EyeSim: A Mobile Application for Visual-Assisted Wormhole Attack Detection in IoT-enabled WSNs

Niki Tsitsiroudi
SORCE LTD
The Pavilion, Newbury Business Park,
London Rd, Newbury RG14 2PZ
niki.tsitsiroudi@sorce.co.uk

Panagiotis Sarigiannidis
Dept. of Informatics
and Telecommunications Engineering,
University of Western Macedonia,
Karamanli & Ligeris Street,
50100, Kozani, Greece
psarigiannidis@uowm.gr

Eirini Karapistoli
& Anastasios A. Economides
IPPS in Information Systems
University of Macedonia
Thessaloniki, 54006 Greece
{karapis, economid}@uom.gr

***Abstract*—Internet of Things (IoT) have emerged as a valuable, flexible, and interoperable network of devices, objects, items, and electronics. Fuelled by recent advances in networking, communications, computation, software, and hardware technologies, IoT has stepped out of its infancy and is considered as the next breakthrough technology in transforming the Internet into a fully integrated Future Internet. Wireless Sensor Networks (WSNs) are utilized by IoT to collect, exchange, and deliver data remotely leveraging the potential of IoT in practical applications and services. However, delivering data remotely might be threatened by various and serious security attacks. This work focuses on developing a visual-assisted tool for exposing security threats in IP-enabled WSNs. The proposed tool, called EyeSim, is a human-interactive visual-based anomaly detection system that is capable of monitoring and promptly alerting for the presence of wormhole links. In addition, it is capable of indicating the malicious nodes that form the wormhole link. EyeSim may expose adversaries by conducting cognitive network data analysis based on dynamic routing information. The efficacy of EyeSim is assessed in terms of detection accuracy. The simulation results show that EyeSim has the capabilities to accurately detect multiple wormhole attacks in real-time.**

***Index Terms*—IoT-enabled Wireless Sensor Networks, wormholes, visual-assisted wormhole detection**

I. Introduction

Without doubt, IP-enabled Wireless Sensor Networks (WSNs) emerge as a promising framework, which brings into play many interesting features of IoT [1]. A WSN can be established everywhere due to the requirements that a sensor has because they are not the same with a wired network. An example of sensor network implementation is to collect data for the temperature in a tank with fuel and send them to service tasks' execution. A WSN is an improved wired network, but there are differences between them, such as (i) in a WSN there is no limit for the number of nodes that comprise the network, (ii) the topology of a WSN can change due to the mobility of its nodes or by adding or removing a node from it, (iii) a sensor node have power, computation and memory limits, (iv) a sensor node is prone to failure that may happen from the environment where the network is established. These differences between a wired and a wireless network affect the secure data transmission in a WSN. Indeed, the security threats are a major deterrent in many applications IP-enabled WSNs are envisaged to support [2]. For example, every WSN uses a radio channel for the packet transfer, which makes the network affected to denial-of-service attack. The energy and processing power limit of a node, deter the use from a public key cryptography.

There are several ways to represent the security events that occur inside a network. One of them is visualisation. Visualisation is the process of generating images using data analysis tools [3]. According to Colin Ware [4], a visual representation is better than a text "the human visual system is a pattern seeker of enormous power and subtlety. This is because, the eye and the visual cortex of the brain form a massively parallel processor that provides the highest-bandwidth channel into human cognitive centers." A picture carries a large amount of information, different colors, shapes, and sizes for every set of data, which is represented to the viewer in a single picture.

Indeed, information visualisation has been deployed in different fields and recently in visualizing network data [5]. Security visualisation is arguably one of the first directions to take when it comes to understanding, analyzing and finding suspicious network activity in vast amounts of data. To aid the network analysts in this task, researchers have proposed numerous visualisation techniques (i.e., scatter plots, color maps or some form of a graph, etc.) [6]. In addition, these tools contain various visualisation capabilities in order to make it easy to locate incidents of interest and to monitor the network health. While several visualisation tools have been developed to simulate WSN security issues, none of them is able to run on mobile devices. The development of a visualisation system for WSN security based on mobile technologies is a necessity due to the fact that this technology is used everywhere.

Accordingly, the main contribution of this paper is the development of an application that can simulate and visualize threats and attacks, which occur in an IP-enabled sensor network. The proposed application, called EyeSim, is a prototypical security visualisation system for mobile devices whose software is Android. EyeSim is capable of monitoring and promptly alerting for the presence of wormholes indicating at the same time the malicious nodes that form the wormhole link. In order to expose adversaries, EyeSim conducts cognitive network data analysis based on dynamic routing information coming from network logs.

The remainder of the paper is organized as follows. Section II reviews existing security visualisation approaches aimed at detecting attacks launched against IP-enabled WSNs. Section III outlines the network assumptions made in this work. Section IV introduces the EyeSim mobile app and its architecture. In Section V, the EyeSim system is evaluated through a simulated attack scenario. Finally, Section VI concludes the paper and discusses future extensions.

II. Related Work

IoT security has received a lot of attention recently [7]. However, the research work that has been published in the literature is limited. For example, the work [8] is a strong indication that the area of security visualisation for IP-enabled WSNs is still at an infancy stage. In contrast, substantial research has already been conducted in the area of visualisation for computer security [4], [9]). Several security visualisation solutions have also been proposed for 802.11-like networks [10]. Many of these approaches have been proven to be effective at allowing users to discover malicious activities such as worms, DoS attacks as well as probing attacks. Though powerful the aforementioned solutions are, they are not directly applicable to WSNs because several of the characteristics these networks possess, impose a re-examination of the security visualisation problem. Compared to the previous security visualisation methods, within EyeSim, we apply as well as develop novel visualisation algorithms for detecting wormholes in one single view.

In the context of WSNs, several other network visualisation tools have been proposed to graphically monitor real-world or simulated sensor network deployments [11] and display live information about the network topology and the collected sensory data in order to enable live debugging of the deployed sensor network.

Early in 2004, Wang and Bhargava [12] proposed a security enhancing visualisation mechanism for WSNs, called MDS-VOW, which is capable of identifying the occurrence of a wormhole attack in stationary wireless sensor networks. Wang and Lu [13] extended the MDS-VOW concept proposing an improved detection mechanism, called interactive visualisation of wormholes (IVoW). IVoW mechanism efficiently integrates automated intrusion detection algorithms with visual representation and user interaction to support visualisation of several wormholes in large-scale dynamic WSNs. Simulation results showed that IVoW accelerates the detection process and improves the algorithm accuracy when compared to the MDS-VOW approach.

Lu *et al.* [14] developed an integrated approach to detect Sybil attacks in mobile WSNs through visualizing and analyzing multiple reordered topology patterns. Automated reordering and evaluation algorithms used here reveal the malicious nodes in the network topology faster and more accurately. The proposed approach also provides a time-series analysis in order to identify attack durations. This analysis is based on time histograms and an automated time segmentation method. Overall, this approach was evaluated through a series of real-life attack scenarios, and has shown success at unveiling unknown Sybil attacks.

Abuaitah et al. [15], developed a security visualisation system, called SecVizer, capable of parsing any QualNet generated traffic trace from both wired and wireless networks. SecVizer combines topology visualisation with the parallel coordinate plot technique in order to obtain a faster and more effective detection of network vulnerabilities. By exploring noticeable traffic patterns at both the network topology window and the parallel plot window, the tool has demonstrated its ability to detect various malicious network activities, most notably, Distributed Denial of Service (DDoS) attacks. The tool, in its current status, is intuitive enough to allow an analyst to process network events in real-time, but further drill down depth is needed to come up with a firm final resolution.

Recently, Shi *et al.* [16] proposed the Sensor Anomaly visualisation Engine (SAVE) system; a representative approach to sensor network anomaly detection and fault diagnosis through visualisation. The SAVE system encompasses three distinct visualisation components that interpret the topological, correlational and dimensional sensor data dynamics and their anomalies. SAVE was validated through a case study deployment on the real-world large-scale WSN system called GreenOrbs.

Unlike previous approaches, EyeSim provides a single, effective, and clear-cut visualisation environment that exposes multiple wormhole links. EyeSim is portable to a smart phone and is characterized by a pervasive look that offers the security analyst a monitoring tool to directly analyse and detect ongoing threats experienced by the IoT under investigation.

III. Network Assumptions

A. Network Model

EyeSim monitors an IP-enabled WSN comprised of N sensor nodes in a deployment area of E metric units. All nodes are IEEE 802.15.4-compliant and they are allowed to move or remain stationary. In addition, nodes are unaware of their location; their exact coordinates are unknown. Moreover, the initial position of the nodes is also unknown. Each node has a fixed radio transmission range of R radius. As a result, each node forms a sensing coverage of a disk equal to πR^2 quadratic metric units. Nodes are able to move without restrictions in the area having a predefined speed of S metric units per sec. It is assumed that there is not a predefined and known moving pattern that nodes follow on their movement.

Two nodes are considered as neighbors when the distance between them is less or equal to the transmission range R. Let M_i define the neighbor list of node i, where $1 \leq i \leq N$. Obviously, these lists are continuously changed since the nodes always change position. Furthermore, let H_i denote the next hop node of each node i in the network. Each node forwards the sensed data to this node, H_i, and this process continues until the data reaches the sink node (typical reporting paradigm in WSNs). Similarly, the routing path R_i of each node is defined, where R_i includes all hops that a data packet passes until reaching the sink node.

A controller node is a node that collects network data from each sensor node and then it proceeds to data integration, process, and visualisation [17]. In the context of this paper it is considered that the sink node plays the role of the controller node. Thus, the sink node collects and process specific network data periodically. The data collection includes a) the neighbor list M_i, b) the routing path R_i, and c) the next hop H_i of each i node. The time period in which the data collection takes place is denoted by T. It is worth mentioning that the data collection is feasible when the nodes are connected with the sink node. If a node remains unconnected (e.g., due to its movement beyond the network coverage) then the controller node assumes that this node is currently out of the network coverage.

B. *Wormhole Attack Model*

A wormhole attack is a special type of attack on sensor networks in which two colluding malicious nodes use wormhole links to capture and replay communicated messages in order to disrupt the network protocol. To launch a wormhole attack, the colluded malicious nodes establish a direct communication channel between themselves bypassing several intermediate nodes. The established channel can be an out-of-band high-speed communication link or an in-band logical tunnel. Once established, the wormhole link attracts most of the traffic since the control packets traversing through a wormhole link advertise a much better link metric. Selection of such links results in denial-of-service (DoS), affecting the performance of the network severely. It is even possible to occur more than once wormhole links making the problematic situation yet harder. It has been shown that a strategic placement of the wormhole can disrupt on average 32% of all communication across the network [18].

Due to the nature of the deployment procedure it is considered that no node can be fully trusted since there is lack of any trust model in the network. Besides, nodes are not aware of their actual location and they are not equipped with location verification equipment.

When a wormhole attack is launched then a link of malicious nodes is formed. One of those two nodes advertises a low link metric and attracts the traffic originated from its neighbors. The next hop of this node is the other edge of the wormhole link. In reality, the malicious nodes are not neighbors. As a result, the network traffic that is bypassed through the wormhole link is lost. The malicious nodes that form the wormhole link are allowed to move similar to the other legitimate nodes.

IV. EyeSim: A Mobile Application for WSN Threat Visualisation

A. *Motivation*

The motivation behind the implementation of EyeSim lies in the fact that all nodes that are harmfully affected (victims) by the wormhole link remain unconnected as long as the wormhole link stays alive. Further, there is a common feature that all victims share; they all have recorded one of the malicious node (the one that advertises the low link metric) in their routing path. Hence, the malicious nodes might be exposed if the controller node forms the list of the unconnected nodes and processes the last recorded routing path of each one of those nodes. However, this methodology should be carefully followed since a false alarm may occur if the controller results in a misleading outcome. For example, it is possible for the controller node to trigger a false alarm if two (or more) legitimate nodes move out of the network coverage and loss their connection with their neighbors.

B. *EyeSim Graphical User Interface (GUI)*

EyeSim is a system that fully leverages the power of both visualisation and anomaly detection analytics to guide the user to quickly and accurately detect wormhole attacks in IP-enabled WSNs. Figure 1 illustrates the main Graphical User Interface (GUI) of the EyeSim mobile application. Figure 1a highlights the sensor network topology under investigation that is comprised of 50 randomly deployed nodes. Figure 1b depicts the network under normal operation. The lack of red warning lines indicates that no wormhole links are currently present in the network. In Figure 1c, multiple red "eye" lines are present in the visual display alerting the user that the network is under attack.

The EyeSim system builds on two core components; the *wormhole anomaly detection engine* (WAD), and the *visualisation engine*. The WAD component represents the system's automated anomaly detection logic, while the visualisation engine, is the projection tool. Next we describe, each of these components in detail.

C. *The Wormhole Anomaly Detection Engine*

The WAD engine was devised to monitor, detect, and isolate wormhole attacks in mobile WSNs that do not have a common authentication entity. This engine was designed to analyze observable patterns from time-dependent network routing dynamics. WAD is enhanced with a cognitive wormhole detection algorithm which is capable of exposing concurrent wormhole links.

Algorithm 1 illustrates the steps of WAD towards detecting and isolating the potential malicious nodes in the WSN. The algorithm receives the number of sensor nodes (N), the time period (T), the neighbor list of all nodes $M_1, M_2, \ldots, M_N$, and the next hop list of all nodes $H_1, H_2, \ldots, H_N$ and triggers or not an alarm. If an alarm is triggered a detected list of malicious nodes are provided (W). The algorithm runs at each T time period. Initially, it forms the U list of nodes that during this time period remained unconnected. The controller identifies the unconnected nodes by checking whether it received data traffic from them (during the time window T). Then it interprets the neighbor list of the unconnected nodes. It calculates the intersection of all those lists. If a common element is found then it computes the whole wormhole link by

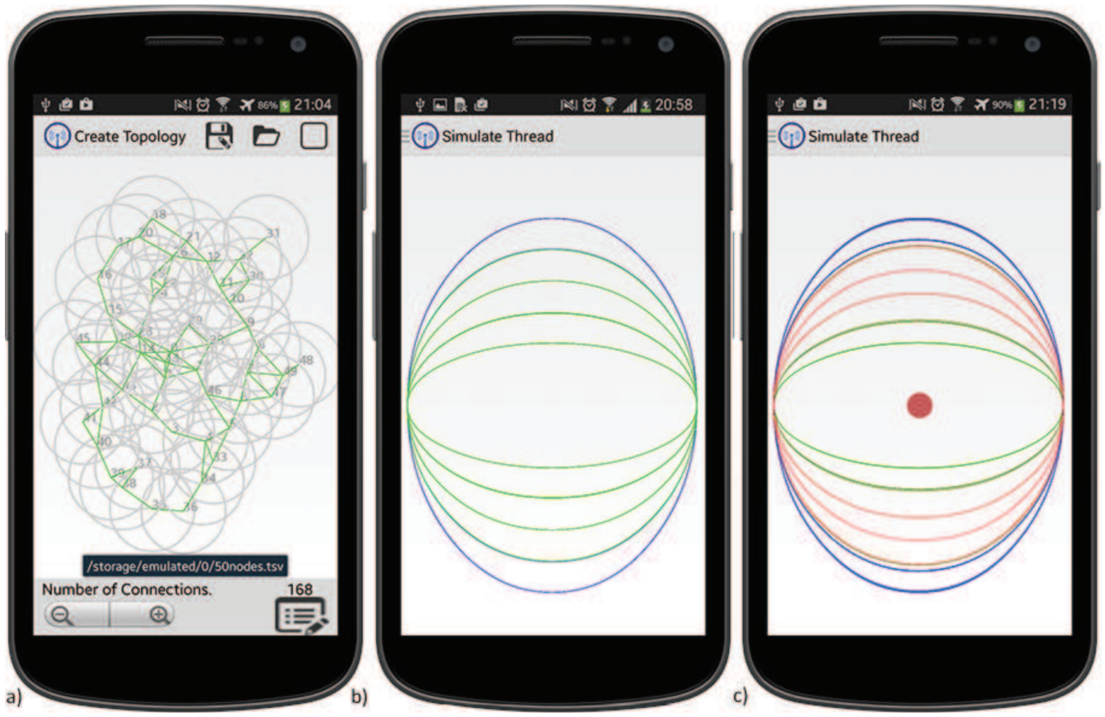


Fig. 1. EyeSim GUI showing a) the network topology comprised of 50 randomly deployed nodes, b) the network under normal operation (no warnings), and c) the network under attack (6 warnings).

taking the next hop of the initial nodes that has been extracted from the intersection. An alarm is triggered if the W set is not empty.

Algorithm 1 WAD: The Wormhole Anomaly Detection Algorithm

```
INPUT: The number of sensor nodes ($N$), the time period ($T$), the neighbor list of all nodes $M_1, M_2, \ldots, M_N$, the routing path of all nodes $R_1, R_2, \ldots, R_N$, and the next hop list of all nodes $H_1, H_2, \ldots, H_N$
OUTPUT: The list of detected malicious nodes ($W_1, W_2, \ldots$)
for each time period $T$ do
    form the $U$ list
    $W = R_{U_1} \cap R_{U_2} \cap \ldots$
    $W = W \cup H_{W_1} \cup H_{W_2} \ldots$
    if $W \neq \emptyset$ then
        Trigger an alarm
        Isolate the nodes that are included in $W$
    end if
end for
```

D. The Visualisation Engine

The main objective of the EyeSim's visualisation engine is to project the outcome of the WAD in an effective visual-assisted way. The following targets are set regarding the visual-assisted environment; a) the engine should produce a simple yet effective, dynamic visualisation interface, b) the produced visualisation interface should be cut-clear and direct by revealing the actual results of the WAD in an efficient way, c) the projected visual forms should be real-time, informative, and capable of directly indicating potential threats even in limited monitor screens such as smart phones and tablets.

In essence, EyeSim projects an eye in a 2D planar view (see Figure 1b). The eye is shaped using multiple ellipses. The behavior of each node is represented by an ellipse. Each ellipse has a characteristic color and a unique height, while all ellipses have the same width.

The height of the ellipse represents the node's latency. The node latency is measured by the controller and it is defined as the time elapsed between the last successful data reception by a sensor node and the current time. Normally, the node latency reveals how long a legitimate node remains unconnected.

Three main colors are utilized for identifying the state of each node (blue, green and red). A node may be in normal mode, unconnected, or a victim node. A time window equal to $z \times T, z = 1, 2, \ldots$ is used as a threshold for distinguishing the node state. If the node latency is lower than the time threshold, then the mode of this node is normal. The ellipse that corresponds to this node is colored with blue. On the other hand, if a node experiences latency higher than the threshold, then the WAD engine is triggered. The result of the WAD determines the state of this node. If no alarm is triggered, then all nodes that experience higher latency than the threshold are classified as unconnected. Those nodes are colored with green. Finally, if malicious nodes are exposed by WAD, those legitimate nodes that have included malicious node(s) in their routing path are deemed as victim nodes. The ellipse color for

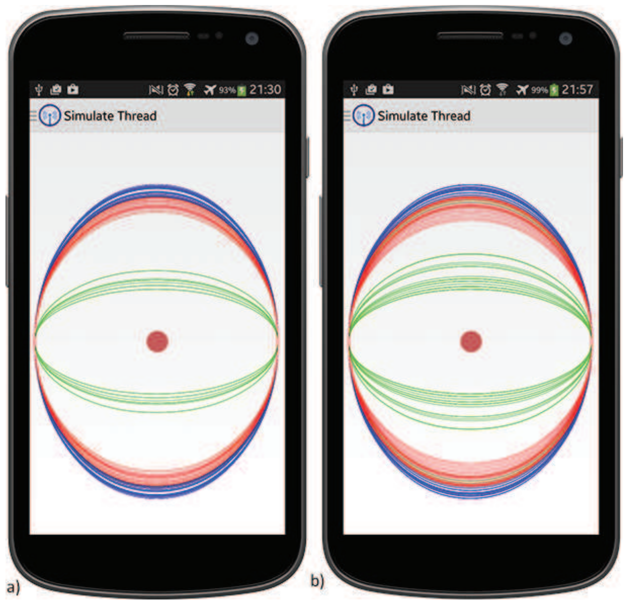
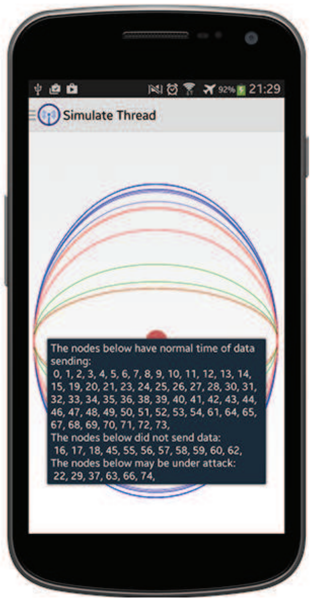


Fig. 2. EyeSim GUI showing a) a 75-node network, and b) a 100-node network under attack.



```
The nodes below have normal time of data
sending:
 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 19, 20, 21, 23, 24, 25, 26, 27, 28, 30, 31,
32, 33, 34, 35, 36, 38, 39, 40, 41, 42, 43, 44,
46, 47, 48, 49, 50, 51, 52, 53, 54, 61, 64, 65,
67, 68, 69, 70, 71, 72, 73,
The nodes below did not send data:
 16, 17, 18, 45, 55, 56, 57, 58, 59, 60, 62,
The nodes below may be under attack:
 22, 29, 37, 63, 66, 74,
```

Fig. 3. Message outputs produced by the EyeSim GUI.

highlighting the victim nodes is the red one (see Figure 1c). On the contrary, if the routing path of such a node does not include malicious node(s), then this node is characterized as unconnected node and the corresponding ellipsis is shaped with green color.

An important observation regarding EyeSim's visual design is the fact that it does not face scalability issues that many security visualisation tools typically encounter. As illustrated in Figure 2, the two different in size networks (comprised of 75 and 100 nodes respectively) are under concurrent wormhole attacks. Despite the growth in the amount of data being created and the size of the data set, EyeSim does not have any problem in efficiently projecting the scalable data, and at the same time, it is not lead to other undesired phenomena such as occlusion or overcrowding of the display [19]. Finally, it is important to note that the visualisation engine produces various messages and alerts that inform the user about the current network status or ask the user to take some actions (see Figure 3). For example, by pressing the red button in the center of the application the administrator can see the list of nodes that belong to each category, i.e., legitimate, unconnected, and victim nodes.

E. System Architecture

The architecture of the EyeSim application consists of four parts: a) the mobile client, b) the web page, c) the server, and d) the Google Cloud Messaging (GCM). Each of these parts is responsible for a different task such as for representing the network's behaviour or for calculating the new points for each node of the network, etc.

The mobile client is an application written in Java, while the web page is based on the CodeIgniter PHP web framework. The web server contains a set of scripts that help the system calculate the new position of network nodes. Additionally, both the web site and the mobile client use a MySQL database to process the network data. Besides that the database also stores user account information as well as the last position of the network nodes. Finally, the GCM is a service provided by Google and allows the system to generate and send push notifications to the mobile device. The EyeSim uses the client-server communication model between all parts. This means that the model is applied between the mobile client and web server, as well as, the web site and the web server.

F. System Analysis

As already revealed, EyeSim uses different technologies and tools to allow the application to make calculations and display the result of these calculations to the user's device. These technologies are used either as stand-alone technologies or in a combination with others to produce the expected result.

The Android operating system was chosen as the most suitable system for the development of the main part of this system, namely the mobile client. The mobile client is an application, which is written using the Java programming language in order to set up the functionality and the XML language in order to produce the user interface.

The web site is a dynamic web page, which have been written using one of the most known frameworks for web programming, namely the CodeIgniter. This framework uses the PHP programming language to connect to a database in order to store and retrieve data from it. Additionally, the CodeIgniter allows the user to create a connection to a database easily, so he/she can execute queries such as select, update or insert. This framework separates the web site into three different components; the model, the projection and the controller. Each of these components are responsible for different actions. The component of the model provides the necessary code in order to execute code against to the database such as store, retrieve or update data. The projection component provides the graphic design of the web page, which is used for the interaction with the user. Finally, the controller component provides the connection between the model and the projection, so that the latter can present to the user the information the system retrieved from the database.

Regarding PHP, this is a well known server side programming language. This programming language is executed on the web server and presents the result of the code execution to the user using the HTML language. PHP is an object orient programming language (OOP). Similar to every OOP language, it allows the user to create objects with several attributes/characteristics and to modify them through the PHP code. Within EyeSim, the PHP web programming language has been used for the development of several scripts, which are used from the mobile client in order for example to authenticate the user or to request from the server new coordinates for the network nodes. Additionally, as mentioned before, PHP was used in order to implement the functionality of the web site.

Another technology that is used within the EyeSim application is the Jquery framework. This framework allows the developer to write Javascript code in order to trigger special handlers for several HTML elements on specific user actions, such as the button click or value change in a input area.

JSON is another programming language used within EyeSim. This language is used to describe objects. It is used for the data exchange between client and server. As the name of this language shows, it is a part of JavaScript language as well. This language is independent of the programming language that is used, which make it ideal for data exchange. JSON consists of two parts. The first part is a pair of name-value, which is called object. The second part is a list of values that are ordered, and it is called array.

MySQL is the last technology used by EyeSim. MySQL is one of the biggest Relational Database Management System (RDBMS) and it is an open source. This is a type of database, which is used for web applications and runs on a server. This kind of database can be applicable to big and small web applications. It is fast, reliable and easy to use. Finally, it can be executed in the most of the well-known platforms using the standard SQL for the queries.

V. Performance Evaluation

A. Evaluation Scenario

In order to assess the proposed framework a real-time simulation scenario was considered. According to the scenario a set of 802.15.4-compliant sensor nodes was considered in a deployment area of 1280×720 meters. The number of sensor nodes was varied from 50 to 100 with a step of 10 nodes. All nodes were allowed to move within the deployment area. A fixed communication range of 100 meters was assumed for all nodes. The initial position of each node is random within the deployment area. In addition, the nodes are unaware of their position coordinates. Each node was free to move anywhere in the area. The velocity is fixed for all nodes and equal to 1m per minute. The moving pattern is simple: each node randomly changes its direction each $T = 60$ seconds. It is worth mentioning that even the sink node is always on the move. The threshold time window was set equal to $3 \times T$.

Regarding the routing protocol a simple hop count metric was considered, where the node that advertises the minimum hop distance towards the sink node is preferred by its neighbors. Two nodes are neighbors when their Euclidean distance is less or equal to the communication range (note that no obstacles were inserted in the deployment area).

EyeSim was assessed in order to demonstrate its efficacy in detecting wormhole attacks within the deployment area. To this extent, we deliberately considered two wormhole links which were active at the beginning of the experiments. Each node that forms the wormhole link advertises a quite small hop metric to its neighbors. Moreover, it advertises a next hop that is not a neighbor of the wormhole link node.

The detection accuracy of EyeSim is measured in detecting none, one, or all the wormhole links. A True Positive (TP) result is associated to EyeSim when a sensor node is identified as a wormhole link node and that node is indeed a malicious one. On the other hand, a False Negative (FN) outcome occurs when EyeSim identifies a node as a malicious one but this node is a legitimate one. Next, we demonstrate numerical results related to the probability that EyeSim detects a TP or a FN case as the number of sensor nodes is changed.

Each simulation experiment was conducted for 20 times. The average values of all experiments were extracted. An Android-based smart phone was used for the EyeSim process and evaluation.

B. Results

Figure 4 illustrates the accuracy of EyeSim in terms of TP and FN. The number of sensor nodes is varied from 50 to 100 with a step of 10. First, it is clear that as the network becomes more dense the accuracy of EyeSim becomes higher. The probability of a TP result is about 0.4 when the number of nodes is 50, while it is getting much better, equal to 0.8, when the number of nodes reaches the maximum value which is 100. This is attributed to the fact that the accuracy of EyeSim depends on the network density. As the number of sensor nodes is getting larger it is unlikely for a node to be unconnected. Given that an unconnected node may trigger a false alarm it is evident that a dense sensor network guarantees better results in terms of accuracy. Nevertheless, Figure 4 reveals the relation between TP and FN. If EyeSim detects correctly a threat then TN wins; otherwise the result is deemed as false and FN wins.

Figure 5 depicts the probability of resulting a single or double detection. Obviously, it is considered that in this case the EyeSim result was a TP case. As previously mentioned, the number of the correct decisions is getting higher as the network nodes are increased. This is the rationale behind the increase of the single detection as the network becomes more dense. More accurate decisions happen; most of them are

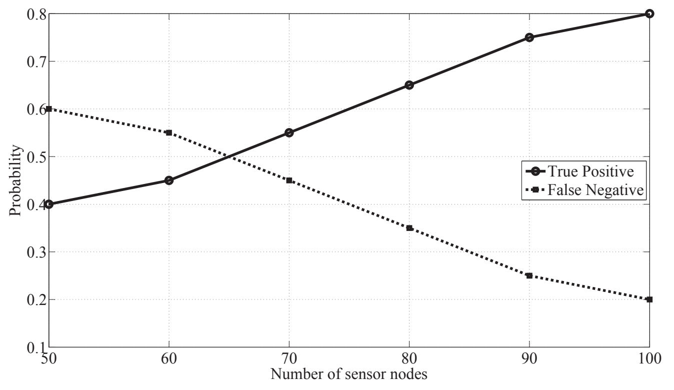


Fig. 4. Wormhole threat detection accuracy in terms of true positive and false negative.

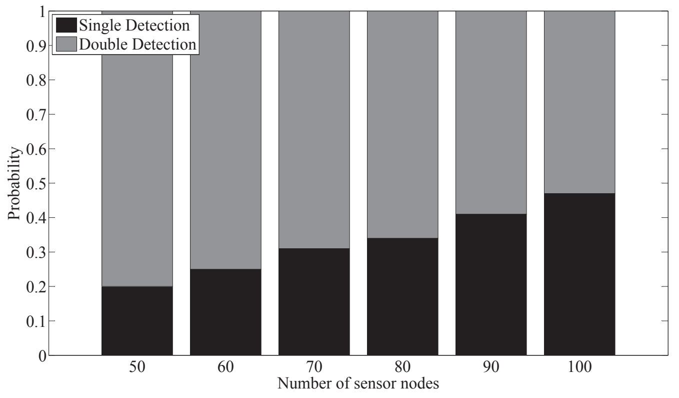


Fig. 5. Wormhole detection results. Single detection means that one of totally two wormhole link has been revealed. Double detection means that both two wormhole links have been detected.

classified as single detection. However, it is worth mentioning that the portion of totally wormhole detection is fairly high. For example, the half of detection attempts was totally correct, revealing both wormhole links, when the number of nodes was 100.

Overall, EyeSim offers a way of detecting serious threats, such as wormhole attacks, in dynamic, moving sensor networks, where multiple attacks are possible. We assessed EyeSim under pressing and demanding evaluation environment; the results demonstrate an accurate visual-assisted component which is capable of detecting multiple wormhole attacks in dynamic traffic conditions. The detection capabilities of EyeSim are maximized when the sensor network becomes dense, where the discovery of concurrent threats reaches 80%.

VI. Conclusions

The ever-increasing amount of security events reported in mission-critical applications IP-enabed WSNs are envisaged to support asks for new tools to deal with them. As a novel network security visualisation tool, EyeSim stands out as one such solution. In this work, we proposed a robust, visual-assisted anomaly detection system that is capable of identifying concurrent wormhole attacks; one of the most daunting challenges in the sensor network security field. In the future, we intend to validate the EyeSim system through extended user studies where network analysts and experts will use the system and provide feedback on its usability. Moreover, we will extend the capabilities of the EyeSim system in order to enable the tool to detect a series of new attack patterns, such as Sybil attacks, Sinkhole attacks, etc.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless Sensor Networks - A Survey,” *Computer Networks*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[2] X. Chen, K. Makki, K. Yen, and N. Pissinou, “Sensor network security: a survey,” *Communications Surveys Tutorials, IEEE*, vol. 11, no. 2, pp. 52–73, 2009.

[3] R. Marty, *Applied Security Visualization*, 1st ed. Pearson Education Inc., 2009.

[4] G. Conti, *Security Data Visualization: Graphical Techniques for Network Analysis*, 1st ed. No Starch Press, Oct. 2007.

[5] S. Card, J. Mackinlay, and B. Shneiderman, *Readings in Information Visualization: Using Vision to Think*. San Francisco: Morgan Kaufmann Publishers, 1999.

[6] D. A. Keim, “Information Visualization and Visual Data Mining,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 1–8, Jan. 2002.

[7] M. Abomhara and G. Koien, “Security and privacy in the internet of things: Current status and open issues,” in *Privacy and Security in Mobile Systems, 2014 International Conference on*, May 2014, pp. 1–8.

[8] E. Karapistoli and A. A. Economides, “ADLU: a novel anomaly detection and location-attribution algorithm for UWB wireless sensor networks,” *EURASIP Journal on Information Security*, vol. 2014, no. 1, pp. 1–12, 2014.

[9] K. Lakkaraju, R. Bearavolu, A. Slagell, W. Yurcik, and S. North, “Closing-the-Loop in s. In NVisionIP: Integrating Discovery and Search in Security Visualization,” in *Visualization for Computer Security, IEEE Workshops on*, 2005, pp. 75–82.

[10] K. Prole, J. R. Goodall, A. D. D’Amico, and J. K. Kopylec, “Wireless Cyber Assets Discovery Visualization,” in *Proceedings of the 5th International workshop on Visualization for Computer Security*, ser. VizSec ’08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 136–143.

[11] B. Parbat, A. K. Dwivedi, and O. P. Vyas, “Article: Data visualization tools for wsns: A glimpse,” *International Journal of Computer Applications*, vol. 2, no. 1, pp. 14–20, May 2010, published By Foundation of Computer Science.

[12] W. Wang and B. Bhargava, “Visualization of wormholes in sensor networks,” in *ACM workshop on Wireless Security*. ACM Press, 2004, pp. 51–60.

[13] W. Wang and A. Lu, “Interactive wormhole detection in large scale wireless networks,” in *Visual Analytics Science And Technology, IEEE Symposium On*, 2006, pp. 99–106.

[14] A. Lu, W. Wang, A. Dnyate, and X. Hu, “Sybil attack detection through global topology pattern visualization,” *Information Visualization*, vol. 10, no. 1, pp. 32–46, Jan. 2011.

[15] G. Abuaitah and B. Wang, “Secvizer: A security visualization tool for qualnet-generated traffic traces,” in *Proceedings of the 6th International Workshop on Visualization for Cyber Security (VizSec)*, ser. VizSec ’08, 2009, pp. 111–118.

[16] L. Shi, Q. Liao, Y. He, R. Li, A. Striegel, and Z. Su, “SAVE: Sensor anomaly visualization engine,” in *IEEE Conference on Visual Analytics Science and Technology (VAST)*, oct. 2011, pp. 201–210.

[17] W. Wang and A. Lu, “Visualization assisted detection of Sybil attacks in Wireless Networks,” in *Proceedings of the 3rd international workshop on Visualization for computer security*, ser. VizSEC. IEEE, 2006, pp. 51–60.

[18] M. Khabbazian, H. Mercier, and V. Bhargava, “Severity analysis and countermeasure for the wormhole attack in wireless ad hoc networks,” *Wireless Communications, IEEE Transactions on*, vol. 8, no. 2, pp. 736–745, 2009.

[19] H. Shiravi, A. Shiravi, and A. A. Ghorbani, “A survey of visualization systems for network security,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 8, pp. 1313–1329, 2012.