Towards Ad-Hoc Cloud Based Approach for Mobile Intrusion Detection

Toufic Dbouk, Azzam Mourad
Dep. of Comp. Science and Math.
Lebanese American University
Beirut, Lebanon
{toufic.dbouk, azzam.mourad}@lau.edu.lb

Hadi Otrok
Dep. of Elec. and Comp. Eng.
Khalifa University
Abu Dhabi, UAE
hadi.otrok@kustar.ac.ae

Chamseddine Talhi
Dep. of Soft. Eng. and IT.
Ecole de Technologie Superieure
Montreal, Quebec
chamseddine.talhi@etsmtl.ca

***Abstract*—As the usage of smart devices is increasing, malware affecting such devices is rapidly evolving as well. Security risks affecting the confidentiality, integrity, and privacy of smart devices are rapidly emerging. Mobile security suites exist to defend the device against malware and other intrusions. However, they require extensive resources which is a constraint of the device itself. In this paper, we address the problem of intrusion detection for smart devices taking into account the devices' limited resources such as energy, CPU usage, and internet connectivity. We provide an ad-hoc mobile cloud based intrusion detection framework that takes advantage of Wi-Fi Direct as means of achieving connectivity and sharing resources and services for providing security. The proposed framework allows exchange of data with or without the availability of internet connection. The paper also provides experiments, carried out using real devices, showing various improvements of using our approach. Up to 61% and 40% enhancement in energy consumption and response time respectively is reached compared to local execution.**

***Index Terms*—offloading; ad hoc offloading; mobile cloud; Wi-Fi Direct; security as a service;**

I. Introduction

Cloud computing consists of using distributed remote servers over the internet to store, search, acquire, and process data instead of using local servers [1]. Such computing allows access to shared and easily provisioned computing resources, in an on-demand fashion, in order to perform a task and return its result on behalf of the user. In the context of mobile devices, a mobile cloud could be split in two models. Mobile ad hoc cloud and server based cloud. Server based cloud [2] consists of internet based remote servers that provide services for smart-devices to take advantage of or to perform a specific task. Such clouds benefit from high computational power and resources of remote servers. On the other hand, the ad hoc cloud model [3] is created by the effort of nearby volunteering devices allowing the devices to share resources and computations. Ad hoc clouds benefit from an intrinsic characteristic that enable them to be created anywhere and anytime due to the availability of volunteering resources (without requiring additional infrastructure) [4]. Therefore, such clouds positively contribute in reducing the cost needed for their setup due to their spontaneous creation. In the same manner, Wi-Fi Direct; a technology that allows devices to connect directly to each other without relying on an internet connection, is being used in several areas such as gaming, social media, and medical fields [5], [6], [7]. However, none has proposed its usage for creating mobile ad hoc cloud in the context of security detection and analysis.

In parallel, smart-devices are being heavily used in today's life especially with the world's convergence to E-commerce and cloud based computing. They are also being used as major and primary elements in mobile phone sensing networks [8], [9] providing a wider coverage area, easier deployment, strengthened features and a social aspect. They take advantage of resources and computational power available by remote servers via the internet. Services on the remote servers perform most, if not all, of the high intensive tasks on behalf of the mobile device. Since smart-devices are based on architectures that are similar to other computational devices such as desktops and laptops, they are subject to similar threats (*Worm, Trojan, Virus, Ransomware, Scareware, Botnet, Backdoor* etc...) [10]. In [11], the authors classify different Android malware and categorize their malicious activities between *Privilege Escalation, Remote Control, Financial Charge, and Information Gathering*. Moreover, the authors of [12], manage to state some of the ways allowing malware to trivially disguise and bypass detection such as *Repacking, Disassembling and Reassembling, Identifier Renaming, Data Encoding, and Junk Code Insertion*. Therefore, as stated by [13], smart-devices require special high level protection against such malware taking into consideration the device's limited resources such as battery and memory memory in addition to other constraint such as limited keyboard and bootstrapping PINs from Passwords.

Server-based cloud computing serves as a solution to overcome limitation caused by the device's resource constraints. Intrusion detection systems running on mobile devices impose several problems such as memory and CPU consumption, battery drainage, and energy usage. Furthermore, the response time of IDS on a mobile device suffers from low specifications of the device's hardware. In some cases the IDS mobile agent can result in exhaustiveness and irresponsiveness of the device which negatively impact the user. Several approaches [14], [15], [2] have been proposed to ameliorate the performance of IDS on smart-devices by offloading the analysis and scanning to off-device infrastructures such as clouds and remote servers. They take advantage of powerful cloud based infrastructure to

perform the detection while keeping a mobile agent to assist in providing a minimal security and communicating with servers. However, these approaches suffer from various limitations and drawbacks that put them behind. The shortcomings are summarized as follows:

1) Suffering from sketchy internet connections and being bound by the availability of internet connections.
2) Charging additional costs due to extra quota usage, especial mobile data, and server's subscriptions.
3) Overloading of the current mobile data infrastructure by connected devices and their usage of existing communication technologies. The authors of [16] expect a tremendous increase reaching 50 Billion connected devices by year 2020.

In this paper, we address the aforementioned problems by elaborating an ad hoc mobile cloud framework for intrusion detection. The proposed approach takes advantage of Wi-Fi Direct protocols to establish an ad hoc mobile cloud that serves as a cluster of nodes to monitor and detect intrusions. The framework embeds a set of interconnected modules: *Ad hoc Mobile Cloud Manager, Profiler, Offloading Manager, Communication Manager, Intrusion Detection Engine, and Distribution Controller*. These modules work together in order to analyze a given intrusion, establish a cluster between devices, decide whether performing the detection should be locally on the device or offloaded to surrogate devices, and handle communication and data exchange between the active devices. We were able via our approach to reduce the energy consumption on the device by 61% and increase the response time of the detection around 40%. In this context, the main contributions of this paper are separated into three folds:

- Elaborating an ad hoc mobile cloud based security as a service approach that uses Wi-Fi Direct as a mean of communication. To the best of our knowledge, none of the current approaches provide security as a service over mobile ad hoc cloud.
- Offloading to reduce CPU and power consumption on the device and augment the performance of intrusion detection systems by reducing its response time.
- Allowing real-time intrusion detection in the absence of an internet connection without levying any additional charges due to data consumption and/or subscribing to servers.

The rest of the paper is organized as follows. Related work is summarized in Section 2. Section 3 introduces the proposed framework. Experiments are presented in section 4. Finally, we conclude the paper in section 5 and point to the future work.

II. Related Work

In this section we review the literature behind cloud based intrusion detection systems. Several approaches have been addressed in this context to improve the performance of IDS on mobile devices. We chose to divide these approaches into 2 categories: cloud based and intrusion detection in MANETs. We also show the scope of Wi-Fi Direct applications since its a core factor of our work.

A. Cloud Based Intrusion Detection

[15] proposed a mechanism by which a device is replicated in the cloud and examined for malicious behaviors. Their mechanism allows multiple detection engines to run in parallel in order to gain a better detection rate by running multiple emulators of the same virtualized device. In case of failures, the mobile agent is responsible for putting the smart-device in a recovery mode synchronized with the cloud. The cloud service is responsible for the analysis and detection of suspicious behaviors.

[14] proposed an architecture composed of two components, cloud service and host agent, to gain an improved detection rate by moving the detection to the cloud. More resources can be allocated to the detection engines running in parallel to provide mobile devices with high positive detection rates. By moving the analysis to the cloud, the power and CPU consumption on the device itself are reduced. Their approach reduces the complexity held by the device itself due to failures resulting from compatibility issues on multiple platforms.

[17] proposed an architecture for moving the IDS computation to remote servers that contain exact copies of the devices. Their technique allows for multiple detection algorithms to run simultaneously on replicated devices in the virtual environment. A minimal set of traces allowing solid replication of the device is sent by the mobile agent to the remote server via recording and replaying framework.

[2] proposed a framework that caters a real-time and strong off-device protection by continuously synchronizing the device with an emulated device on the cloud. A client agent continuously passes device's inputs and communication data to be examined on the cloud and takes actions against threats. The other component is the cloud itself that hosts emulated devices, receives events from different interfaces of the device, and ensures a periodic backup of the emulate device.

[18] introduces a hybrid IDS for smart-devices that automatically decides whether the detection should take place locally on the device or be offloaded to the cloud for analysis. However continuous communication between the mobile agent and the cloud is required. Diverse detection engines are integrated in the cloud to gain protection against a larger set of malwares. The proposed framework consists of several components most of which run on the device itself. In addition, it decides on the level, security and type of detection algorithms to be enforced on the device.

However, to the best of our knowledge, none of the proposed approaches consider using ad hoc cloud. Their approaches impose extra power consumption when used with mobile data instead of Wi-Fi to send traces, input events, and data. In addition, mirroring the traffic and data to the cloud via mobile data levies extra charges on the user as it consumes his mobile data quota. It also requires the usage of remote servers, that are not free, which inflicts additional costs. Moreover, such approaches suffer from the intrinsic latency drawback present

in current Wi-Fi technologies especially in long distance communications [19].

[3] presented a hybrid approach consisting of a new elastic computing platform for smart-device. It is based on combining both ad hoc virtual cloud and infrastructure based cloud to achieve higher scalability. An ad hoc virtual cloud composed of smart-devices in close proximity connected via wireless radio such as Bluetooth work cooperatively to accomplish offloading tasks. An infrastructure cloud composed of cloud phones performs computing intensive tasks. Nonetheless, their paper focuses more on the server-based cloud aspect without illustrating or providing information pertaining to the offloading and cooperation aspects of their ad hoc virtual cloud. The authors plainly state that an ad hoc virtual cloud can be used for offloading without portraying further details. The paper also doesn't exhibit any experiments investigating the performance and impact of cooperative execution using the cluster of smart-devices.

B. Intrusion Detection in MANETs

Furthermore, literature review also shows a different approach to guarantee protection in MANETs. [20] proposed an FSM IDS that detects attacks on the Dynamic Source Routing (DSR) protocol in mobile ad hoc clouds. Their distributed intrusion detection system works by periodically selecting a monitor node to detect suspicious or misbehaving nodes based on their incorrect behaviors. The authors of [21] proposed a zone based IDS to prevent routing attacks (impersonating and black/gray attacks). Every node in each different zone has its own local IDS. Nodes communicate together to exchange intelligence about intruders. This corporation allows nodes to have a better detection rate and raises more specific alarms. Such approaches differ from our work as they tackle intrusion detection systems focused on routing protocols. They allow the detection of malicious nodes by analyzing their behaviors and actions. In our work, we focus on intrusion detection systems monitoring the system activities of a node.

C. Wi-Fi Direct Approach

SuperBeam is an application that uses Wi-Fi Direct to enable fast, easy, and reliable file transfer between two devices[7]. BombSqaud is another application that uses Wi-Fi Direct to allow users to create multiplier games anywhere and anytime[22]. Spaceteam, an action multiplayer game, allows up to 8 players to join at the same time using Wi-Fi Direct [23]. FireChat, a social media application aimed mainly towards sending messages and photos, is built on Wi-Fi Direct and Bluetooth to provide its features. Furthermore, The authors of [5] suggest using Wi-Fi Direct as means of real-time communication and data exchange between medical applications in order to mitigate conflicts that rise from wireless interference of access points with Wi-Fi systems. They perform experiments in the medical field showing that Wi-Fi Direct results in a higher data rate by 65% on average. [6] proposes a new framework using Wi-Fi Direct for exchanging data between users of close proximity in social networks. Their approach circumvents the growing traffic of mobile data and avoids overloading the network infrastructure and ISP's.

Indeed such approaches take advantage of Wi-Fi Direct's high data rate due to less wireless interference, fast and easy setup, and availability on new devices, but none proposes or suggests using Wi-Fi Direct as means of communication for intrusion detection systems. Their scope is limited to multiplayer games, file sharing applications, advertising, social networks, and real time data transfer applications.

III. Approach Overview and Architecture

In this section of the paper, we provide the overall architecture of the framework. Our ad hoc cloud cluster is built using Wi-Fi Direct. Devices within a given range connect together to form an ad hoc cloud where data can be exchanged in a secure way. Communication between nodes is limited to data needed for constructing and analyzing traces on devices. Each node runs a local intrusion detection engine. Additionally, nodes can run different instances of the IDS based on altered engines. Each engine will then perform its scan on the same piece of data resulting in further and more powerful detection of malevolent data. A node is selected as the group owner (GO) and all other volunteers within a cluster connect to it. By doing so, volunteers in the cloud share their resources to reduce the computational load on one node. This is very helpful and applicable especially in scenarios where the requester device is running on low resources such as battery, CPU, memory, etc;

As illustrated in Figure 1, our approach consists of a mobile agent that is responsible for discovering nearby devices through Wi-Fi Direct and establishing a connection between them. A Group Owner (GO) assigned to this cluster, is responsible for exchanging data between the different slaves. Moreover, the GO is in charge of handling the distribution mechanism, the transferring of data, and receiving the end result. In order to perform these tasks in a seamless and effective way, a service running on the Master Node manages the process. In addition, a service running on all nodes in the cluster coordinate with the Master Node's service using a set of predefined protocols. The Master Node, receives the data to be scattered on the different nodes for analysis after being authenticated. An algorithm divides and sends the data equally among available nodes in parallel using threads. The service waits for all nodes to return the result of analysis and transfers it back to the node that requested the analysis. Analysis is done through the default service running on all nodes. Deep analysis of the data is performed via the crafted detection engine after which the final result is directed to the Master Node which waits for all results from different nodes. Finally the Master Node sends the analyzed results to the Requesting Node indicating each data chunk that was remotely examined. The node can now act upon the data by removing the detected threat. In order to generate data required by the detection engine, *strace* diagnosis utility can be used to intercept and record the system calls resulting from the interactions of applications with the Linux kernel. The

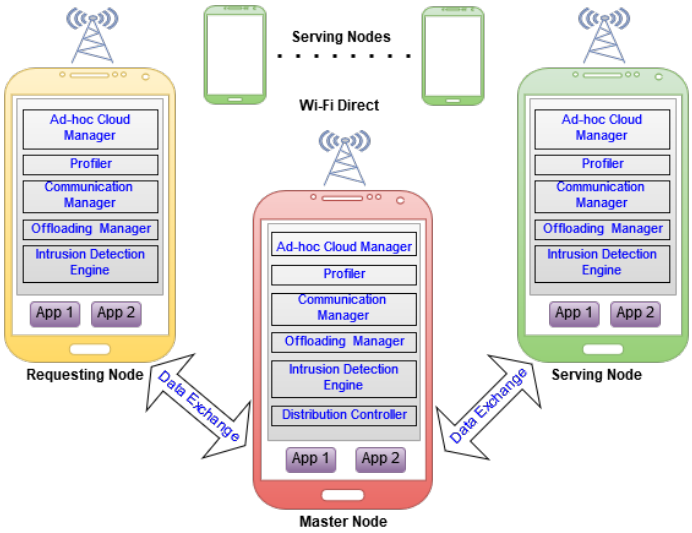

Fig. 1. High Framework Architecture

execution traces are generated and saved in a log to be used by the detection engine. Figure 1 shows the high level system architecture of the proposed framework. Below, we list the different components composing the tackled framework which consists of three main elements: *Requesting Node, Master Node, and serving node*.

- Requesting Node is any node, be it an element of a mobile ad hoc Cloud or a standalone node, that wishes to perform a scan to identify malicious behavior. It decides whether the detection process should be performed locally on the node itself or offloaded to a mobile ad hoc cloud. It consists of the following sub-components: *Ad-Hoc Cloud Manager, Profiler, Communication Manager, Offloading Manager, and detection algorithm*.
- Master Node is a node that is assigned to be the Group Owner in a specific Wi-Fi Direct Group and therefore is responsible for managing the mobile ad hoc Cloud and requests by peers. It manages all connected peers in the cluster and decides on the nodes that are most suitable to handle the offload request. Also, it is responsible for communicating data in the mobile ad hoc cloud between the nodes. It consists of all the sub-components consisting of a normal node in addition to the following sub-component: *Distribution Controller*.
- Serving Node is a node in the mobile ad hoc cloud that the Master Node offloads work to. In other words, it is a surrogate node that performs tasks, in accordance with the Master Node, on behalf of a Requesting Node. It will perform the task required from and return the result back to the Master Node. It consists of the following sub-components: *Ad-Hoc Cloud Manager, Profiler, Communication Manager, Offloading Manager, and detection algorithm*.

Each of these three components is divided into several sub components that allow a seamless and offloading mechanism, communication system, and distributed intrusion detection process in an ad hoc mobile cloud using Wi-Fi Direct. The aforementioned interactions between these sub-components of the framework are depicted in Figure 2.

A. Ad-Hoc Mobile Cloud Manager

The Cloud Manager acts in 2 different perspectives depending on the type of the node. On a Master node, it is responsible for creating and monitoring the mobile ad hoc cloud within a certain range. It accepts connections to join and dynamically keeps track of the number of connected nodes and their status. Furthermore, it accepts a profile of each device once it connects and keeps an updated table of the profiles to be used later in offloading decisions. It also differentiates between the different types of nodes and assigns specific roles to each. In case of a Requester or Serving Node, the Cloud Manager searches for a Group Owner (GO) and requests to join its cluster. After this discovery phase, it maintains the connection with the cluster it joined in order to send and receive data from the Master node. Once the connection is established, the Cloud Manager sends the profile of the device to the GO. Finally, it is liable for reconnecting or disconnecting in failure scenarios. In case of the absence of a formed cluster, the Requester node acts as a GO to create a cluster for offloading.

B. Profiler

The profiler is developed as a service that runs in its own process and uses native code to access the ”virtual file system” and captures run-time system information. It is responsible for collecting resources consumption and availability from the Kernel’s information center (*/proc* file system). It collects information related to the device’s resources such as CPU usage per process, memory used by each process, and battery level. It also determines the amount of free memory on the device, the name of the device, and differentiates between system and user processes. The data collected by the profiler is updated at a regular basis (periodically) and can be directly requested anytime by other modules in the framework (on-demand). The output of this profiler is a well structured file for easy and fast access containing all gathered information. This module is aimed towards the future work to support intelligent offloading.

C. Offloading Manager

The Offloading Manager is used to allow a node to decide whether it needs to offload the detection or execute it locally. In case of local execution, the Offloading Manager doesn’t communicate with the Master Node. However, in case of an offloading decision, this manager performs 2 main functions. First, it gathers the collected execution traces. Second, it reaches out to the Communication Manager for further handling. The decision is a smart decision based on several factors such as the battery level of the device, number of available nodes in the cluster, and other profiling criteria. This module is allowed to directly communicate with the *Profiler* component to collect updated profiling criteria. We used an ”always offload strategy” in the scope of this paper.

D. Communication Manager

The Communication Manager is a data socket layer developed as a service to allow and facilitate the exchange of data between nodes. This layer specifies a set of protocols used to define and abstract the communication according to each node's requirement. It arbitrates two communication mechanisms, P2P and client-server, based on the type of the node and the operation to be performed by several modules of our framework. It is also responsible for sending the collected traces from the Offloading Manager module of the Requester node to the Distribution Controller of the Master node, and then back to the Requester node.

E. Intrusion Detection Engine

The Detection Engine is the core component for detecting any intrusion. It is in control of identifying any signatures that are matched with a predefined malicious data-set. Several algorithms can be used for developing such engines such as, *Aho-Corasick Algorithm, Low Memory Keyword Trie, Wu-Manber* offered by *Snort* [24]. Genetic Algorithms and Data mining are used in building such engines as well [25] .In our work, we implemented a JAVA detection engine built on top of the Aho-Corasick Algorithm to allow matching of real time gathered system calls against a data set. It constructs a string pattern finite state machine (Trie) from a malicious signature database. It adds all required transitions and failure links between internal nodes of the automaton to allow fast and efficient traversals and searching of elements. Since in our case, we are using a known database (predefined data set), the complexity of the algorithm tends to be linear in terms of input length and number of harmonized patterns. Each node, can use the constructed FSM to scan patterns in a background thread. The Detection Engine works with two types of data depending on the Offloading Manager decision. In case of a local detection, the engines scans data of the same device and acts accordingly (raise alerts, block execution etc...). In the second scenario, an offloading decision is set and the Engine scans foreign data (data of a Requester Node) that is fed to it by the Master Node. The Engine then communicates the result of the detection back to the Master Node which in turn acts upon it.

F. Distribution Controller

The Distribution Controller is a vital sub-component running on the Master Node only. It is developed as a service and pledged for handling the data to be distributed on other nodes and gathering the results. It receives the data sent by the Requester node through the Offloading Manager using the Communication Manager and prepares it for distribution. The service runs in the background and spans multiple threads (one thread per available node) dedicated to nodes. It divides the data into equal segments upon available nodes in the cluster. In some cases, where equal segmentation of the data is impossible, the last segment tends to be relatively larger than the rest. Each thread associated with a node is responsible for transmitting the data, in accordance with the Communication

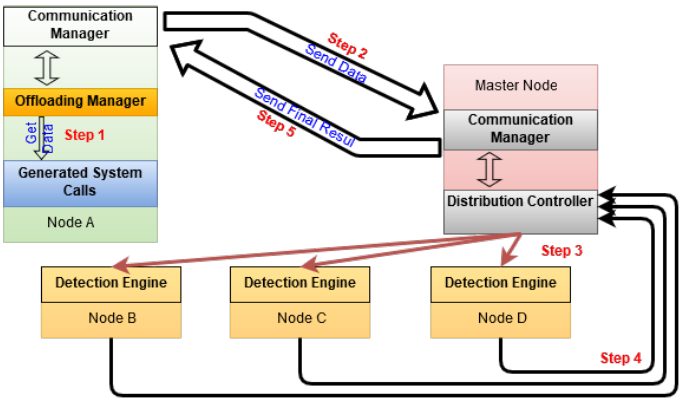

Fig. 2. Framework Component's Interactions

Manager, in a guaranteed and reliable way to the node. The threads are also accountable for accepting the result of their associated nodes. Therefore, the distribution process is executed in a parallel manner achieving an efficient mechanism. In addition, the Distribution Service accepts data from spanned threads composed of the duration of the scanning process, number of malicious items, and each individual malicious item. After accepting results from all threads, the service communicates the final result back to the requester node (the node that requested the offloading process) through the Communication Manager. In case of a failure in any thread or node, the requester is notified to cover up. This module will be transformed into an intelligent distributor based on several factors including, but not limited to, the maximum load a device can handle at the moment. The intelligent strategy is currently under development.

IV. Experiments and Performance Analysis

A. Testbed Setup

In this section, we investigate the performance of our proposed model pertaining to execution time, energy consumption, CPU and RAM usage. In addition, we provide a comparison of our scheme with current server-based approaches. The experiments are divided into 3 scenarios. All three include running the intrusion analysis task in order to evaluate the behavior of our model. The first scenario consists of comparing our ad hoc model to a local execution in order to show the advantages of our model. The second scenario consists of examining our ad hoc model approach to server-based ones using Wi-Fi (IEEE 802.11n wireless networking standard) for communicating with the server. The third scenario investigates the performance analysis of our model compared to server-based models using mobile data (3G) as means of communication. Both second and third schemes, are conducted in order to illustrate that our ad hoc cloud based approach outputs very close and competitive results to server-based ones in terms of the aforementioned metrics. Moreover, they are subdivided into 2 cases that replicate the behavior of the mobile ad hoc

approach. In other words, if the ad hoc approach is offloading half of the task to another device and executing the other half locally, we correlate this to executing half of the task on the server and the other locally in the server-based approach. We run the same analysis code written in *Java* using our proposed ad hoc model and the server-based model. The remote server side is running Ubuntu 14.04 with 2 GB of memory and a 2.4Ghz dual core Intel(R) Xeon(R) E5-2630L v2.

We used three devices running Android Operating System in our experiments. Two of which are Samsung Galaxy S5 with the following specifications: Quad Core 2.5 GHz CPU, 2 GB RAM, and 16 GB storage. The third device is a LG Nexus 5 with the following specifications: Qualcomm Snapdragon 800 2.27 Ghz, 2 GB RAM, Adreno(TM) 330 @ 450 MHz, and 32 GB storage. The Nexus device is randomly selected as the *Requester Node* deciding to offload and the Samsung devices are used as participating nodes in the cluster (one of which is the *Master Node*). We established the ad hoc cloud by using Wi-Fi P2P on Android that uses 802.11 networking standards. In our experiments, we focus on 4 aspects of the device: performance, energy, CPU, and RAM. Performance is measured in terms of the response time of the IDS i.e. time needed execution of the task. Energy is measured as the total energy consumed by CPU, Wi-Fi, and 3G. CPU and memory are analyzed in two ways, the CPU and RAM load resulting from the IDS and the percentage of the CPU and RAM usage while an IDS detection is being performed on the device.

The dataset consists of a set of malicious executions signatures generated from the study of 90 malicious applications downloaded from (blog Contagio: Mobile malware) (http://contagiominidump.blogspot.ca/). Based on various executions of each of these malwares, syscall traces have been generated and extracted using *strace* Linux tool. Depending on the studied malwares, the length of the generated traces varies from 10000 system calls to 60000 system calls. For each trace, a set of sub-traces is extracted where each of which is related to an execution thread. Applying the techniques introduced in [26] on the sub-traces of malware belonging to same family, we generated execution signatures characterizing the malicious behavior of that family.

B. Analysis and Results

Figure 3 shows the energy variation between the different test cases. Energy consumed by the ad hoc IDS is less than the energy consumed by the IDS running on locally on the device. In case of small data sets, the difference is not much however, once the data grows in size, the advantage of ad hoc IDS becomes clearer and more desirable. For example, an IDS scanning a 10 MB file consumed 212.5 J on the device while it only consumed 110 J using the proposed ad hoc IDS with only 2 nodes. In this case, the ad hoc IDS saved 48.2% of the consumed energy. Using 3 devices, the energy is reduced by 65.5% on a 10 MB file. Comparing a cloud based approach using Wi-Fi (moving the execution of the IDS to a remote server) to our ad hoc approach shows very similar results. Energy consumed by the ad hoc IDS in a cluster of 3 nodes is 73.2 J which is only 13% more than the energy consumed by the Cloud based approach (63.2 J). However, using 3G as the internet connection in the cloud based approach resulted in higher energy consumption. Energy consumed by the cloud based approach using 3G on a 10 MB file is 169 J which is 62.6% higher than the energy consumed using Wi-Fi.

Figure 4 shows the IDSs respond time of each approach. The respond time of the Adhoc approach on a 100 KB file took almost the same time compared to running the IDS locally on the device. An IDS scanning a 1 MB file took 29.8 seconds while it took 24.45 and 15.2 seconds using our proposed mechanism with 2 and 3 nodes respectively. Comparing the results of the 10 MB file, it is clear that our architecture has a better performance than having the IDS to locally run on the device. It took 459 seconds for the IDS to finish its task on the device while it took 264.5 and 177 seconds using our ad hoc approach with 2 and 3 nodes respectively. These results show that our framework reduced the respond time by 42.3 and 61.4% respectively. Comparing our ad hoc cloud IDS to a server-based cloud approach shows that our approach increases the response time by only 10.7% on a 10 MB data. However when using 3G, the cloud based approach introduces extra performance overhead by almost 47%. Taking the 10 MB data, the ad hoc cloud took 177 seconds to respond compared to 334 seconds using a server-based cloud approach.

Figure 5 shows the CPU evaluation for the different test cases. The figure focuses on the duration of which the CPU is assigned for the IDS task. Since the same IDS runs on the different nodes, the CPU load (the CPU percentage) ranges between 37% and 47% which is only a 10% change. Our approach using 3 nodes reduced the CPU time held by the IDS by 49.6% compared to running the IDS locally on the device of a 1 MB data chunk. On a 10 MB data, we benefit more by reducing the CPU time by almost 78.9%. Comparing the previous result with moving the IDS to a remote server using Wi-Fi, our ad hoc architecture takes an additional 2.4% (11.4 seconds) overhead. Using the ad hoc cloud with 3 nodes, we reserve the CPU for 97.2 seconds compared to 85.86 seconds using a remote server. On small-size chunks (100KB and 1MB), the IDS holds the CPU for almost the same time between the different presented approaches. However, when using 3G to gain access to internet, the results vary. A 10 MB chunk analyzed via our ad hoc cloud using 3 nodes requires 97.2 seconds of CPU time compared to 217 seconds using 3G through a remote server. This imposes an extra 26% overhead compared to our approach.

Figure 6 shows the memory consumption of the IDS using different test cases. We emphasize 2 main factors affecting the device, the memory reserved by the IDS in MB and the time until the reserved memory is released. With 100 KB and 1 MB segments, memory reserved by our ad hoc based IDS is very close to the one reserved when running the IDS locally. On a 10 MB data, a 20% decrease in memory is noticed when using our ad hoc approach. Server-based Cloud approach requires less memory (40 MB) compared to other approaches. On the other hand, the duration of the memory

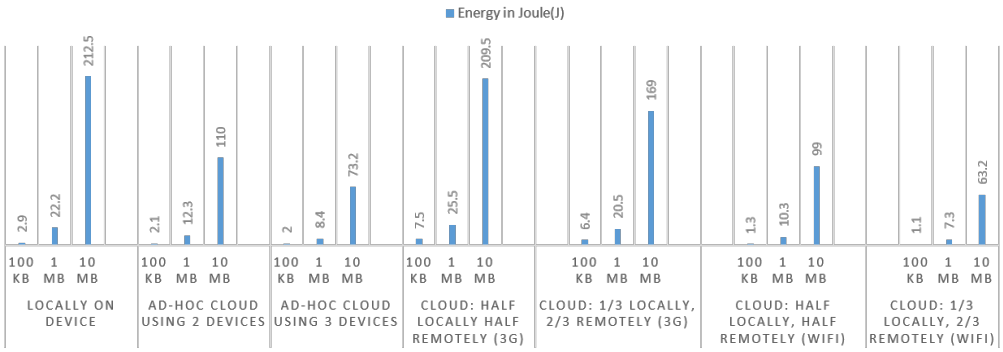

Fig. 3. Energy Consumption

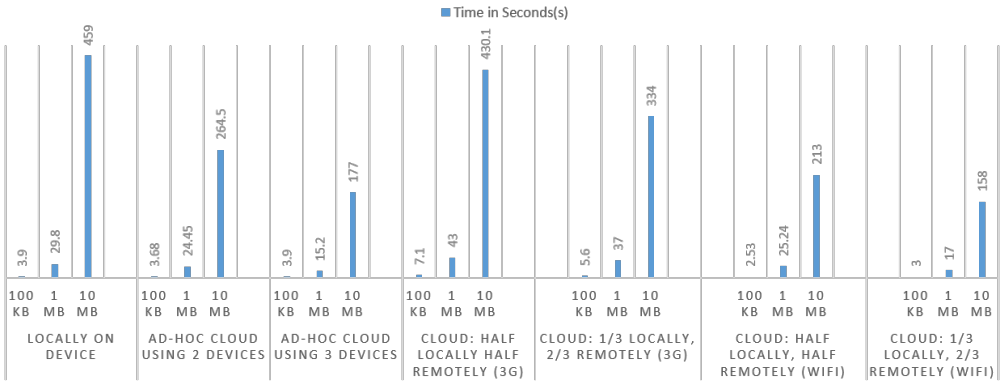

Fig. 4. Performance Evaluation

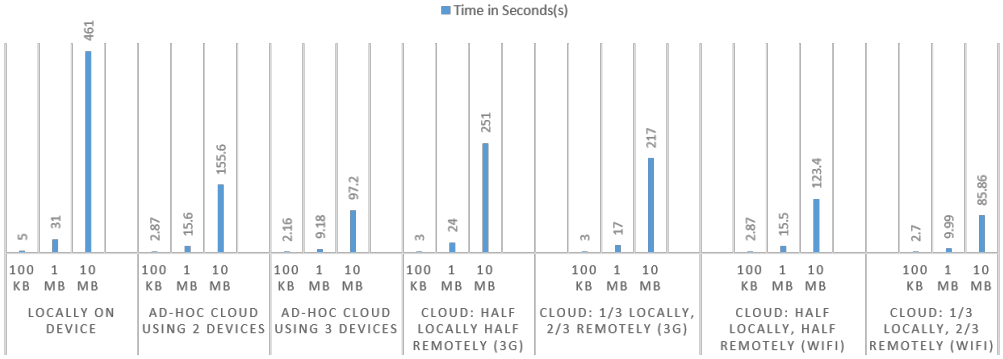

Fig. 5. CPU Evaluation

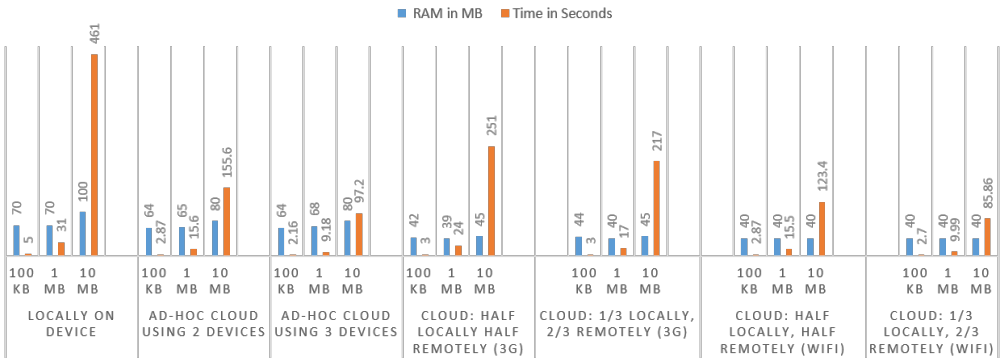

Fig. 6. Memory Evaluation

reservation is consistent with the CPU time since the memory is released once the IDS task finishes executing on the device. Shortly stated, our approach reduces the memory reservation duration compared to server-based cloud approaches using 3G and extends the time by 11% on average compared to using Wi-Fi connection to the remote server.

V. Conclusion and Future Work

This paper addressed problems related to the performance of running an intrusion detection system (IDS) locally on smart devices. We introduced an approach by which nodes in an ad hoc mobile cloud share their resources to minimize the load of detection on one device and to speed up the response time of the IDS. Particularly, we created an effective cooperative medium between nodes for offloading tasks and services to surrogate nodes in an ad hoc mobile cloud. In this context, the paper showed promising results. It reduced the response time by 40%, decreased the memory and CPU usage duration (time spent by the IDS reserving these resources), and lessened the energy consumption by 50% compared to local execution. We showed that our architecture's performance is similar to server based ones. We also gained up to 55% increase when using 3G for communication with the server. As such, the device benefits from remote resources while keeping its resources' consumption minimal. Furthermore, the approach imposes a major enhancement by permitting its usage in situations where internet connection, be it Wi-Fi or mobile data, is unavailable. Finally, using ad hoc mobile cloud eradicates any additional financial charges in contrast to using mobile data (3G, 4G etc...) for server offloading.

For future work, the framework can be extended in several ways to enhance and improve the decision and distribution process. Dynamic assignment of the Master Node can improve the clustering and creation of the ad hoc cloud. Also, smart offloading and profiling can advance the distribution process and determine the nodes that best fits for offloading in real time.

VI. Acknowledgment

This work was supported by the Associated Research Unit of the National Council for Scientific Research CNRS-Lebanon, Lebanese American University (LAU), Khalifa University of Science, Technology & Research (KUSTAR) and Defense Research and Development Canada (DRDC Valcartier).

References

[1] Q. Hassan, "Demystifying cloud computing," *The Journal of Defense Software Engineering,(CrossTalk)*, pp. 16–21, Jan/Feb 2011.

[2] S. Zonouz, A. Houmansadr, R. Berthier, N. Borisov, and W. H. Sanders, "Secloud: A cloud-based comprehensive and lightweight security solution for smartphones," *Computers and Security*, vol. 37, pp. 215–227, September 2013.

[3] W. Zhang, Y. Wen, J. Wu, and H. Li, "Toward a unified elastic computing platform for smartphones with cloud support," *Network, IEEE*, vol. 27, no. 5, pp. 34–40, 2013.

[4] A. Elhabbash, R. Bahsoon, P. Tino, and P. R. Lewis, "Self-adaptive volunteered services composition through stimulus-and time-awareness," in *Web Services (ICWS), 2015 IEEE International Conference on*. IEEE, 2015, pp. 57–64.

[5] C. Jin, J.-W. Choi, W.-S. Kang, and S. Yun, "Wi-fi direct data transmission for wireless medical devices," in *Consumer Electronics (ISCE 2014), The 18th IEEE International Symposium on*. IEEE, 2014, pp. 1–2.

[6] Y. Wang, A. V. Vasilakos, Q. Jin, and J. Ma, "A wi-fi direct based p2p application prototype for mobile social networking in proximity (msnp)," in *Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference on*. IEEE, 2014, pp. 283–288.

[7] LiveQoS, "Superbeam," https://play.google.com/store/apps/details?id=com.majedev.superbeam&hl=en/.

[8] R. Mizouni and M. El Barachi, "Mobile phone sensing as a service: Business model and use cases," in *2013 Seventh International Conference on Next Generation Mobile Apps, Services and Technologies*. IEEE, 2013, pp. 116–121.

[9] R. Mizouni, A. Salah, S. Kolahi, and R. Dssouli, "Merging partial system behaviours: composition of use-case automata," *IET software*, vol. 1, no. 4, pp. 143–160, 2007.

[10] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. S. Gaur, M. Conti, and M. Rajarajan, "Android security: A survey of issues, malware penetration, and defenses," *Communications Surveys & Tutorials, IEEE*, vol. 17, pp. 998 – 1022, May 2015.

[11] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, May 2012, pp. 95 – 109.

[12] V. Rastogi, Y. Chen, and X. Jiang, "Catch me if you can: Evaluating android anti-malware against transformation attacks," *Information Forensics and Security, IEEE Transactions on*, vol. 9, pp. 99 – 108, December 2013.

[13] M. Jakobsson, "Why mobile security is not like traditional security," 2011.

[14] J. Oberheide, K. Veeraraghavan, E. Cooke, J. Flinn, and F. Jahanian, "Virtualized in-cloud security services for mobile devices," in *MobiVirt '08 Proceedings of the First Workshop on Virtualization in Mobile Computing*. ACM, NY, USA, June 2008, pp. 31–35.

[15] R. S. Khune and J. Thangakumar, "A cloud-based intrusion detection system for android smartphones," in *Radar, Communication and Computing (ICRCC), 2012 International Conference on*. IEEE, December 2012, pp. 180–184.

[16] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, pp. 1–11, 2011.

[17] G. Portokalidis, P. Homburg, K. Anagnostakis, and H. Bos, "Paranoid android: versatile protection for smartphones," in *ACSAC '10 Proceedings of the 26th Annual Computer Security Applications Conference*. ACM, NY, USA, December 2010, pp. 347–356.

[18] D. Damopoulos, G. Kambourakis, and G. Portokalidis, "The best of both worlds: a framework for the synergistic operation of host and cloud anomaly-based ids for smartphones," in *EuroSec '14: Proceedings of the Seventh European Workshop on System Security*, no. 6. ACM, NY, USA, April 2014.

[19] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, 2009.

[20] P. Yi, Y. Jiang, Y. Zhong, and S. Zhang, "Distributed intrusion detection for mobile ad hoc networks," in *SAINT-W '05 Proceedings of the 2005 Symposium on Applications and the Internet Workshops*. IEEE Computer Society Washington, DC, USA, January 2005, pp. 94–97.

[21] N. Soms, R. Priya, A. Banu, and P.Malathi, "A comprehensive performance analysis of zone based intrusion detection system in mobile ad hoc networks," in *Signal Processing, Communication and Networking (ICSCN), 2015 3rd International Conference on*. IEEE, March 2015, pp. 1–8.

[22] E. Froemling, "Bombsquad," https://play.google.com/store/apps/details?id=net.froemling.bombsquad&hl=en.

[23] H. Smith, "Spaceteam," https://play.google.com/store/apps/details?id=com.sleepingbeastgames.spaceteam&hl=en.

[24] M. Norton and D. Roelker, "Snort 2.0: Hi-performance multi-rule inspection engine," *Sourcefire Network Security Inc*, 2002.

[25] V. K. Kshirsagar, S. M. Tidke, and S. Vishnu, "Intrusion detection system using genetic algorithm and data mining: An overview," *International Journal of Computer Science and Informatics ISSN (PRINT)*, vol. 2231, p. 5292, 2012.

[26] Y.-D. Lin, Y.-C. Lai, C.-H. Chen, and H.-C. Tsai, "Identifying android malicious repackaged applications by thread-grained system call sequences," *computers & security*, vol. 39, pp. 340–350, 2013.