Providing Mobile Traffic Analysis As-a-service: Design of a Service-based Infrastructure to Offer High-accuracy Traffic Classifiers Based on Hardware Accelerators

Mario Barbareschi[1], Alessandra De Benedictis[1]
Antonino Mazzeo[1] and Antonino Vespoli[1]
[1]Department of Electrical Engineering and Information Technology
University of Naples Federico II,
Naples, NA 80125, Italy
name.surname@unina.it

JDIM
Journal of Digital Information Management

ABSTRACT: *Mobile traffic is significantly growing, thanks to the increased access capacity provided by 3G and 4G technologies and to the rising computing power of the latest smart devices. Due to the widespread diffusion of mobile applications that require and process sensitive customers' data, mobile traffic is more and more subject to security attacks. Recently, traffic analysis techniques are being successfully adopted to characterize, from a security point of view, applications' and networks' behaviour in order to detect and avoid intrusion attempts, malware injections and data theft. When applied to the mobile domain, such techniques have to cope on the one hand with the performance constraints posed by the limited devices' resources and, on the other hand, with the need for accurate and up-to-date traffic models, resulting from a continuous processing of meaningful traffic data and threat-related information.*

To face these issues, we propose a two-tier service-based traffic analysis infrastructure: at the mobile network layer, mobile devices run a high-accuracy hardware traffic analyser, which allows for the processing of large data sets in an energy-efficient way; at the service layer, a high-scale traffic analyser takes advantage of the correlation of traffic data involving heterogeneous and geographically distributed sources, in order to produce enhanced traffic models, which can be distributed to the devices in an on-demand fashion, according to the as-a-Service approach.

To show the feasibility of our proposal, we provide a case study based on the implementation of a decision tree-based traffic analyser on a Xilinx Zynq 7000 architecture, and present an overview of the service layer, by referring to a cloud infrastructure for its implementation.

Subject Categories and Descriptors:
C.1.4 [Parallel Architectures]: Mobile processors

General Terms: Mobile Traffic, Mobile Security

Keywords: Mobile Traffic Analysis, Traffic Analysis As-a-service, Hardware Accelerators, Mobile Security, Reconfigurable Computing

Received: 28 February 2015, Revised 4 April 2015, Accepted 9 April 2015

1. Introduction

Mobile traffic is significantly growing, thanks to the increased access capacity provided by 3G and 4G technologies and to the rising computing power of the latest smart devices. According to Cisco's Global Mobile Data Traffic Forecast Update [1], global mobile data traffic grew 81 percent in 2013, reaching 1.5 exabytes per month at the end of the year. Most of this traffic is generated by smart devices including smartphones, tablets and wearable devices, which are increasingly adopted in domains where sensitive information are exchanged and processed, such as entertainment, healthcare and business applications.

Traffic monitoring and traffic analysis are a fundamental means to acquire information about incoming flow for the detection of anomalous behaviour, viral infections and hacking attempts, in addition to being a powerful tool to

define accurate network models, to validate new protocols and applications, to diagnose network failures, and to enhance network performance and quality of service. Nevertheless, the application of traffic analysis techniques to the mobile domain is not straightforward, mainly due to the huge amount of data generated by distributed devices and to their limited resources. Moreover, due to applications' dynamicity, traffic models used by mobile devices should be periodically updated to include new threats and to take into account the feedback coming from other devices.

Machine learning algorithms have been widely adopted for traffic analysis, thanks to their flexibility and the capability of dynamically generating updated models from feedback knowledge. These techniques, which typically require high computational capabilities for large data sets, can be effectively implemented in hardware, by adopting for example hardware accelerators [2], which allow complex computations at reduced power compared to software approaches. However, even with a hardware traffic analyser, the efficacy of the traffic analysis mechanism is still limited as long as the traffic model is statically embedded in the hardware configuration. To overcome this issue, hardware reconfiguration techniques [19] can be adopted, which enable the dynamic update of the traffic model based on network behaviour and on received feedback.

With respect to the accuracy of traffic models used by devices, it is quite evident that the traffic involving a single device may not be enough to carry out complex evaluations, while the correlation of traffic data involving heterogeneous and geographically distributed sources may sensitively enhance the model used for traffic classification.

In such a scenario, where a direct exchange of information among nodes is clearly not feasible and too expensive, a service-based approach provides an effective solution to enable an easy sharing of nodes' information about unclassified flows or detected threats and the dissemination of traffic model updates to heterogeneous and physically distributed nodes. In this paper, we propose a two-tier service-based traffic analysis infrastructure. At the *mobile network layer*, mobile devices run a high-accuracy hardware traffic analyser that allows for the processing of large data sets in an energy-efficient way. At the *service layer*, a high-performance processing entity collects traffic data, mis-classification events and threats information to continually update the traffic model, which is distributed to the devices according to an as-a-Service approach.

To show the feasibility of our proposal, we present and discuss, for the *mobile network layer*, the implementation of a decision tree-based traffic analyser on a Xilinx Zynq 7000 architecture and, for the *service layer*, the service layer APIs.

In particular, for the service layer we refer to a cloud infrastructure, since it features flexible high-performance computing and high extensible storage and is particularly suited to dynamically process huge amount of distributed data.

The remainder of this paper is structured as follows. In Section 2, we present and overview of adoption of traffic analysis in the mobile domain for security purposes. In Section 3, we illustrate the architecture of our solution for efficient traffic analysis in large and geographically distributed mobile networks, while in Section 4 we discuss some experimental results gathered through simulation campaigns. Finally, in Section 5, we draw some conclusions and show future directions.

2. Traffic Analysis and Network Security

In network management, traffic analysis is the basic task to guarantee flow prioritization, QoS, traffic policy, traffic diagnosis and so on. Moreover, the analysis turns out to be useful for planning future network architectures, especially in the mobile domain.

Smartphone traffic analysis has been recently addressed by several research papers, aimed at profiling the involved application protocols. In [4], the authors proposed a method to collect raw information about traffic using an application installed on end-user terminals. The aim is to collect data to infer offline characteristics about the smartphones' traffic. By exploiting a very accurate analysis, traffic controllers can bring several benefits to network security, by providing powerful tools for intrusion detection and access control. Attackers get unauthorized access to the devices using the Internet: they typically publish malwares through applications or malicious websites and, once infiltrated in the device, attempt to get the resource control and/or collect, redirect or delete data from the memory. Clearly, in this scenario, traffic analysis should be an online task in order to be able to avoid an attack. Several security applications of traffic analysis exist, such as those described in [21-22], which address the e-health domain and use semantic methodologies to classify and protect sensitive data.

Several approaches have been proposed to provide smart devices with intrusion detection features. For instance, in [7] the authors described a security framework applied to the smartphone architecture, in which every security issue was covered by a specific module of the proposed framework.

The authors devised the introduction of an IDS for network monitoring activities against policy violations, but did not provide any implementation of this module. Similarly, the authors of [8] proposed an IDS tailored to smartphones (SIDS), that runs directly on mobile devices to improve the protection for some security threats. Even in this case, no details were given about the implementation, nor on the impact on the smartphone systems. Authors in [5]

discussed a malware detection technique for the Android operating system by analysing the URLs.

Conversely, in [6] the authors proposed a technique to analyse the Internet traffic by exploiting a machine learning approach.

Indeed, recently, in the research community, the machine learning-based classification has been receiving a lot of attention, since it provides a high accuracy in classification and a great adaptability in dynamic contexts, such as that of traffic analysis [9], [10].

The approach requires the availability of a set of pre-classified packets (training set) to automatically extract knowledge by means of a learning algorithm. The resulting model (i.e., the predictor model) is actually used to classify the packets of interest, by exploiting a predictor algorithm. When the initial set is enriched with new entries (e.g. new security attacks, new traffic flows, etc.), an updated model can be generated.

Usually, the update is required when a significant number of mis-classifications occurs. The literature proved the Decision Trees (DTs) machine learning approach to be one of the most effective in traffic analysis, since it is able to achieve high efficacy in discriminating traffic behaviour. DTs consist in internal nodes, containing rules, and leaves, which are labelled with a classification value: basically, a DT represents a set of rules which classify data according to the conditions specified by the nodes. For instance, a condition typically specifies an attribute (feature) in the dataset, a relational operator (such as $<$, $\leq$, $\neq$) and a constant, but it may contain a more complex expression. Starting from the root of the tree, the evaluation of such conditions determines which child node has to be selected for next evaluations. At the end, when a leaf has been reached, its label is returned as the decision value of classification.

In traffic analysis, DTs can intuitively be considered as a list of rules that have to be applied to data traffic in order to predict the traffic behaviour. Typically, the prediction algorithm applies a rule on a traffic/packet feature (e.g. TCP port number, IP address, packet size, and so on). The classification results could be either the traffic predicted behaviour or an action that has to be accomplished. For instance, a network firewall based on DT can classify a packet by deciding whether to drop it or to report an anomaly.

As said, mobile devices, such as the smartphones, continuously receive data from the Internet. This makes a software approach for the prediction too expensive, due to limited computational resources and battery. Even with more powerful computation capability, the traffic analysis task would still negatively affect the usage of resources shared with the other system tasks, while draining the battery very fast.

To mitigate the intrusiveness of a software approach, we proposed in [11] a hardware approach to the traffic analysis that allows to save computational resources for other tasks and to increase battery performance. Indeed, in our approach, each node is equipped with a hardware predictor module that analyses the incoming traffic to detect anomalies or intrusion attempts and rises an exception when a suspicious event is observed. The anomaly is then managed via a software routine, which makes a decision about the action to take.

This approach is less intrusive than a purely software one, and enables the processing of large data sets in an energy-efficient way but, as pointed out in Section 1, an analysis that is merely restricted to the traffic involving a single node and that exploits a static model is not accurate enough to protect a device from the various existing threats. Indeed, the accuracy of the traffic model, used by the hardware predictor module, is limited if the model is not periodically updated to take into account information about new threats and possible feedback deriving from the experience of other nodes. To cope with these issues, we devised the introduction of a further processing layer dedicated to aggregating mobile devices' traffic information and to generating and distributing up-to-date traffic models for the reconfiguration of part of the nodes' hardware architecture. In the next sections, we provide an overview of the proposed two-tier traffic analysis infrastructure and illustrate the main involved components and services.

3. The Proposed Two-Tier Traffic Analysis Infrastructure

Figure 1 shows the proposed traffic analysis infrastructure, composed of two layers, namely the *mobile network layer* and the *service layer.* At the mobile network layer, nodes run a high-accuracy hardware traffic analyser based on an embedded traffic model. On occurrence of specific events (e.g., inability in classifying a packet or a packet flow), a node may invoke the service layer through proper APIs, in order to get information about unclassified traffic and receive updates of the model, represented by a partial bitstream to re-configure the node's hardware with. At the service layer, traffic data involving heterogeneous and geographically distributed sources are correlated and processed with respect to the latest information about security threats and application vulnerabilities, to produce enhanced traffic models, which can be distributed to devices in an on-demand fashion according to the as-a-Service approach.

The adoption of the cloud computing paradigm is particularly suited to implement the service layer thanks to the availability of virtually unlimited resources, given the need for processing huge data sets coming from large networks. Therefore, in the remainder of this paper, we will refer to such an infrastructure to discuss the service layer organization.

In the following subsections, we will describe the two lay

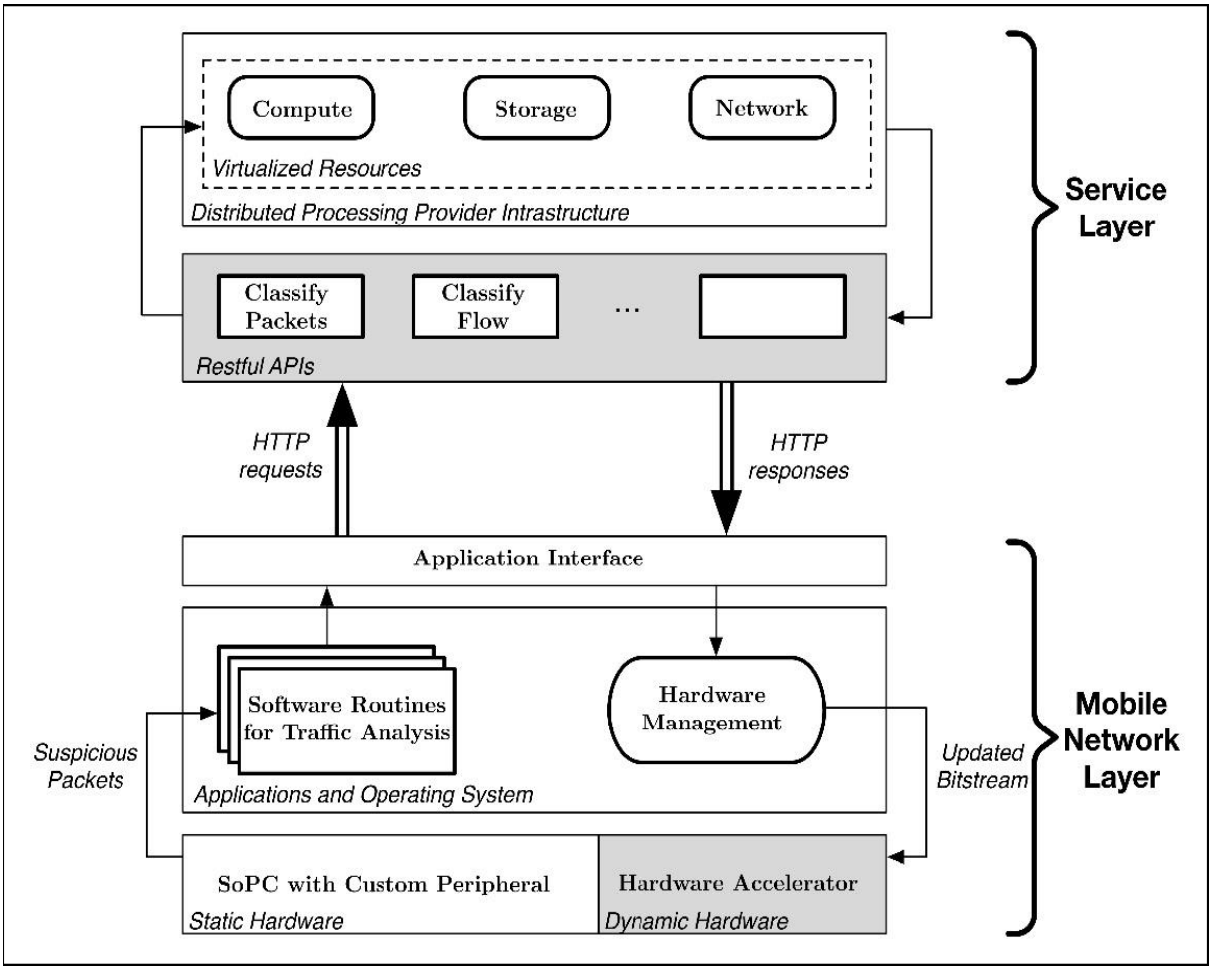

Figure 1. The proposed two-tier traffic analysis infrastructure

ers by detailing respectively the architecture of a mobile node and the services offered by the service layer.

3.1 The Mobile Network Layer

As mentioned in Section 2, in a previous work [3,11] we designed a hybrid HW/SW solution to implement an energy efficient traffic analyser on a System on Programmable Chip (SoPC). The hardware system is composed of (i) a static design (not programmable), including a SoC and some peripherals, such as the I/O interfaces and sensors, and (ii) a programmable logic, which can be realized through an FPGA structure and allows to define custom hardware components. The software portion of the system is represented by the operating system: due to its wide diffusion in the smartphone domain, we chose the well supported and easy to configure Google Android OS [12].

In such architecture, the hardware traffic analysis accelerator continuously executes the algorithm searching for anomalies or intrusion attempts. During this time, no additional resources are allocated onto the processing system by the operating system. When anomalies or attack attempts are detected, or in case of inability in classifying a packet/flow, the accelerator generates an interrupt in order to manage those events by means of software routines. An application has to be executed in order to apply a decision on the basis of the hardware prediction: for example, in the case of anomalous behaviour of the data traffic, the application may warn the user about the risk related to the device network activity or autonomously drop the dangerous packets.

In addition to guaranteeing high classification throughput, this approach is very low intrusive, because other software tasks never share the computational resources with the traffic analysis ones, except for the situations that need a software or user intervention, which are kept quite rare by the machine learning approach adopted by the system.

To implement the mobile network layer, we leveraged the above discussed device architecture and enhanced it by introducing the dynamic reconfiguration of the hardware portion representing the predictor model with a partial bitstream obtained by the service layer. In particular, in the enhanced version, when a node is unable to classify some flow and asks the service layer for an updated traffic model, the Android OS takes care of managing the reconfiguration of the programmable logic of the device with the delivered bitstream. Clearly, the distribution of the partial bitstream over the public network opens up potential security threats (e.g., FPGA malicious re-programming, Intellectual Property theft, etc.): this issue was analysed in [19], where the authors proposed an

infrastructure for the secure distribution of hardware digital content, which can be successfully exploited to deliver model updates in a trustworthy way.

In the next subsection, we provide some details about the node architecture.

3.1.1 The Node Architecture

To provide a proof of the effectiveness of the proposed model, we realized a working prototype of a node belonging to the mobile network layer. In particular, we adopted the Avnet ZedBoard [20], a development board equipped with a Xilinx Zynq-7020 SoPC, an USB OTG 2.0, 10/100/1G Ethernet, a SD Card support, a 256 MB Flash memory, 512 MB RAM, HDMI and a VGA interface. The Zynq SoPC is an architecture composed of a programmable logic, based on Xilinx Artix FPGA technology, and a static hardware core processor based on the ARM Cortex A9 dual core.

To implement the hardware traffic analyser, we exploited a hardware accelerator of the DT prediction algorithm, as illustrated in [2]. The model is generated by means of C4.5 algorithm and it takes into account 12 features including protocol, port number, size, etc. The model extraction and synthesis can be automatized, as illustrated in [14]: starting from the dataset, a C4.5 learning algorithm extracts the model in a standard encoding format, such as PMML, that can be converted in a HDL language in order to be synthesized for a specific technology.

The hardware accelerator was integrated in the Zynq programmable logic. As shown in Figure 2, the traffic analyser receives packets from the ethernet DMA, while a control logic manages the packets manipulation and their analysis. Once classified, the packets are stored in the available memory.

When needed, the embedded traffic model can be updated with the one provided by the service layer by properly using the Self Dynamic Partial Reconfiguration (SDPR) technique. Indeed, the Zynq family is equipped with a Hardware Internal Configuration Access Port (HWICAP), whereby the FPGA can partially reconfigure itself, without interfering with other hardware subsections, when provided with a partial bitstream.

As previously said, the software part of the node is represented by the Android OS. In order to enable the execution of such operating system on our hardware architecture, a porting process to the Zynq architecture was carried out. To this aim, we opened a public project called Zedboard [15], which provides a complete guide to run Android on the board and supports the community in prototyping and studying Android with hardware special units on Zynq FPGA.

In Section 4, we will briefly discuss the performance and accuracy of the described architecture, while in the next subsection we are going to provide some insight into the service layer.

3.2 The Service Layer

As anticipated, the service layer can be designed as a cloud computing infrastructure, able to cope with flexibility, scalability and high-computation and storage requirements typical of a large-scale traffic analysis application.

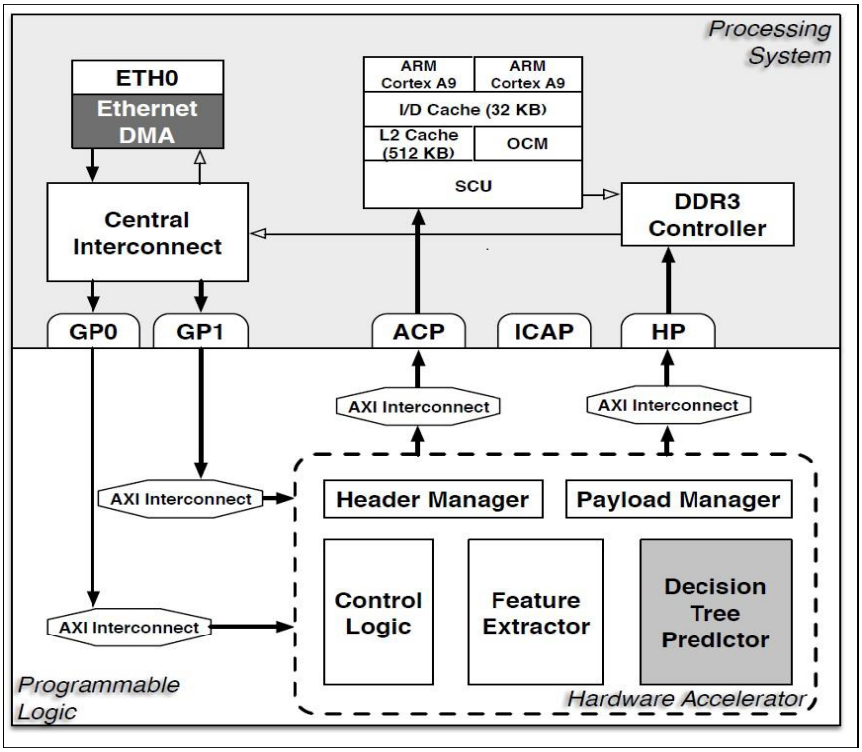

Figure 2. Hardware schematic of the mobile node: implementation on a Xilinx Zynq

The way our cloud-based traffic analysis application is to be offered to customers in our context is well represented by the Software-as-a-Service (SaaS) model. From the business point of view, in our architecture, a cloud provider offers a traffic analyser software to its customers, represented by the mobile devices owners, on a per-user/ pay-as-you go basis. As an example, the cloud provider may allow all registered (free) customers to download a basic configuration for the hardware accelerator implementing the local traffic analyser and to obtain periodic updates for it, taking advantage in turn of the amount of precious data coming from the devices needed to enhance the model. On the other hand, premium customers may be provided with more sophisticated models and with more frequent updates.

Several technologies may be adopted to implement the mentioned cloud services. As our focus in on the whole architecture, and we are not interested in showing the actual implementation of the services themselves, in this paper we assume that the cloud provider offers RESTful based interfaces (RESTful APIs) for its services [13], and discuss those APIs.

RESTful services are currently one of the most popular trends in developing cloud services. RESTful APIs follow the SOA model and, therefore, are often used by web service-based software architectures via XML or JSON for integration purposes. They are consumed via an Internet browser or by web servers and are a very suitable solution for the remote consumption of data processing services by heterogeneous users. RESTful applications use HTTP methods (PUT, GET, POST, DELETE) to realize the four CRUD (Create/Read/Update/Delete) operations, and can be very powerful, especially when used in combination with JSON messages. JSON (Java Script Object Notation) is a lightweight data interchange format that specifies parameters in the key-value pair fashion. It is very easy for humans to understand and for machines to parse and process.

The service layer can be easily described through the set of exposed RESTful APIs, which can be invoked by nodes to obtain predictor model updates or single responses to a classification request. In our solution, such APIs adopt JSON to specify the parameters contained in the HTTP requests and responses exchanged among the cloud services and the mobile nodes. We envisioned the following main APIs:

• **Classify packet**: obtain a classification for a given packet, identified by means of a set of features;

• **Classify flow**: obtain a classification for an entire packet flow, identified by means of a set of features;

• **Get Updated Model**: obtain the new predictor model in a well-known format (e.g., Predictor Model Markup Language - PMML);

• **Get Updated Configuration**: obtain the new predictor model in form of a partial bitstream to load onto the device;

• **Send flow**: send the features related to a given flow (this actually increases the volume of data available to the service to enhance its model).

Each of the APIs has its own URL and HTTP method (GET, POST, PUT, or DELETE). In particular, the *classify packet* and *classify flow*, along with the *send flow* APIs adopt the POST method, to push information about the packets to classify, while the *get updated configuration* API uses the GET HTTP method. The *get updated model* API has been introduced to extend the approach also to cases of pure software systems willing to take advantage of the cloud distributed computation capabilities to perform high-accuracy traffic analysis tasks.

To make an API call, the client running on the mobile device has to perform a HTTP request. Method parameters are passed in the URL query string or in the message body, depending on the HTTP method. API responses are returned as JSON objects, which typically include metadata (i.e., the response status code and error message, if any) and the actual response, which contains the actual data formatted in the form *key:value*. In the following, an example of request and response for the *classify packet* method is shown. As illustrated in the following section, our predictor takes into account a data model with12 features and 22 classes, representing different levels of alarms. In the following example, we show the format of a request for the *classify packet* API, in which the features of a packet to classify are included. The response will contain the predicted class resulting from the processing carried out by the service layer.

4. Evaluation of the Proposed Infrastructure

In this section, we present some experimental and simulation results obtained from the analysis of both the hardware classifier, embedded in the mobile devices, and the high-scale traffic classifier implemented at the service layer. The aim of this analysis is to demonstrate the feasibility and effectiveness of the proposed infrastructure and to study its behaviour with respect to increasing network size and workload and in presence of a variable threat characterization.

4.1 Analysis on the Mobile Network Layer

As discussed in [3], the synthesis of the hardware accelerator described in Section 3.1.1 produces a design occupying an area of about 3500 slices and the hardware traffic classifier is characterized by a maximum throughput of 29.33 Gbps.

The bottleneck of the accelerator is represented by the feature extractor component, which extracts the relevant features from a packet flow for its analysis. Such component is able to reach the maximum throughput of 15.56 Gpbs.

Synopsis: POST {Well Known URI of the Service}	
POST /request HTTP/1.1 Accept: application/jsonrequest Content-Encoding: identity Content-Length: xx Content-Type: application/jsonrequest Host: example.org {“feature1”:2.34,“feature2”:6,“feature3”:9844,.. . ,“feature12”:0.65}	HTTP/1.1 200 OK Content-Type: application/jsonresponse Content-Length: xx {“predicted class” : 9}

Table 1. Example of JSON request/response for the *classify packet* API

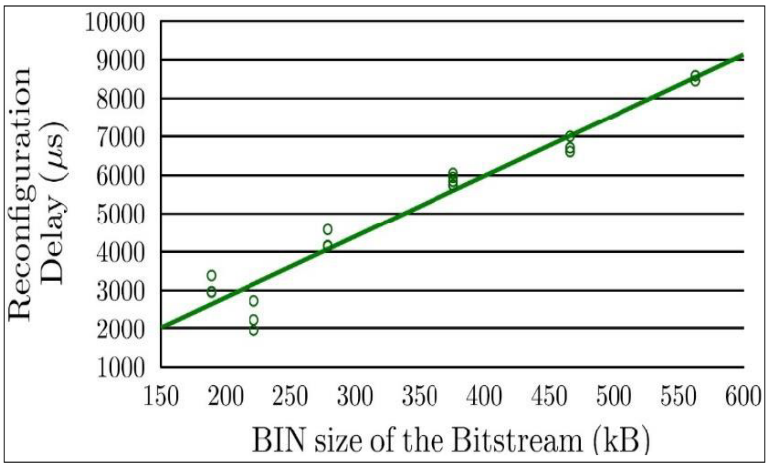

Figure 3. Latency introduced by reconfiguration on mobile devices

In this paper, we are interested in evaluating the latency introduced by the dynamic reconfiguration task, as it affects the efficacy of the local traffic analysis operations: indeed, the classifier component is not available throughout the reconfiguration interval, therefore all the traffic involving the device during this time must be buffered and analysed later on. Clearly, possible threats involving such traffic during reconfiguration will not be detected timely.

In the proposed architecture, on the occurrence of an unclassification event, the features identifying the unclassified packets are forwarded to the service layer by invoking the classify flow RESTful API, to enrich the training set for future improvement of the predictor model. The service layer may return an updated version of the predictor model, in form of a bitstream to be used to reconfigure the device dynamic hardware portion. In order to get an idea of the reconfiguration overhead, we measured the time needed to configure a partial bitstream by means of HWICAP, the internal configuration access port available of the Zynq programmable logic. Hence, we report in Figure 3 the reconfiguration latency against the bitstream size. To obtain such measurements, we deployed an Android application that manages the HWICAP by seeing it as a file. Then, we measured the time required to copy an updated bitstream file, received by the device in the JSON body of an HTTP response, onto the HWICAP one. As shown, the reconfiguration time linearly varies with the bitstream size. For our purposes, the delay introduced by the reconfiguration process is quite acceptable, because the hardware we adopted can buffer some packets to avoid their loss.

4.2 Analysis on the Service Layer

In order to evaluate the performance of the service layer, we realized a prototype application that provides some services, such as the acquisition of traffic flow from devices and the model update, and executes the business logic.

The server interface collects packets and/or flows that were marked as suspicious by devices because they were unable to classify them, and launches a further analysis based on the latest created traffic model. If the suspicious flow or packet is recognized as either malicious or non-malicious, it is discarded by the server and the device is notified about the classification result and is possibly updated with the latest model available. Otherwise, the involved traffic is stored in a queue for further analysis, in order to determine its nature, i.e., in order to establish whether it represents a real threat for mobile devices or not. This task can be accomplished by an expert teamwork or by semi-automatic procedures, but we do not discuss it here because it is out of the scope of this paper.

New information about traffic behaviour extends the original

dataset available on the server so that new knowledge can be extracted, by means of a machine learning algorithm, and new updated models can be exploited for traffic analysis.

Knowledge extraction needs high computational resources, since the involved dataset contains a huge amount of traffic data. Similarly, the update of the traffic model on mobile devices is an energy and time consuming task, and should not be too frequent. Hence, the service layer needs a model update scheduling that has to take into account: (i) the model aging, (ii) the global amount of unclassified traffic and (iii) the amount of malicious packets and flows that are added to the initial dataset, i.e. which are not yet exploited for the generation of new models. In our prototype, we considered three different model update policies to take into account such features.

We implemented the prototype using the Java language, as we are not strictly interested in the timing performance, but only on the application behaviour under different workload conditions. Hence, we exploited the multithreading approach to simulate devices' network traffic, by varying the packets throughput, the devices population, and the suspicious and malicious packets' and flows' probabilistic distribution. In our setup, we exploited a real network traffic dataset that we partitioned into a *learning set*, used to generate the first traffic model, and a *test set*, which we used to feed the simulated mobile devices. In particular, to carry out our simulation experiments, we picked 50 threats over the available 150 threats to build the learning set, and used the remainder to simulate a real workload environment. As for the data traffic distribution, we modelled the threats arrival distribution with a bathtub curve, which is quite close to a typical distribution of application vulnerabilities: indeed, the vulnerabilities reach the highest peak right after the first update release, then they decrease with subsequent software upgrades and patches, and finally they increase due to application aging and to the improvement of attacker skills. The distribution for the non malicious but suspicious traffic instead was modelled as uniform, since it includes traffic generated by new applications or new released Internet protocols.

As anticipated, the traffic model processed by the service layer is updated according to a scheduling policy that evaluates the first applicable condition among the following: (i) the model aging threshold exceeds 4 days, (ii) the number of new traffic flows/packets inserted in the analyser queue exceeds 2000, (iii) the number of new detected threats exceeds 10.

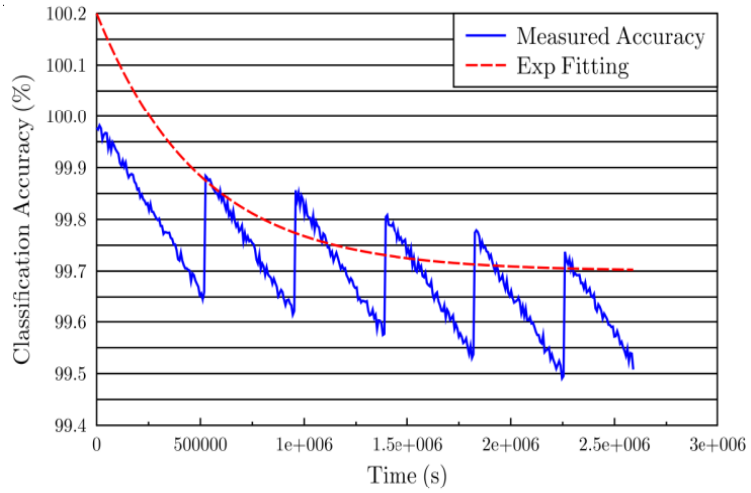

Figure 4. Analysis of the accuracy reached thanks to dynamic reconfiguration

In our simulations, we considered a time interval of about 1 month and let the device population and threat distribution vary to measure accuracy. In particular, in Figure 4, we show how the traffic analysis accuracy varies over time for a large network of 6 million devices. As shown, due to the chosen probability distribution, the accuracy (blue curve) presents a generally decreasing trend. Moreover, accuracy peaks can be observed in correspondence to update events, thanks to the acquisition of new knowledge about traffic. Note that the time between two subsequent updates is not constant due to the different scheduling policies that are activated by our application. Clearly, the accuracy does not reach the maximum possible value at each update because of the delay experienced in the update process: when one of the three policies is triggered, a learning task is scheduled on the server, but since its execution takes a certain amount of time, in this interval new traffic information is not taken in to account. Hence, when the new traffic model is released, the evaluated accuracy does not achieve the original values. The peaks' trend (green curve) is exponentially decreasing to a stable value around 99.6%.

5. Conclusion

In this paper, we illustrated an innovative architecture to improve mobile devices' security through the adoption of a flexible and efficient two-tier traffic analysis infrastructure. At the mobile network layer, mobile devices run a high-accuracy hardware traffic analyser based on a decision-tree algorithm. At the service layer, a high-scale traffic analyser takes advantage of the correlation of traffic data

involving heterogeneous and geographically distributed sources, in order to produce enhanced traffic models, which can be distributed to the devices in an on-demand fashion, according to the as-a-Service approach. We developed a prototype both of the hardware classifier and of the traffic analysis service and conducted several experiments and simulations to show the feasibility and effectiveness of our approach.

Thanks to its flexibility and suitability in the field of constrained devices applications, the synergy between hardware-based high-performance applications and high-level scalable services can be successfully adopted to efficiently implement adaptive security and in particular of Moving Target Defense (MTD) techniques, which have recently emerged as powerful tools for thwarting cyber attacks and consist in continually changing a systems attack surface to increase the uncertainty of attackers and limit the exposure of vulnerabilities.

Such techniques have already been used to protect resource constrained devices [16], [17] and may significantly take advantage of the existence of a powerful

6. References

[1] Index, C. V. N., (2013). Global mobile data traffic forecast update, 2013-2018,*Cisco white paper*.

[2] Amato, F., Barbareschi, M., Casola, V., Mazzeo, A. (2014). An fpga-based smart classifier for decision support systems, *In*: *Intelligent Distributed Computing VII. Springer*, 289–299.

[3] Barbareschi, M., Mazzeo, A., Vespoli, A., Network traffic analysis using android on a hybrid computing architecture, **In**: *Algorithms and Architectures for Parallel Processing. Springer*, 141–148.

[4] Falaki, H., Lymberopoulos, D., Mahajan, R., Kandula, S., Estrin, D. (2010). A first look at traffic on smartphones, *In*: Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. *ACM*, 281–287.

[5] Arora, A., Garg, S., Peddoju, S.K. (2014). Malware Detection Using Network Traffic Analysis in Android Based Mobile Devices, *Next Generation Mobile Apps, Services and Technologies (NGMAST)*, Eighth International Conference on , 66,71, 10-12 September.

[6] Zaman, M., Siddiqui,, T., Amin, M. R., Hossain, M. S. (2015). Malware detection in Android by network traffic analysis, *Networking Systems and Security (NSysS)*, 2015 International Conference on , 1,5, 5-7 January.

[7] Luo, H., He, G., Lin, X., Shen, X. (2012). Towards hierarchical security framework for smartphones, *In*: Communications in China (ICCC), 2012 1st IEEE International Conference on, 214–219. .

[8] Salah, S., Abdulhak, S. A., Sug, H., Kang, D.-K., Lee, H. (2011). Performance analysis of intrusion detection systems for smartphone security enhancements, In Mobile IT Convergence (ICMIC), 2011 International Conference on. IEEE,15–19.

[9] Jeong, H., Yoo, Y., Yi, K. M., Choi, J. Y. (2014).Two-stage online inference model for traffic pattern analysis and anomaly detection, *Machine Vision and Applications*, 25 (6) 1501–1517.

[10] Ye, W., Cho, K. (2014). Hybrid p2p traffic classification with heuristic rules and machine learning, *Soft Computing*, 1–13.

[11] Barbareschi, M., Mazzeo, A., Vespoli, A. (2015). Malicious Traffic Analysis on Mobile Devices: a Hardware Solution, *International Journal of Big Data Intelligence*.

[12] Inc, G. (2007). Android source web-site. [Online]. Available: https://source.android.com/

[13] Fielding, R. T. (2000). REST: architectural styles and the design of network-based software architectures," Doctoral dissertation, University of California, Irvine. [Online]. Available: http://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm

[14] Amato, F., Barbareschi, M., Casola, V., Mazzeo, A., Romano, S. (2013). Towards automatic generation of hardware classifiers, In: *Algorithms and Architectures for Parallel Processing*. *Springer*, 125–132.

[15] Barbareschi, M., Mazzeo, A., Vespoli, A. (2013). Zedroid: Android 2.2 (froyo) porting on Zedboard, September 2013. [Online]. Available: http://wpage.unina.it/mario.barbareschi/zedroid/index.html

[16] Casola, V., De Benedictis, A., Albanese, M. (2013). A moving target defense approach for protecting resource-constrained distributed devices, In: *Information Reuse and Integration* (IRI), 2013 IEEE 14th International Conference on. IEEE, 2013, 22–29.

[17] A multi-layer moving target defense approach for protecting resource-constrained distributed devices, *In*: *Integration of Reusable Systems. Springer International Publishing*, 299–324.

[18] Barbareschi, M., Battista, E., Mazzeo, A., Venkatesan, S. (2014). Advancing wsn physical security adopting tpm-based architectures, In: *Information Reuse and Integration* (IRI), 2014 IEEE 15th International Conference on, August.

[19] Cilardo, A., Barbareschi, M., Mazzeo, A. Secure distribution infrastructure for hardware digital contents, *In*: *Computers & Digital Techniques*, IET 8 (6) 300-310.

[20] The AVNET ZedBoard. Online Resource: http://www.em.avnet.com/en-us/design/drc/Pages/Zedboard.aspx.

[21] Amato, F., Casola, V., Mazzocca, N., Romano, S. (2013). A semantic approach for fine-grain access control of e-health documents. *Logic Journal of IGPL*, 21 (4) 692-701.

[22] Amato, F., Casola, V., Mazzeo, A., Romano. S.

(2010). A semantic based methodology to classify and protect sensitive data in medical records. *In*: *Information Assurance and Security (IAS)*, Sixth International Conference on 240-246. IEEE.

7. Author Biographies

Mario Barbareschi is currently a Ph.D student in Computer and Automation Engineering at University of Naples Federico II, Italy, under the supervision of prof. Antonino Mazzeo. He graduated, cum laude, in Computer Engineering from the University of Naples Federico II in 2012. He is a member of the Embedded System Lab and SecLab. His research activitiesa are focused on Cyber Physical Systems with particular emphasis on security and embedded systems devices based on the FPGA technology.

Alessandra De Benedictis got a Ph.D. in Computer and Automation Engineering in 2013 at the University of Naples Federico II, where she currently works as a post-doctoral fellow. She received a B.S. degree and an M.S. degree in Computer Engineering, both cum laude, from the University of Naples Federico II in 2006 and 2009 respectively. She has been working for some years on the design and evaluation of secure architectures for the protection of distributed resource-constrained devices, with particular focus on wireless sensor and ad-hoc networks. Recently, she has started working on new topics including the analysis and application of moving target defense techniques and the negotiation and dynamic enforcement of security solution in the cloud environment.

Antonino Mazzeo is Full Professor at the University of Naples Federico II, Napoli, Italy, where he teaches computer architectures. He led major research programs in conjunction with international universities, research agencies (CNR, ASI and EC), and technology leading industries in Italy and abroad. He has wide experience in the field of complex systems modelling, embedded systems, general and special purpose parallel architectures, and performance evaluation.

Antonino Vespoli is a computer engineering researcher at University of Naples Federico II. He is a member of the Embedded System Lab and his research activities are focused on mobile Cyber Physical Systems. Currently, he is experiencing Google Android operating system and new FPGA hybrid architectures.