a given period are small and due to normal traffic generated and received by the UE. In this case, the UE does not spend long periods in "active" states, i.e. FACH and DCH, quickly transitioning down to Idle (bottom figure), and the classifier (top figure) does not generate any alarms regarding the signalling behaviour of the UE as one would expect.

Next we examine in Fig. 3.5 how our algorithm performs when presented with a heavy

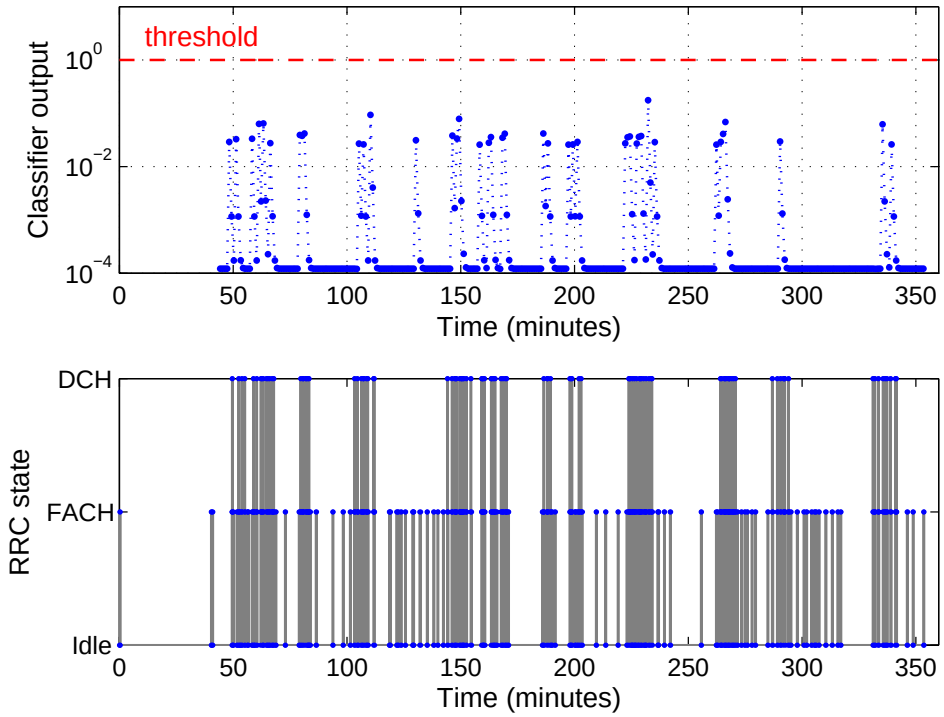


Figure 3.4: Classifier output (top) and RRC state transitions (bottom) for a normal UE.

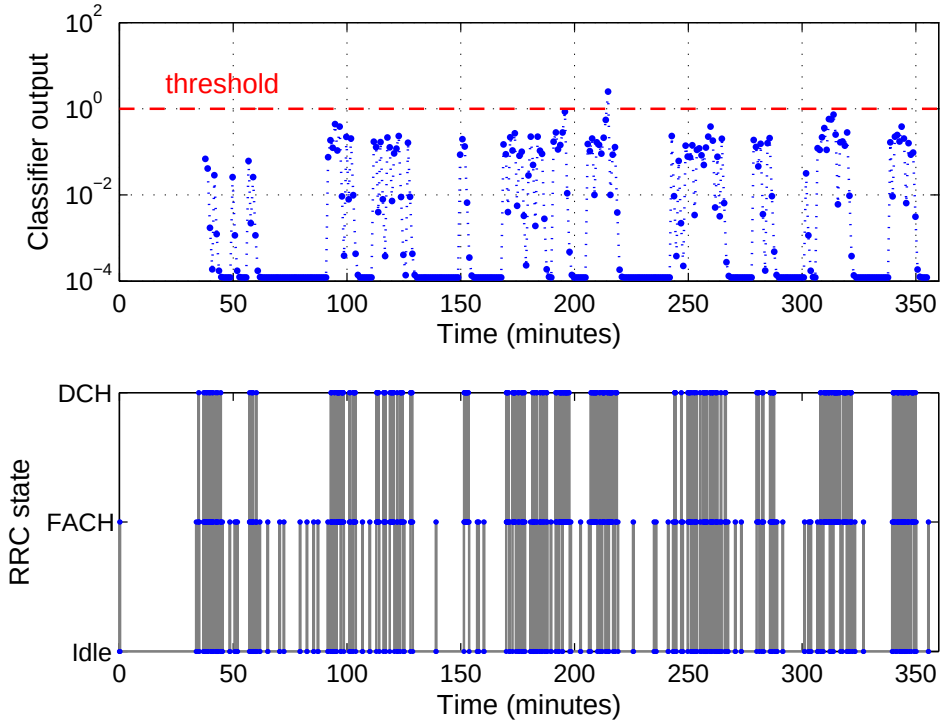


Figure 3.5: Classifier output (top) and RRC state transitions (bottom) for a heavy UE that generates significantly more signalling than the average user in the simulation experiment.

normal user that generates significantly more state transitions than the average normal user in our simulations. Interestingly enough, the classifier outputs a single alarm (out of 360 samples) when the corresponding state transitions are indeed excessive. Since the anomaly detection algorithm is supposed to be activate only when there is a signalling overload condition, such classification decisions may not always be considered as false alarms, as the goal would be to identify users that are causing congestion, regardless of whether they are attacking deliberately or not.

A well-known approach for assessing the performance of a binary classifier is to plot true positive rate (TPR) against false positive rate (FPR) in what is commonly refereed to as receiver operating characteristic (ROC) space. The TPR (also known as sensitivity or recall) is the fraction of attack instances that have been correctly identified by the

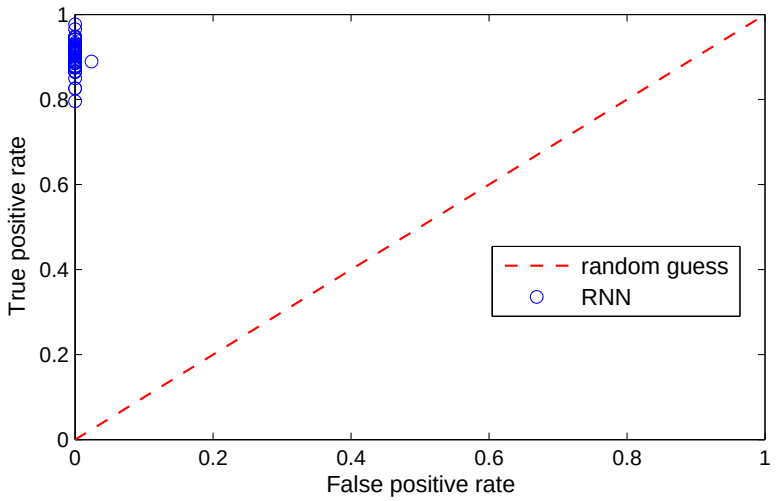


Figure 3.6: Detection results in the ROC space, where the diagonal line corresponds to random guessing. There are 50 points, each representing the classification results for a misbehaving UE over an activity period of 6 hours.

classifier, while FPR or fall-out is the proportion of normal samples that have been mistakenly classified as malicious. We assume that if a UE generates at least 1 attack packet within a time slot, then the corresponding output of the classifier should be larger than 1, otherwise a false positive is declared. We obtain the TPR and FPR per UE from the 360 classification decisions taken during the simulation experiment (6 hours, and the resolution of the detector is $\Delta = 1$ minute). The results for 50 UEs are depicted in Fig. 3.6 showing that the FPR is zero for all but one case, while the TPR is on average 90% which can be further improved, at the cost of higher FPR, by reducing the window size W. Finally, Fig. 3.7 illustrates the accuracy of our classifier, namely the proportion of correct decisions (both true positives and true negatives) out of all test samples. The results indicate an accuracy between 88% and 98% with an average of 93% over the 50 test cases. This fluctuation, which can also be observed in Fig. 3.6, can be attributed to the fact that our algorithm does not classify an attack as such until few time slots have passed (depending on w), and therefore misbehaving UEs with many silent periods will produce higher false positives; fortunately, these less aggressive UEs will generate lower signalling loads.

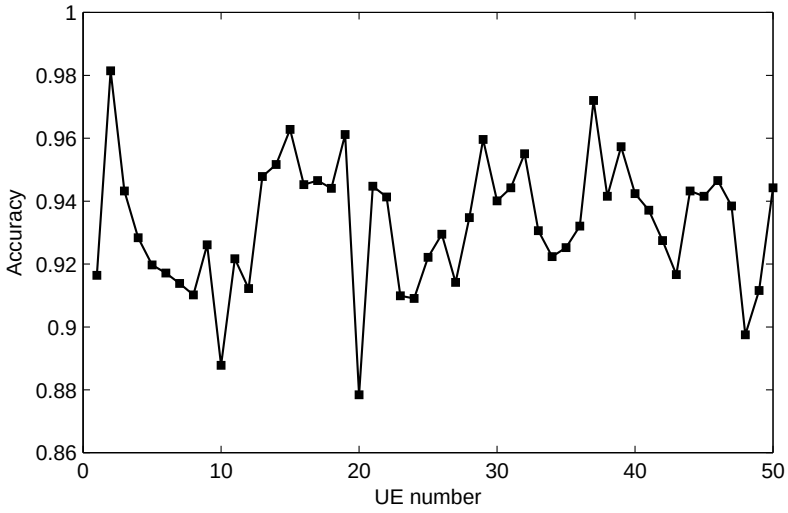


Figure 3.7: The accuracy of the RNN algorithm, measured as the fraction of correct decisions over the activity period of 6 hours, for 50 misbehaving UEs.

3.4 Mathematical Analysis

In this section, we present a mathematical model [31] which allows us to evaluate and optimise the performance of the two storm detectors that we have developed so far: the above CDR approach, and the signalling based detector in D4.1 [6] which counts the number of nuisance RRC transitions between low and high bandwidth states, and enforces a mitigation policy if this number exceeds a threshold. The analysis allows us to derive the optimum value of the counter's threshold, given a detection rate for the overall system, and to show that this optimum value substantially reduces both the average number of attacking devices and the amount of signalling traffic.

The analysis and discussion in this section is conducted using very elementary and well established modelling techniques [30,33] that are widely used in telephony and teletraffic. We represent the set of normal and malicious mobile calls in the system by a state $s(t)$ at time t as:

$$s(t) = (b, B, C, A_1, a_1, \ \ldots\ , A_i, a_i, \ \ldots) \tag{3.1}$$

where:

- b is the number of mobiles which are just starting their communication in low bandwidth mode,

- B is the number of normal mobiles which are in high bandwidth mode,
- C is the number of normal mobiles that have started to transfer or receive data or voice in high bandwidth mode,
- A_i is the number of attacking mobiles which are in high bandwidth mode and have undergone a time-out for $i-1$ times,
- a_i is the number of attacking mobiles which have entered low bandwidth mode from high bandwidth mode after i time-outs.

We assume a Poisson arrival process of rate λ of new "calls" or mobile activations, and a call that is first admitted in state b then requests high bandwidth at rate r. Note that r^{-1} can be viewed as the average time it takes a call to make its first high bandwidth request to the network.

With probability $1-\alpha$ such a call will be of normal type and will then enter state B, while with probability α it will be an attacking call and will request high bandwidth and hence enter state A_1 indicating the first request for bandwidth that is made by a defectively operating application or malware that can contribute to a storm. A schematic diagram of the model is presented in Fig. 3.8.

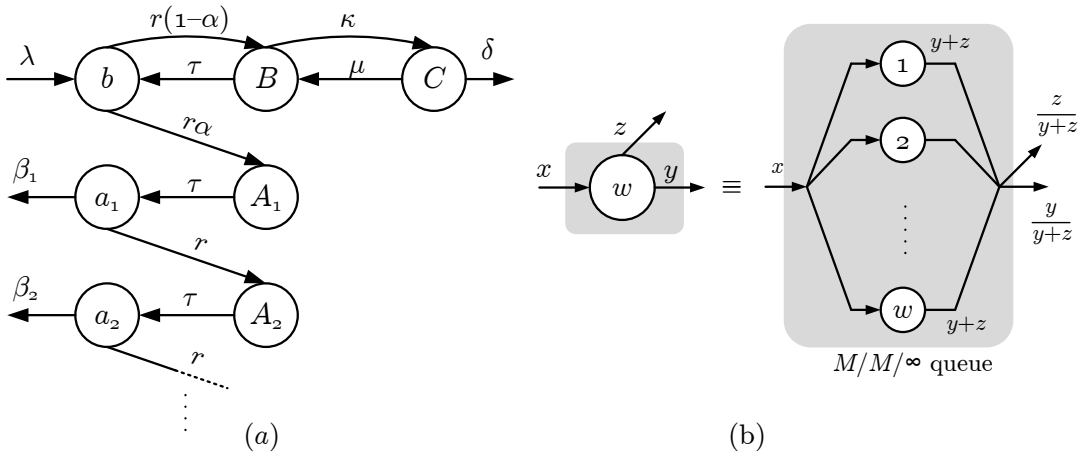


Figure 3.8: A schematic representation of the mathematical model: (a) evolution of the number of normal and attacking calls in the system, and (b) the $M/M/\infty$ queueing system representation for a node in the model with arrival rate x and departure rates y and z.

Once a call enters state A_1, since it is misbehaving, it will not start a communication and will time-out after some time of average value τ^{-1}. Note that the time-out is a parameter that is set by the operator, and in practice it is of the order of a few seconds. After entering state a_1, the call may be detected as being anomalous, and will be removed or blocked from the system at rate β_1, where β_1^{-1} is the average time it takes the detector to identify that this call has the potential to contribute to a storm, and to block the call from further activation. However, it is very unlikely that the system is so smart that it can make this decision correctly regarding the call so early in the game, so typically $\beta_1 \simeq 0$ and the call will manage to request high bandwidth and then enter state A_2 at rate r.

Proceeding in the same manner, in state A_i the anomalous call will again not start a normal communication, so it will eventually time-out after an average time τ and enter state a_i, and so on. As a consequence, the rates at which calls enter these states is simply:

$$\begin{aligned} \Lambda_{A_1} &= \alpha\Lambda_b, \\ \Lambda_{a_i} &= \Lambda_{A_i}, \\ \Lambda_{A_{i+1}} &= \Lambda_{a_i}\frac{r}{r+\beta_i} = \alpha\Lambda_b\prod_{l=1}^{i} f_l, \end{aligned} \tag{3.2}$$

where $f_l = \frac{r}{r+\beta_l}$, and Λ_b is the rate at which calls enter state b, which will be determined below from a more detailed analysis. Different calls will interfere each other via (i) the access to limited wireless bandwidth, and (ii) possible congestion due to signalling and other traffic in the backbone network. However if we neglect these points as a first approximation, calls act independently of each other so that the *average number* of calls in each of the "attacking" states, that are denoted by a_i and A_i, is the *average arrival rate* of calls into the state, multiplied by the *average time* spent by a call in that state, so that we have:

$$\begin{aligned} N_{A_1} &= \frac{\alpha\Lambda_b}{\tau_1}, \\ N_{A_i} &= \frac{\alpha\Lambda_b}{\tau_i}\prod_{l=1}^{i-1} f_l,\ i>1, \\ N_{a_i} &= \frac{\alpha\Lambda_b}{r_i+\beta_i}\prod_{l=1}^{i-1} f_l,\ i>1. \end{aligned} \tag{3.3}$$

As a consequence, the total average number of malicious calls becomes:

$$N_a = \sum_{i=1}^{\infty}[N_{a_i} + N_{A_i}] = \alpha\Lambda_b \sum_{i=1}^{\infty}\{[\prod_{l=1}^{i-1} f_l][\frac{1}{\tau} + \frac{1}{r+\beta_i}]\}. \tag{3.4}$$

Now with regard to normal calls, once a call requests high bandwidth and enters state B, it will start communicating and this will be expressed as a transition rate κ which takes the call into "communication state" C. From C the call's activity may be interrupted, as when a mobile device stops sending or receiving data to/from a web site, or when a voice call has a silent period, in which case the call will return to state B at rate μ. Similarly, the call may end at rate δ, leaving the system.

From B it may either return to C at rate κ signifying that transmission or reception has started once again, or it may time-out at rate τ and return to state b. Once it returns to state b after a time-out, the call can try again to enter state B or state A as a normal or attacking call, since we have to include the fact that a normal call may become an attacking call after acquiring malware during its "normal" communication with a web site or with another mobile. As a consequence, we can calculate the rates at which the calls enter these normal operating states become:

$$\begin{aligned}
\Lambda_b &= \lambda + \frac{\tau}{\tau+\kappa}\Lambda_B, \\
\Lambda_B &= (1-\alpha)\Lambda_b + \frac{\mu}{\mu+\delta}\Lambda_C, \\
\Lambda_C &= \frac{\kappa}{\kappa+\tau}\Lambda_B,
\end{aligned} \tag{3.5}$$

which yields:

$$\begin{aligned}
\Lambda_B &= \gamma\Lambda_b, \quad \text{where } \gamma = \frac{1-\alpha}{1-\frac{\mu\kappa}{(\mu+\delta)(\kappa+\tau)}}, \\
\Lambda_b &= \frac{\lambda}{1-\frac{\tau}{\tau+\kappa}\gamma} = \frac{\lambda}{1-\frac{\tau(1-\alpha)}{\tau+\kappa-\frac{\mu\kappa}{\mu+\delta}}}, \\
\Lambda_B &= \frac{\lambda\gamma}{1-\frac{\tau}{\tau+\kappa}\gamma}, \\
\Lambda_C &= \frac{\kappa\lambda\gamma}{\kappa+\tau(1-\gamma)}.
\end{aligned} \tag{3.6}$$

3.4.1 Optimum Counter for Mitigation

In this section we describe how to optimise the detection of signalling storms using both signalling [6] and billing related information. The anomaly detector based on signalling protocols counts the number of successive RRC transitions that a mobile triggers without actually making use of the requested bandwidth. If this number reaches a threshold n, then a mitigation policy is activated to prevent the mobile from making excessive requests. On the other hand, the billing based detector conducts analysis of the user's behaviour, and can also check other attributes such as destination IP addresses or port numbers that may be associated with malicious activities.

A large value of n will improve the chances of *correctly* detecting a misbehaving mobile user, providing the system with full confidence to activate the mitigation policy. If n is small the counter based detector will have high false positives, giving a bigger role for the billing based detector. Thus the higher the n, the faster the decision can be to invoke mitigation, using only signalling information. Based on this principle, and with reference to our earlier definition of β_i, we have:

$$\beta_i = \begin{cases} 0, & 1 \leq i < n, \\ \beta(n), & i \geq n \end{cases}$$

where the detection rate $\beta(n)$ increases with the threshold n, with a slope or derivative with respect to n expressed as $\beta^{'}(n)$. Using the previous analysis, the average number of malicious calls becomes:

$$N_a = \alpha\Lambda_b[(n - 1 + \frac{r}{\beta})(\frac{1}{\tau} + \frac{1}{r}) + \frac{1}{\tau}] \tag{3.7}$$

while the resulting signalling load from the attack is given by the total rate of malicious transitions between low and high bandwidth states:

$$\Lambda_a = \alpha\Lambda_b + \sum_{i=1}^{\infty}[\Lambda_{a_i} + \Lambda_{A_i}] = \alpha\Lambda_b[2n + 1 + \frac{2r}{\beta}] \tag{3.8}$$

With some further simple analysis we can show that the value n^* that minimises *both* N_a and Λ_a, is the value that satisfies:

$$\beta(n^*)^2 \approx r.\beta^{'}(n^*). \tag{3.9}$$

Figure 3.9 shows N_a and Λ_a versus n when $\beta(n) = 0.02n$ with $r = 0.5\ secs^{-1}$, and we see that $n^* = 5$ as predicted by (3.9). Note that since the mitigation strategy prevents misbehaving mobiles from generating signalling traffic for a short period, it cannot completely eliminates the attack; however, the optimum value n^* is able to reduce the average number of misbehaving devices by about 60%.

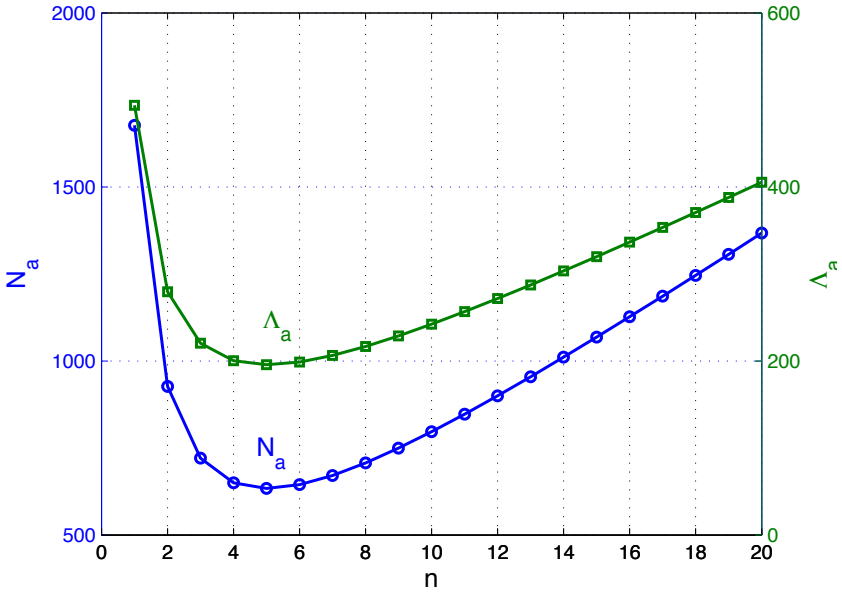


Figure 3.9: Number of attacking mobiles (left) and resulting signalling overload (right) versus the number of false transitions that triggers the mitigation mechanism, when the rate of new connections $\lambda = 10\ calls/s$, timer $\tau^{-1} = 5s$, percentage of malicious calls $\alpha = 0.1$, average connection setup time $r^{-1} = 2s$, and normal user's traffic characterised by $\kappa^{-1} = 10s$, $\delta^{-1} = 5\text{mins}$, and $\mu^{-1} = 5s$.

3.5 Final Remarks

While this section has focused on the real-time detection of signalling anomalies, the proposed RNN based approach is generic and can be applied to detect and classify a variety of attacks targeting both the mobile user and the network. This requires the selection of appropriate features and the adjustment of the algorithm's parameters, as we have illustrated in the case of signalling storms. Furthermore, the insights gained from the mathematical model, which captures the interactions between the signalling and billing based detectors, will be utilised in the integration phase in order to improve the overall performance of the NEMESYS solution.

4 Graph-based Descriptors for the Detection of Billing Related Anomalies

This section presents a graph based approach for the detection of billing-related anomalies in mobile networks. The technique uses graph based descriptors (cf. section 4.1) to capture the network billing activity for a specific time period, where nodes in the graph represent users and servers, and edges represent communication events. Graph traversal techniques are then utilised in section 4.2 to create multiple graphs in each vertex neighbourhood, the size of which is manually defined by the analyst based on the task at hand. Section 4.3 presents expressive features that are extracted from the neighbourhood graphs, and used in order to train a random forest classifier [15] to recognise anomalous graphs as presented in section 4.4. The algorithm is finally applied in section 4.5 to detect anomalous users in two simulated datasets for SMS spam and signalling storm. A schematic representation of the proposed anomaly detection approach is presented in Fig. 4.1.

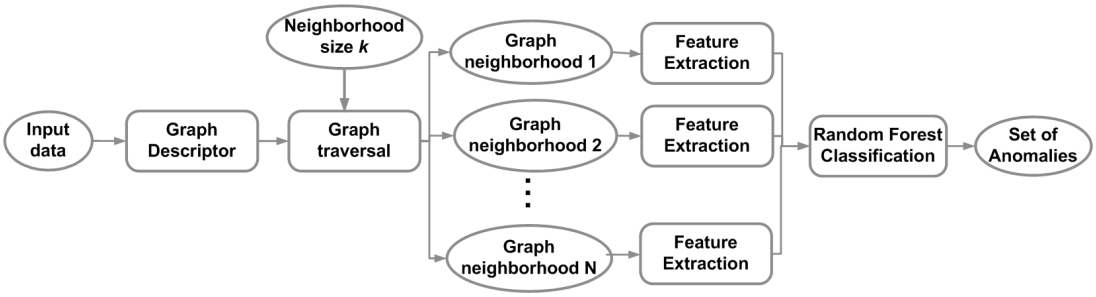


Figure 4.1: An overview of the anomaly detection approach based on graph-descriptors.

4.1 Graph Descriptor

A graph descriptor is a graphical representation of input information that captures its main structural characteristics. A graph $G(V, E)$ comprises a set of vertices $V = \{v_1, v_2, ..., v_{|V|}\}$ and a set of edges $E = \{e_1, e_2, ..., e_{|E|}\}$, where $E \subset V \times V$. A graph

descriptor is a weighted directed graph in which $W : E \to \mathbb{R}^n$ is a function that takes as input a specific edge and returns its corresponding edge weight, which is an n-dimensional vector. The vertices of the graph descriptor can thus represent network entities (e.g. users, servers, etc.), the edges correspond to communication events between them, and the edge weights capture the attributes of these communications.

In the context of CDR, the graph descriptor represents the billing activity of the users in the network. Specifically, the nodes of the graph are the source and destination in the CDR, e.g. user IDs for calls/SMS and IP servers for Internet traffic. The edges connect specific sources to their destinations and are directed, while the weights of the edges represent specific attributes of the CDR records, e.g. the number of calls/SMS, or the size of the Internet packets. The exact weights of the edges are selected based on the task at hand, and the scenario under investigation. In this respect, the proposed graph descriptor provides a holistic view of the CDR activity in the mobile network, and provides a first step towards identifying anomalous behaviours.

Table 4.1 shows a small example of call records for three users: User-1 called User-2 two times, and User-2 called User-3 one time. The graph descriptor for this example is created by associating each user with a vertex, and connecting users that exchanged at least one call. Since there is only one attribute in the CDR, edge weights represent the number of calls between users. The direction of the edges encodes the direction of the communication. The resulting graph descriptor of the CDR in Table 4.1 is illustrated in Fig. 4.2, where the widths of the edges reflect their weights: the edge connecting User-1 to User-2 has weight 2, and all other edges have weight 1.

Table 4.1: An example CDR data representing calls between users.

Call ID	**Origin user**	**Destination users**
1	User-1	User-2
2	User-1	User-2
2	User-2	User-1
3	User-2	User-3
4	User-3	User-4
5	User-4	User-2

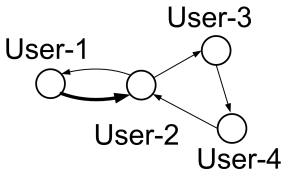


Figure 4.2: The graph descriptor of the calls in the CDR data of Table 4.1. The weights of the edges represent the number of calls between users: User-1 called User-2 two times.

4.2 Graph Neighbourhoods

The graph descriptor provides a method to represent the communication activities of all the users in a network. In order to identify anomalous behaviours in the structure of the graph, a traversal method is applied to the initial graph, which results in the creation of multiple smaller graphs, each representing the neighbourhoods of a vertex. The resulting graphs are subsequently used for feature extraction and classification of anomalies.

Let $N_k(v_i)$ denote the set of k-neighbours of vertex $v_i \in V$. This set is comprised of all the nodes that have graph geodesic distance smaller than or equal to k, where the geodesic distance $GD(v_i, v_j)$ between two vertices v_i and v_j is the length of the shortest path connecting them. Hence, the k-neighbours of vertex $v_i \in V$ are defined as $N_k(v_i) = \{v_j | \forall GD(v_i, v_j) \leq k\}$. Graph traversal then consists in creating a new graph for each vertex v_i, denoted as $G_i(V_i, E_i)$, where $V_i = N_k(v_i) \bigcup v_i$, $E_i = \{e_j = \{v_k, v_h\} | \forall e_j \in E,$ and $v_k, v_h \in V_i\}$, and E is the set of edges of the initial graph descriptor G that is being traversed. Fig. 4.3(a) shows an example of k-neighbourhood graph for different values of k, where the graph is created for the central large vertex; the larger the value of k the larger the resulting graph.

Since the graph is directed, we denote by e_j^s the source vertex of edge $e_j \in E_i$, and by e_j^d its destination vertex. For each k-neighbours of vertex $v_i \in V$, the set of outward directed edges represent all the edges that have their source in the set $N_k(v_i)$ and their destination outside of this set. More precisely, the set of outgoing edges for $N_k(v_i)$ is given by:

$$E_i^{out} = \{e_j | e_j \in E_i, \forall e_j^s \in N_k(v_i) \text{ and } e_j^d \notin N_k(v_i)\} \tag{4.1}$$

The set of ingoing edges E_i^{in} for the set $N_k(v_i)$ is defined in a similar manner. Fig. 4.3(b) shows the set of inward and outward directed edges for each k-neighbourhood of the central vertex in Fig. 4.3(a). It can be observed that all the edges that intersect the

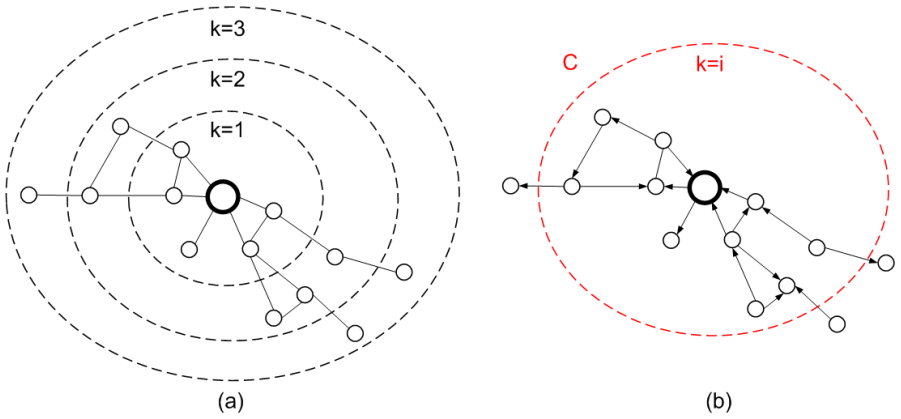


Figure 4.3: (a) Example of k-neighbourhood graph for the central large vertex under different values of k. (b) Example of inward and outward directed edges for the k-neighbourhood graph, with $k = i$, which consists of all the edges that intersect the dashed red ellipsoid C.

dashed red ellipsoid C belong to either the set of ingoing edges (edges that are directed towards the centre) or the set of outgoing edges (edges that are directed away from the centre).

Fig. 4.4 illustrates the graph traversal process, showing the graphs that are created in each step of the iteration using k-neighbourhoods for $k = 1$ and $k = 2$. All the vertices of the graph are traversed, and for the specific vertex under consideration in the current iteration (called active vertex), all its k-neighbourhoods are utilised to create the corresponding neighbourhood graph. The red node in Fig. 4.4 represents the active vertex in each step and the green vertices are its k-neighbourhoods in the graph. The end result is the creation of a set of neighbourhood graphs, one for each vertex: $\{G_1, G_2, .., G_{|V|}\}$, where $|V|$ is the number of vertices of the entire graph descriptor G.

4.3 Graph based Feature Extraction

This section describes how the aforementioned graph neighbourhoods can be used to extract relevant features that are able to characterise the network activity. The neighbourhoods that are created correspond to a set of weighted graphs $G_i\,(V_i, E_i)$, one for each vertex, with a weight mapping function W. From these representations, multiple features can be extracted as described below, taking into account the weights and direction of the edges, where each feature captures a different aspect of the graphs and all

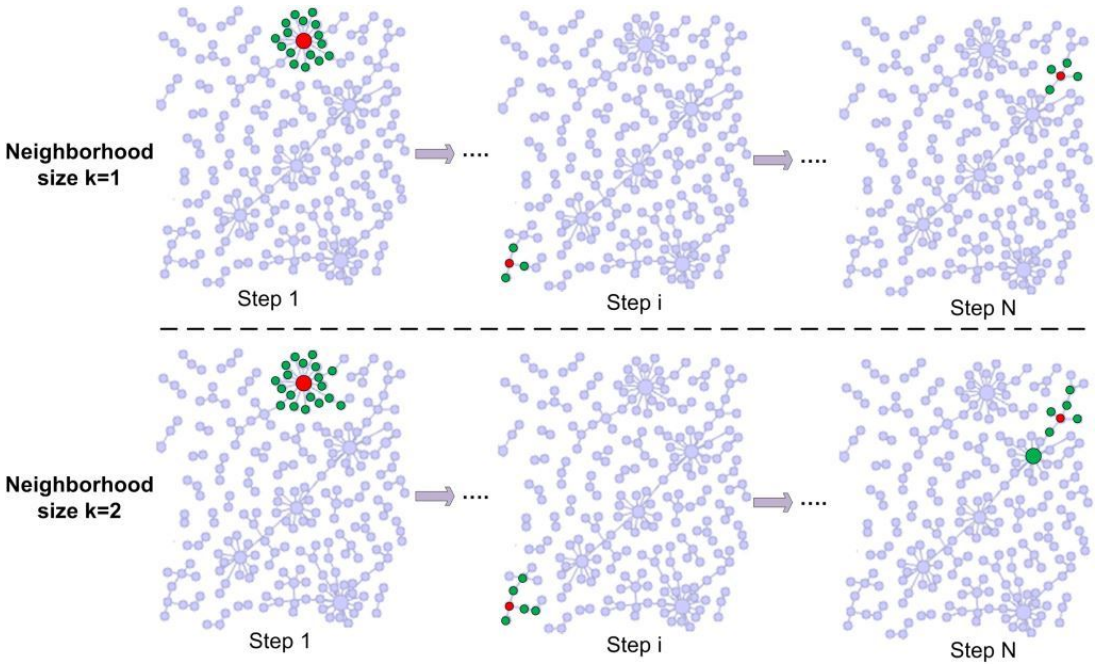


Figure 4.4: An example of graph traversal with $k = 1$ and $k = 2$. The red vertex represents the active vertex in each step, and green vertices are its k-neighbours in the graph.

the features together provide a complete view of the network activity.

4.3.1 Volume

The volume feature f_{vol} captures the size of the graph, and consequently the degree of network activity, which is important for identifying anomalies related to large network activities. The volume feature is defined as follows:

$$f_{vol}^{G_i} = \sum_{e_j \in E_i} g\left(W(e_j)\right), \tag{4.2}$$

where:

$$g(x) = \begin{cases} 1, & \text{for } |x| \neq 0 \\ 0, & \text{for } |x| = 0 \end{cases}$$

and G_i denotes the neighbourhood graph of vertex V_i, E_i its set of edges, and $W(e_j)$ the weight of edge $e_j \in E_j$.

4.3.2 Edge Entropy

Shannon's entropy takes as input a set of symbols and their underling distribution, and returns their average amount of information; the input symbols represent distinct entities in a set of objects, which in this case are the edge weights. Hence the edge entropy f_{ee} captures the amount of information in the edge weights as characterised by their underling distribution. Large difference in the entropy value between distinct graphs indicates that their weight distribution is quite different. The edge entropy of a graph G_i is defined as:

$$f_{ee}^{G_i} = -\sum_{j=1}^{Y^i} \frac{y_j^i}{y_{total}^i} \log\left(\frac{y_j^i}{y_{total}^i}\right) \tag{4.3}$$

where Y^i is the number of different edge weight instances of graph G_i, y_j^i is the number of occurrences of the j-th weight, and $y_{total}^i = \sum_{j=1}^{Y^i} y_j^i$ is the total number of weight occurrences. Note that this feature characterises two possible types of network behaviour: change in the volume of the data, and change in their underling distribution. Large network disturbances not only increase the volume of the data, but also have a direct effect on their distribution. The entropy feature is introduced to capture this effect.

4.3.3 Graph Entropy

Graph entropy f_{ge} measures the structural information content of the graph based on the distribution of the edge connections. The graph entropy was introduced in [40] as a problem of determining the best possible encoding of the information emitted by a source in which pairs of symbols may be indistinguishable. As noted earlier, the symbols represent distinct entities in a set of objects which, in the case of graph entropy, are the vertices. Two symbols are distinguishable if they are connected through an edge, and indistinguishable otherwise. Examples of the entropy value of known graphs (illustrated in Fig. 4.5) include:

- A complete graph with n vertices has graph entropy $log_2(n)$ bits, since all the vertices are distinguishable. In this case, f_{ge} is equivalent to Shannon's entropy.
- A complete bi-partite graph has entropy 1 bit, since it can be partitioned into two sets of indistinguishable vertices.
- A graph with no edges has entropy 0 bits, since all the vertices are indistinguishable.

More formally, Korner's entropy is defined as [61, 62]:

$$f_{ge}^{G_i} = \min_{X,Y} I(X \wedge Y) \tag{4.4}$$

where $I(X \wedge Y)$ is the mutual information of the variables X and Y that have the following properties. The variable X is uniformly distributed and taking its values on the vertices of G_i, while Y on the stable sets of G_i, and their joint distribution is such that $X \in Y$ with probability 1. A subset Y of the vertices V_i of an undirected graph $G_i = (V_i, E_i)$ is a stable set if no edge in the graph has both endpoints in Y.

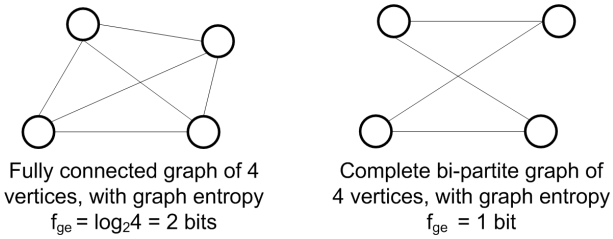


Figure 4.5: Examples of graph entropy.

4.3.4 Maximum Outlier Factor

The maximum outlier factor f_{mof} of all the weights of all the graphs $G_i, \forall i \in \{1, 2, .., |V|\}$ characterises the existence of outliers in the graph, where an outlier represents an edge weight which is not common, and thus deviates from the normal behaviour.

It is important to note that in order to define the f_{mof} feature, all the different edge weights for all the graphs G_i must be taken into account, and not only the weights of the graph under investigation. In this manner, false positives are reduced and only the true outliers stand out, since false positives can be caused by a small sample of weights, which may not be sufficient for an accurate outlier calculation. In our approach, we use the entropy outlier factor (EOF), which was first proposed in [37] to quantify the degree of outlier of each data record in a multidimensional set. The EOF procedure, denoted by $F\big(W(e_k), P\big)$, takes as input an edge e_k plus its weight $W(e_k)$ and the set of all the edge weights $P = \big\{W(e_j) \ \mid \forall e_j \in E_i, \ \forall i \in \{1, 2, .., |V|\}\big\}$, and returns its corresponding outlier factor, which is in $[0, 1]$ with values close to 1 being outliers. The f_{mof} feature is then defined as follows:

$$f_{mof}^{G_i} = \max_{e_k \in E_i} F\big(W(e_k), P\big) \tag{4.5}$$

4.3.5 Edge Weight Ratio

The edge weight ratio feature f_{wr} captures the total difference in the values of the weights between the edges of a graph G_i directed outward from v_i and inward to v_i. Thus,

this feature captures anomalies regarding large imbalances between the communication directions. For example, the number of SMS sent by spammers is usually much larger than what they receive, while a normal user is likely to exhibit comparable number of messages in both directions. We define the edge weight ratio feature of graph G_i as:

$$f_{wr}^{G_i} = \frac{\sum\limits_{e_j \in E_i^{out}} W(e_j)}{\sum\limits_{e_j \in E_i^{in}} W(e_j)} \tag{4.6}$$

where E_i^{out} and E_i^{in} are the set of outward and inward directed edges for the k-neighbours of graph G_i, as described in section 4.2.

4.3.6 Average Outward/Inward Edge Weight

The average outward edge weight f_{avout} represents the ratio of the volume of traffic generated by a node to the number of its destinations:

$$f_{avout}^{G_i} = \frac{\sum_{e_j \in E_i^{out}} W(e_j)}{|E_i^{out}|} \tag{4.7}$$

This feature can be useful to detect, for example, spammers or port scanning worms that typically have high communication activity directed to a large number of destinations, resulting in an average outward edge weight close to 1. On the other hand, normal users are likely to communicate with a small set of destinations, leading to $f_{avout} >> 1$ for high activity users. Similarly, the average inward edge weight f_{avin} can be defined as in (4.8) but using E_i^{in}; this feature will exhibit very small values (close to 0) for the malicious behaviours mentioned above, and moderate to high values for normal behaviour which may further increase when a user receives anomalous traffic.

4.3.7 Number of Outward/Inward Edges with a Specific Weight

These features count the number of outward and inward edges that have weight equal to a specific value w. The feature for outward edges f_{neout} is defined as follows:

$$f_{neout}^{G_i} = \sum_{e_i \in E_i^{out}} 1[W(e_i) = w], \tag{4.8}$$

where $1[x]$ is the characteristic function which takes the value 1 if x is true and 0 otherwise. This feature can recognise deterministic communication patterns which are often

exhibited by malware (e.g. a spammer sending one spam message to each destination, or a bot receiving commands from a remote server a specific number of times). Similarly, we define a feature f_{nein} as the number of inward edges with a specific weight.

4.4 Detection of Anomalous Users

As discussed earlier, each neighbourhood graph is created based on the k-neighbourhood of each vertex in the entire graph descriptor. Each vertex represents an entity in the network (e.g. a user, server, etc.) that is characterised be a set of actions, and the neighbourhood graph describes the network behaviour of this entity. Thus we can use supervised learning methods to train a classifier based on the features extracted from each graph, so as to recognise anomalous neighbourhood graphs, i.e. anomalous entities.

There are a number of methods that may be used for supervised classification. In the context of graph based descriptor framework, random forest [15] has been shown to outperform other supervised learning algorithms [20, 21] in a variety of datasets. In addition, this method does not suffer from overfitting problems, is less sensitive to outlier data, and calculates automatically the importance of each feature in the classification task [9]. Hence we use the random forest algorithm along with the features that we have described in order to perform anomaly detection.

Specifically, the features extracted from each of the neighbourhood graphs are utilised to train the classifier to distinguish between normal and abnormal behaviour. Then, when a new graph neighbourhood is generated, it is provided to the trained algorithm for classification. It should be noted that the random forest is trained using the *Leave-one-out* validation method. In other words, in the case that the random forest must predict the outcome of a specific graph neighbourhood, it is trained with all other graph neighbourhoods that contain both anomalous and normal graphs. In this way, the efficiency of the proposed approach in detecting possible unknown incidents is demonstrated.

The random forest is a collection of multiple decision trees, trained on random samples extracted from the initial dataset. The final classification result is calculated using majority voting. Fig. 4.6 shows an example of decision tree generation for the classification of two coloured circles: red and green. The background colour represents the classification result at the specific iteration, as calculated from the largest class belonging to the corresponding partition. The total space is iteratively partitioned into smaller regions along each feature, in order to create the space partitioning that best represents the data.

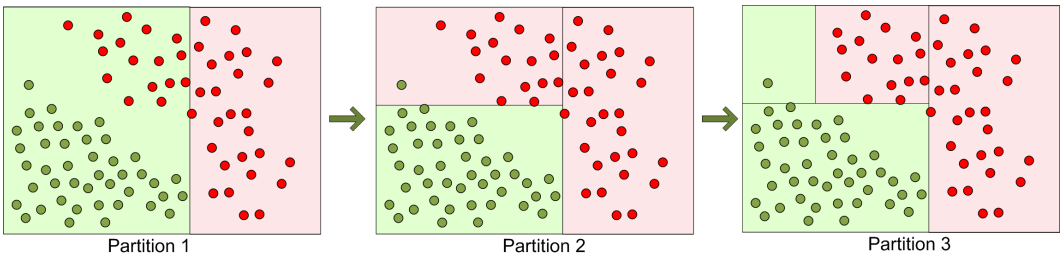


Figure 4.6: Examples of decision tree generation for the classification of two coloured circles (red and green). The total space is iteratively partitioned into smaller regions along each feature, so as to create a partitioning that best represents the data.

4.5 Applications

In this section we evaluate the performance of our anomaly detection algorithm on two distinct datasets, one for a common attack against the mobile users (SMS spam) and the second represents threats targeting the network infrastructure (signalling storms).

4.5.1 Application 1: Spam SMS dataset

The traffic models for the spam scenario are all based on the SMS application described in section 3.3.1. There are in total 10,000 mobile devices which are divided into three distinct groups according to behaviour: (i) 4,000 users exhibit low levels of SMS activity, (ii) 3,000 users exhibit medium levels of SMS activity, and (iii) 4,000 users exhibit high levels of SMS activity. In the simulations, 102 mobile devices, uniformly distributed across the three distinct groups, are assumed to be infected with spam malware. The malware does not have a diurnal cycle, and it regularly sends a configurable number of spam SMS messages, with each message being sent to a destination chosen randomly from the mobile's contact list. Normal mobile users have a very small probability of responding to a spam SMS; this small probability represents the small fraction of users who reply to received spam messages in order to "opt-out" of the service. The goal in this experiment is to identify the infected mobile devices.

The simulation scenario includes out-network, premium and legitimate advertising servers, each having a different messaging behaviour in order to create noise in the data and to produce realistic users' profiles. For example, premium SMS servers may respond to a received message but they do not initiate a conversation, unlike out-network servers which represent mobile devices that are not explicitly simulated. On the other hand,

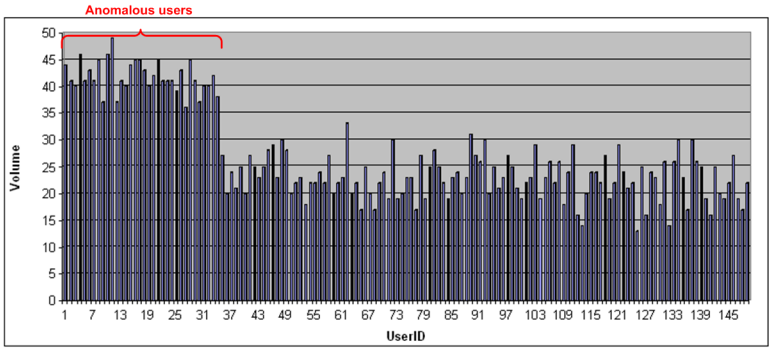
the advertising server is assumed to be a device connected to the mobile network and capable of sending SMS messages at a high rate. We assume that the server has obtained a list of the phone numbers of many mobile users, and frequently selects one or more users at random and sends a message to each. This server follows a diurnal cycle, and it does not respond to any received SMS messages.

Since the important factor in this dataset is SMS activity, the graph descriptor is created to represent this behaviour. Specifically, the vertices of the graph represent the mobile devices, and the edges represent SMS activity between two devices. The weight of an edge represents the number of messages exchanged between the corresponding devices. Thus, each neighbourhood graph represents the SMS activity of each user with respect to the selected destinations and the number of messages directed to them. For the creation of the neighbourhood graphs, k-neighbours with $k = 1$ were used, since a spam message concerns two users, and the addition of extra neighbours would only add noise, since a normal user whose graph geodesic distance to an anomalous user is 2 is not necessarily anomalous.

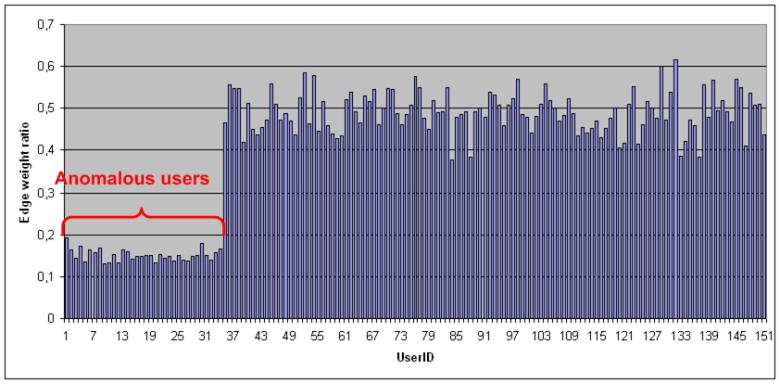
Fig. 4.7a shows the volume feature f_{vol} for 150 users from the low SMS activity class. As one would expect, spammers are characterised by relatively higher volumes, as they communicate with a larger number of destinations. Fig. 4.7b shows the edge weight ratio feature f_{wr} for 150 users from the medium SMS activity class, and the results illustrate that the anomalous users have low values in this feature, which is consistent with the fact that spammers send significantly more messages than they receive.

The edge entropy feature f_{ee} for 150 users taken from the high SMS activity class is presented in Fig. 4.8a. One can observe that anomalous users have large entropy values, indicating that they communicate with a large number of destinations, each at a different frequency. Fig. 4.8b presents the graph entropy feature f_{ge} for the same set of users, and the the results indicate that spammers exhibit lower values for this feature, which is due to the fact that they communicate with a large number of users. For the definition of the number of outward and inward edges with a specific weight (i.e. f_{neout} and f_{nein}), a weight value $w = 1$ was selected based on the assumption that spammers are most likely to send SMS only once to new destinations.

The results of applying our anomaly detection algorithm are summarised in Table 4.2. The algorithm is evaluated based on two metrics, the True Positive (TP) which measures the number of users that are anomalous and also classified as anomalous by the system, and the False Positive (FP) which measures the number of users that are normal but classified as anomalous by the system. As shown in the table, the users were partitioned into three subsets on which both training and testing are performed: (i) low activity users, (ii) low and medium activity users, and (iii) low, medium, and high activity users (i.e. the entire dataset). In all cases the proposed anomaly detection approach was



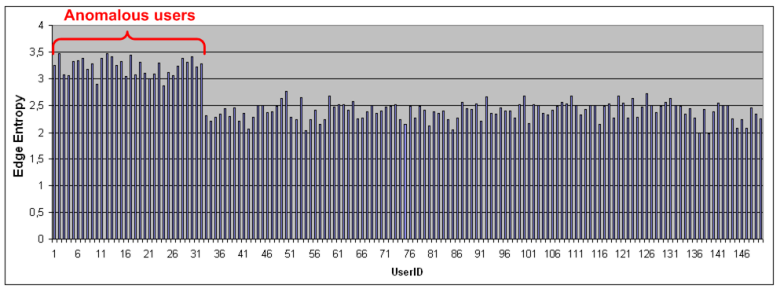
(a) The volume feature f_{vol} .



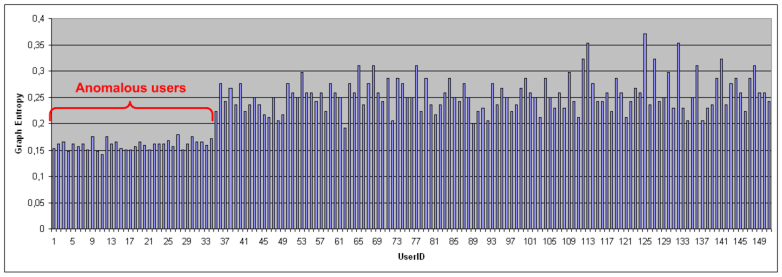
(b) The edge weight ratio feature f_{wr}.

Figure 4.7: Features based on the neighbourhood graphs for 150 users taken from the low (a) and medium (b) SMS activity classes of the spam dataset, using the number of SMS as the weights of the edges. The first 33 users are anomalous, characterised by high volume and low edge weight ratio.

able to identify the anomalous users, producing zero false positives, which indicates that the features we defined can discriminate between normal and malicious behaviours independently of the activity level of the users. In practice, however, we do not expect such level of performance, and therefore we plan to conduct further experiments, as part of the evaluation phase of NEMESYS, using models based on real malware or spam campaigns.



(a) The edge entropy feature f_{ee}.



(b) The graph entropy feature f_{ge}.

Figure 4.8: Features based on the neighbourhood graphs for 150 users taken from the high SMS activity class of the spam dataset, using the number of SMS as the weights of the edges. The first 33 users are anomalous, characterised by high edge entropy and low graph entropy due to the large number of recipients.

Table 4.2: The results of the application of the graph descriptors anomaly detection methods for the identification of spammers in the spam SMS dataset. TP=True Positive, and FP=False Positive

Activity Level	**Normal users**	**Abnormal users**	**TP**	**FP**
Low	3965	34	34	0
Low & Medium	6931	68	68	0
Low, Medium & High	9898	102	102	0

4.5.2 Application 2: RRC Attacks

This section presents the application of the graph descriptor approach to the dataset for RRC signalling storms described in section 3.3.2. There are two classes of anomalous users in this dataset: (i) malicious UEs that send Internet packets whenever a demotion from the DCH state is assumed to have occurred, and (ii) misbehaving UEs that send periodic packets whenever the user is inactive. There are in total 200 users, of which 100 are anomalous, 50 in the first class and 50 in the second one. The goal of this experiment is to evaluate the performance of our algorithm in identifying the anomalous users.

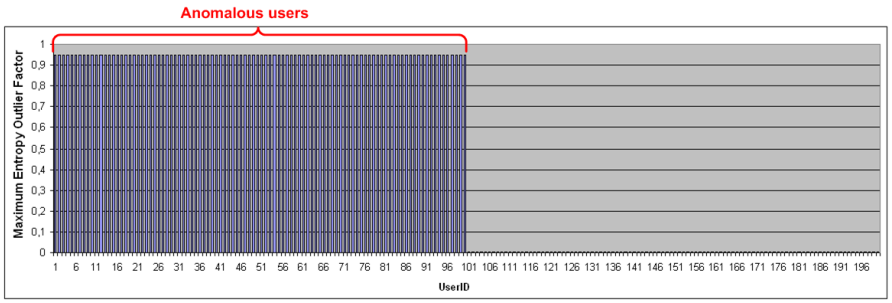
The graph descriptor for this dataset is built in order to reflect its key behavioural factor which is Internet activity. Specifically, the vertices of the graph descriptor represent users and servers, while edges correspond to Internet traffic between them. The weights of edges represent the size of the packets exchanged between the corresponding users and servers. The graph traversal technique takes into consideration only the user vertices for the creation of the graph neighbourhoods (which include both users and servers), and the subsequent identification of anomalous users. For the creation of the neighbourhood graphs, k-neighbours with $k = 1$ were applied so as to take into account the Internet traffic from the user under investigation to its direct neighbours.

Fig. 4.9a presents the maximum entropy outlier factor feature f_{mof} of the neighbourhood graphs for all 200 users in the storm dataset. The feature is used as the main discriminant factor between normal and abnormal behaviour, and it indicates the existence of at least one edge with very high weight in the activity of the abnormal users. Each such edge represents the continuous exchange of Internet packets between the anomalous users and a specific server. Fig. 4.9b indicates that there is no distinct behaviour between normal and abnormal users in the volume feature, confirming our earlier observation that signalling anomalies are difficult to detect using traditional flooding based techniques.

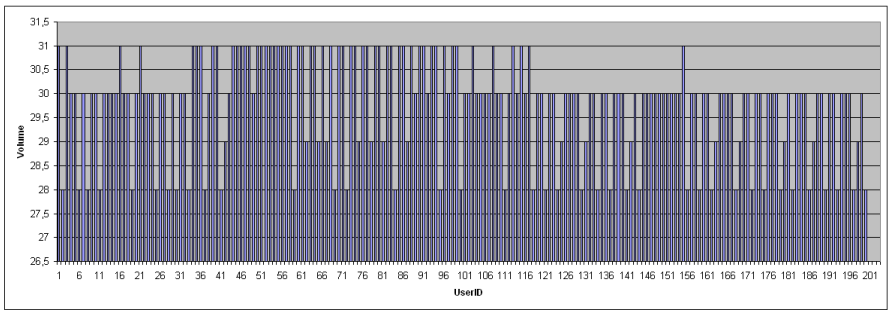
Finally, we summarise the results of applying our graph descriptor anomaly detection method to the storm dataset in Table 4.3, showing that the algorithm was able to identify all the anomalous users, while generating no false positives.

Table 4.3: The results of the application of the graph descriptors anomaly detection method for the identification of abnormal users in the storm dataset.

Normal users	**Abnormal users**	**TP**	**FP**
100	100	100	0



(a) The maximum entropy outlier factor feature f_{mof}.



(b) The volume feature f_{vol}.

Figure 4.9: Features of the neighbourhood graphs created for all 200 UEs in the storm dataset, using packet sizes as the weights of the edges. The first 100 users are anomalous, as indicated by their large outlier degree. However, no distinct behaviour is present in the volume feature.

5 Conclusions

The goal of the NEMESYS project is to develop a novel security framework for gathering and analysing information about the nature of cyber-attacks targeting mobile devices and networks, and to identify abnormal events and malicious network activity. Thus this deliverable described our proposed approaches to the analysis of network traffic and the development of anomaly detection algorithms, which combine modelling and learning from network measurements and billing (meta)data that are readily available to the mobile operator. In contrast to signalling based solutions, the algorithms presented in this deliverable do not require changes to network components and/or protocols, and can be deployed using standard traffic monitoring platforms.

We first presented an online anomaly detection approach based on the random neural network (RNN) [28,29]. Our method uses the notion of an *observation window* in which summary statistics about the behaviour of a mobile user are collected and stored at fixed time intervals (called slots) and used in order to calculate expressive features that can capture both sudden and long term changes in the user's behaviour. The features for the most recent time slot are subsequently fused using a trained RNN to produce the final classification decision. Using our mobile network simulator, we have shown that our technique is able to detect quickly users that are causing signalling overloads in the network, without directly monitoring the control plane itself, and can even identify the end of attacks.

The proposed RNN approach is flexible, providing a number of parameters to optimise the trade-off between detection speed, accuracy and overhead. For example, the size of the observation window and the frequency of statistical measurements (i.e. number of slots within the window) could be adjusted in real-time to respond to network conditions, and to reflect the capacity of the network to tolerate a specific misbehaviour. Our approach is also generic and can be applied to identify a variety of attacks targeting both the mobile user and the network, which requires only the selection of appropriate features and the adjustment of the algorithm's parameters. We concluded our evaluation with a mathematical model that allows us to analyse and optimise the performance of the signalling based detector of D4.1 [6], when used in conjunction with the RNN method; the insights gained from the model will be utilised in the integration phase to improve the overall detection performance of the NEMESYS solution.

Finally, we developed an anomaly detection algorithm which uses graph based de-

scriptors to capture billing related activities in the network, where nodes in the graph represent users and servers, and edges correspond to communication events. In this method, graph traversal techniques are applied to create multiple graphs in each vertex neighbourhood, from which features are extracted and used in order to train a random forest classifier [15] to recognise anomalous graphs. The graph based approach has been validated for the signalling storm dataset, and also for SMS spam data generated by our simulator. The results indicate that the method is able to identify, with high accuracy and precision, anomalous users in both datasets, thus providing a complementary approach to the online RNN algorithm; the latter is activated and configured based on key performance indicators to respond to an urgent condition, while the former can be executed periodically to detect stealthy but non-critical malicious campaigns in the mobile network. These algorithms will be further evaluated in the final phase of the project using our mobile simulator and testbed.

Bibliography

[1] 3GPP TR 23.887 : Machine-type and other mobile data applications communications enhancements (release 12). http://www.3gpp.org/DynaReport/23887.htm, Dec 2013. Technical report.

[2] Cisco visual networking index: Global mobile data traffic forecast update, 2012–2017, Feb. 2013.

[3] NEMESYS Deliverable D2.2: Virtualized mobile honeypot, Oct 2014.

[4] NEMESYS Deliverable D2.3: Lightweight malware detection module for mobile devices, Oct 2014.

[5] NEMESYS Deliverable D3.2: Honey client prototype, Jul 2014.

[6] NEMESYS Deliverable D4.1: Anomaly detection based on signalling protocols, Oct 2014.

[7] H. Abdelbaki. Matlab simulator for the RNN. http://www/cs/ucf.edu/ahossam/rnnsim.

[8] O. H. Abdelrahman and E. Gelenbe. Signalling storms in 3G mobile networks. In *Proc. IEEE International Conference on Communications (ICC'14)*, pages 1017–1022, Sydney, Australia, June 2014.

[9] J. Ali, R. Khan, N. Ahmad, and I. Maqsood. Random forests and decision trees. *IJCSI International Journal of Computer Science Issues*, 9(5), 2012.

[10] Arbor Networks. Worldwide infrastructure security report. http://www.arbornetworks.com/research/infrastructure-security-report, 2012.

[11] AVG. New AVG study reveals smartphone users not aware of significant mobile security risks. http://mediacenter.avg.com/content/mediacenter/en/news/new-avg-study.html, Feb 2011.

[12] A. Baliga, J. Bickford, and N. Daswani. Triton: A carrier-based approach for detecting and mitigating mobile malware. *Journal of Cyber Security*, 3(4):181–212, July 2014.

[13] A. Barbuzzi, F. Ricciato, and G. Boggia. Discovering parameter setting in 3G networks via active measurements. *IEEE Communications Letters*, 12(10):730–732, Oct. 2008.

[14] BBC. Cyber thieves profit via the mobile in your pocket. `http://www.bbc.co.uk/news/technology-20080397`, Nov 2012.

[15] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[16] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: evidence and implications. In *Proc. IEEE INFOCOM'99*, volume 1, pages 126–134, Mar 1999.

[17] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani. Crowdroid: behavior-based malware detection system for android. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices (SPSM '11)*, pages 15–26, Chicago, Illinois, USA, 2011. ACM.

[18] Canalys. Smart phones overtake client PCs in 2011. `http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011`, Feb 2012.

[19] Canalys. Mobile device market to reach 2.6 billion units by 2016. `http://www.canalys.com/newsroom/mobile-device-market-reach-26-billion-units-2016`, Feb. 2013.

[20] R. Caruana, N. Karampatziakis, and A. Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 96–103. ACM, 2008.

[21] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

[22] Cisco. Cisco visual networking index: Forecast and methodology, 2013-2018. White Paper, Jun 2014.

[23] A. Coluccia, A. D'alconzo, and F. Ricciato. Distribution-based anomaly detection via generalized likelihood ratio test: A general maximum entropy approach. *Comput. Netw.*, 57(17):3446–3462, Dec. 2013.

[24] S. Corner. Angry Birds + Android + ads = network overload. http://www.itwire.com/business-it-news/networking/47823, June 2011.

[25] M. Donegan. Operators urge action against chatty apps. Light Reading Report, Jun 2011.

[26] W. Enck, P. Traynor, P. McDaniel, and T. L. Porta. Exploiting open functionality in SMS-capable cellular networks. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05)*, pages 393–404, Nov. 2005.

[27] C. Gabriel. DoCoMo demands Google's help with signalling storm. http://www.rethink-wireless.com/2012/01/30/docomo-demands-googles-signalling-storm.htm, Jan. 2012.

[28] E. Gelenbe. Random neural networks with negative and positive signals and product form solution. *Neural Comput.*, 1(4):502–510, Dec. 1989.

[29] E. Gelenbe. Learning in the recurrent random neural network. *Neural Comput.*, 5(1):154–164, Jan. 1993.

[30] E. Gelenbe. The first decade of g-networks. *European Journal of Operational Research*, 126(2):231–232, October 2000.

[31] E. Gelenbe and O. H. Abdelrahman. Time-outs and counters against storms. *Submitted for publication*, Aug 2014.

[32] E. Gelenbe and G. Loukas. A self-aware approach to denial of service defence. *Computer Networks*, 51(5):1299–1314, April 2007.

[33] E. Gelenbe and R. Muntz. Probabilistic models of computer systems - part I (exact results). *Acta Informatica*, 7(1):35–60, 1976.

[34] G. Gorbil, O. H. Abdelrahman, and E. Gelenbe. Storms in mobile networks. In *Proceedings of the 9th ACM Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet'14)*, pages 119–126, Sept. 2014.

[35] A. Gupta, T. Verma, S. Bali, and S. Kaul. Detecting MS initiated signaling DDoS attacks in 3G/4G wireless networks. In *Proceedings of 5th International Conference on Communication Systems and Networks (COMSNETS'13)*, pages 1–6, 2013.

[36] D. Iland, A. Pucher, and T. Schäuble. Detecting android malware on network level. Technical report, UC Santa Barbara, 2012.

[37] F. Jiang, Y. Sui, and C. Cao. An information entropy-based approach to outlier detection in rough sets. *Expert Systems with Applications*, 37(9):6338–6344, 2010.

[38] N. Jiang, Y. Jin, A. Skudlark, W.-L. Hsu, G. Jacobson, S. Prakasam, and Z.-L. Zhang. Isolating and analyzing fraud activities in a large cellular network via voice call graph analysis. In *Proceedings of the 10th international conference on Mobile systems, applications, and services (MobiSys '12)*, pages 253–266, Low Wood Bay, Lake District, UK, 2012. ACM.

[39] E. K. Kim, P. McDaniel, and T. Porta. A detection mechanism for SMS flooding attacks in cellular networks. In A. D. Keromytis and R. Pietro, editors, *Security and Privacy in Communication Networks*, volume 106 of *LNICST*, pages 76–93. Springer Berlin Heidelberg, 2013.

[40] J. Körner. Coding of an information source having ambiguous alphabet and the entropy of graphs. *Transactions of the Sixth Prague Conference on Information Theory*, pages 411–425, 1973.

[41] A. Ksentini, Y. Hadjadj-Aoul, and T. Taleb. Cellular-based machine-to-machine: overload control. *IEEE Network*, 26(6):54–60, November 2012.

[42] P. P. Lee, T. Bu, and T. Woo. On the detection of signaling DoS attacks on 3G wireless networks. In *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM'07)*, pages 1289–1297, May 2007.

[43] C. Lever, M. Antonakakis, B. Reaves, P. Traynor, and W. Lee. The core of the matter: Analyzing malicious traffic in cellular carriers. In *Proceedings of Network and Distributed System Security Symposium (NDSS'13)*, pages 1–16, San Diego, CA, USA, Feb 2013. Internet Society.

[44] J. Li, W. Pei, and Z. Cao. Characterizing high-frequency subscriber sessions in cellular data networks. In *Proceedings of IFIP Networking Conference*, pages 1–9, Brooklyn, NY, May 2013.

[45] Lookout Mobile Security. The bearer of BadNews. `https://blog.lookout.com/blog/2013/04/19/the-bearer-of-badnews-malware-google-play`, Apr 2013.

[46] D. Maslennikov. Mobile malware evolution: Part 6. Technical report, Kaspersky Lab, Feb 2013.

[47] D. Maslennikov and Y. Namestnikov. Kaspersky security bulletin 2012: The overall statistics for 2012. http://www.securelist.com/en/analysis/204792255/Kaspersky_Security_Bulletin_2012_The_overall_statistics_for_2012, Dec. 2012.

[48] I. Murynets and R. Piqueras Jover. Crime scene investigation: SMS spam data analysis. In *Proceedings of the 2012 ACM conference on Internet measurement conference (IMC '12)*, pages 441–452, Boston, Massachusetts, USA, 2012. ACM.

[49] Nokia Networks. New network-based security solution from NSN helps protect smart device users against malware. http://networks.nokia.com/news-events/press-room/press-releases/new-network-based-security-solution-from-nsn-helps-protect-smart-device-users-against-malw, Feb 2014.

[50] NSN Smart Labs. Understanding smartphone behavior in the network. White paper, http://www.nokiasiemensnetworks.com/sites/default/files/document/Smart_Lab_WhitePaper_27012011_low-res.pdf, Jan 2011.

[51] G. Oke, G. Loukas, and E. Gelenbe. Detecting denial of service attacks with bayesian classifiers and the random neural network. In *Proceedings of Fuzz-IEEE 2007*, pages 1964–1969, London, UK, July 2007.

[52] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Characterizing radio resource allocation for 3G networks. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement (IMC'10)*, pages 137–150, Nov. 2010.

[53] Z. Qian, Z. Wang, Q. Xu, Z. M. Mao, M. Zhang, and Y.-M. Wang. You can run, but you can't hide: Exposing network location for targeted DoS attacks in cellular networks. In *Proceedings of the 19th Annual Network and Distributed System Security Symposium (NDSS'12))*, Feb. 2012.

[54] G. Redding. OTT service blackouts trigger signaling overload in mobile networks. http://blogs.nsn.com/mobile-networks/2013/09/16/ott-service-blackouts-trigger-signaling-overload-in-mobile-networks/, Sept. 2013.

[55] F. Ricciato. Unwanted traffic in 3g networks. *ACM SIGCOMM Comput. Commun. Rev.*, 36(2):53–56, Apr. 2006.

[56] F. Ricciato, A. Coluccia, and A. DAlconzo. A review of DoS attack models for 3G cellular networks from a system-design perspective. *Computer Communications*, 33(5):551–558, Mar. 2010.

[57] A.-D. Schmidt, F. Peters, F. Lamour, C. Scheel, S. A. Çamtepe, and c. Albayrak. Monitoring smartphones for anomaly detection. *Mobile Networks and Applications*, 14(1):92–106, 2009.

[58] C. Schwartz et al. Smart-phone energy consumption vs. 3G signaling load: The influence of application traffic patterns. In *Proc. 24th Tyrrhenian Int. W'shop Digital Communications (TIWDC'13)*, pages 1–6, Genoa, Italy, Sept. 2013.

[59] J. Serror, H. Zang, and J. C. Bolot. Impact of paging channel overloads or attacks on a cellular network. In *Proceedings of the 5th ACM Workshop on Wireless Security (WiSe'06)*, pages 75–84, Sept. 2006.

[60] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, and J. Wang. A first look at cellular machine-to-machine traffic: Large scale measurement and characterization. *SIGMETRICS Performance Evaluation Review*, 40(1):65–76, Jun 2012.

[61] G. Simonyi. Graph entropy: a survey. *Combinatorial Optimization*, 20:399–441, 1995.

[62] G. Simonyi. Perfect graphs and graph entropy. an updated survey. *Perfect graphs*, pages 293–328, 2001.

[63] T. Taleb and A. Kunz. Machine type communications in 3GPP networks: potential, challenges, and solutions. *IEEE Communications Magazine*, 50(3):178–184, March 2012.

[64] S. Timotheou. The random neural network: A survey. *The Computer Journal*, 53(3):251–267, 2010.

[65] P. Traynor, M. Lin, M. Ongtang, V. Rao, T. Jaeger, P. McDaniel, and T. La Porta. On cellular botnets: measuring the impact of malicious devices on a cellular network core. In *Proceedings of the 16th ACM conference on Computer and communications security (CCS '09)*, pages 223–234, Chicago, Illinois, USA, 2009. ACM.

[66] I. Vural and H. Venter. Mobile botnet detection using network forensics. In *Proceedings of the Third future internet conference on Future internet (FIS'10)*, pages 57–67, Berlin, Germany, 2010. Springer-Verlag.

[67] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang. An untold story of middleboxes in cellular networks. *ACM SIGCOMM Computer Communication Review - SIGCOMM'11*, 41(4):374–385, Aug. 2011.

[68] G. Yan, S. Eidenbenz, and E. Galli. SMS-watchdog: Profiling social behaviors of SMS users for anomaly detection. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection (RAID '09)*, pages 202–223, Saint-Malo, France, 2009. Springer-Verlag.