Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Impact of injection attacks on sensor-based continuous authentication for smartphones

Lorena Gonzalez-Manzano [a,*], Upal Mahbub [c], Jose M. de Fuentes [a], Rama Chellappa [b]

[a] Universidad Carlos III de Madrid, Spain
[b] University of Maryland, College Park, United States
[c] Qualcomm Technologies Inc. San Diego, CA, United States

ARTICLE INFO

Keywords:
Continuous Authentication (CA)
Injection attacks
Data Stream Mining
Smartphones

ABSTRACT

Given the relevance of smartphones for accessing personalized services in smart cities, Continuous Authentication (CA) mechanisms are attracting attention to avoid impersonation attacks. Some of them leverage Data Stream Mining (DSM) techniques applied over sensorial information. Injection attacks can undermine the effectiveness of DSM-based CA by fabricating artificial sensorial readings.The goal of this paper is to study the impact of injection attacks in terms of accuracy and immediacy to illustrate the time the adversary remains unnoticed. Two well-known DSM techniques (K-Nearest Neighbours and Hoeffding Adaptive Trees) and three data sources (location, gyroscope and accelerometer) are considered due to their widespread usage Results show that even if the attacker does not previously know anything about the victim, a significant attack surface arises – 1.35 min are needed, in the best case, to detect the attack on gyroscope and accelerometer and 7.27 min on location data. Moreover, we show that the type of sensor at stake and configuration settings may have a dramatic effect on countering this threat.

1. Introduction

Smartphones are involved in our daily life activities where they are being used for countless purposes. They are sophisticated devices equipped with many sensors, such as gyroscope, magnetometer, GPS, microphone, etc., to offer a plethora of services and functions. Indeed, they are increasingly being adopted as the entry point for personalized services in smart cities, such as adapted directions considering the health status of its owner [1]. Due to the relevance of these services, it is essential to authenticate the owner of the smartphone to grant access accordingly [2].

There are multiple ways to achieve such an authentication, most common being a password or a biometric trait like the fingerprint. However, when the access is granted to the smartphone, an attacker could get unrestricted control of it henceforth and perform malicious actions like stealing information. Continuous Authentication (CA) techniques address this shortcoming, for instance, using the accelerometer and gyroscope to periodically verify the user's identity [3] or combining the use of the GPS and screen touches to do regular identity checks [4, 5].

Most smartphone CA proposals leverage sensors data assuming they are trusted sources of information and thus disregarding their potential threats [6]. One of these is the injection attack [7], which is gaining momentum. In this attack, source data is maliciously manipulated to circumvent a given security mechanism. [8] shows the possibility of performing an injection attack in a GPS device to infer location data. As pointed out in [9], position sensors have been exploited by the attackers, enabling them to corrupt or even control their output. This can be achieved, for example, by applying an acoustic or magnetic stimulus over the device so that sensors start malfunctioning. On the other hand, sensorial information coming from touch screens has also been compromised. In particular, [10] proposes a mechanism to create fake touch events by injecting external noise signals through tailored electrical voltages and [11] presents SMAShed, a framework to manipulate several Android sensors. Going a step further, [12] highlights fake sensor data injection as one of the threats in IoT devices and applications. Indeed, Giannetsos et al. already developed a working malware that may alter the data offered by a node within a sensor network [13].

In the context of smartphone CA, injection attacks may open up a significant window of opportunity for the adversary. In particular, they may enable the impersonation of the owner, thus creating the illusion as though the legitimate user controlling the device. This may be helpful to grant access to the device to a robber. Moreover, if the CA decision enables granting some privileges to an external service

* Corresponding author.
E-mail addresses: lgmanzan@inf.uc3m.es (L. Gonzalez-Manzano), upalmahbub@yahoo.com (U. Mahbub), jfuentes@inf.uc3m.es (J.M. de Fuentes), Rama@umiacs.umd.edu (R. Chellappa).

https://doi.org/10.1016/j.comcom.2020.08.022
Received 6 March 2020; Received in revised form 19 June 2020; Accepted 28 August 2020
Available online 5 September 2020

(e.g., access corporate sensitive data), this may lead to unlawful access to them.

This kind of attack can be developed by means of malware, which is increasingly present in smartphones [14,15], or even in more sophisticated threats like advanced persistent ones (APTs) [16]. Indeed, sensor data has been at stake of malwares. For example, in 2019 a malware was shown to selectively activate its functions based on motion sensors, as a means to evade detection methods.[1] If a malware were to compromise a CA mechanism, it could record the sensorial readings from the legitimate user, and then reproduce (i.e., inject) them once the device has been stolen. However, this would involve a non-negligible storage which could simplify its detection. Therefore, it is more realistic to assume that the malware could on-the-fly inject manipulated sensor readings following a given policy (with or without previous knowledge on the victim) without storing any actual readings. However, despite the possibility of such smartphone sensors injection attacks [8], this issue has been barely addressed so far.

To overcome this limitation, the goal of this paper is to study the impact of injection attacks in the accuracy and immediacy of smartphone CA schemes. This serves to illustrate the attacker window — the time the adversary remains unnoticed. It must be noted that the attack succeeds if it is never detected or if detection takes a lot of time. In particular, three research questions are at stake:

- (RQ1) *Are CA mechanisms effective against injection attacks?*
- (RQ2) *Do injection types affect the attacker window?*
- (RQ3) *Does the amount of injected data affect the attacker window?*

To the best of authors' knowledge, no previous work has addressed these questions. There are several related works, e.g. [9,17], but they do not provide any insights on the size of the attacker window. Among all existing approaches for CA, we focus on those based on Data Stream Mining (DSM) techniques because they are specially appropriate for processing endless flows of data, in memory-limited devices, at a really fast pace [18], which is the operational environment of the proposed system. DSM has been also considered useful to prove the authenticity of data sources [19] and more specifically it has been used for CA [20]. In particular, two widespread DSM methods, namely K-Nearest Neighbour (KNN) and Hoeffding Adaptive Trees (HATs), are considered. Concerning the data at stake, motion sensors (gyroscope and accelerometer) and location data are used for this purpose, as they are two well-known authentication features [21]. To ensure real-world validity, our analysis is carried out on a real-life dataset of 47 users collected over three years. The relevance of this dataset for our purposes lies on the size of the time span — having data from several months enables us to determine the long-term validity of our results. Since CA decisions may take place frequently, it is critical to ensure that the data at hand actually describes the daily routine of several users throughout the year. Otherwise, results could be biased by choosing a particular user or period. In this regard, to the best of authors' knowledge this is the largest available dataset. Moreover, we consider two realistic adversaries — one that has some previous knowledge on the victim (e.g., a robber that has previously tricked the user into installing a malicious spyware app) and another one without that knowledge.

This paper is structured as follows: Section 2 presents background discussions. Section 3 describes the underlying model. Section 4 describes an illustrative use case for the attack. The achieved results are presented and discussed in Section 5. Section 6 introduces related work. Finally, Section 7 concludes the paper and points out future work directions.

[1] https://pentesttools.net/new-android-malware-apps-use-motion-sensor-to-evade-detection/

2. Background

This paper focuses on CA techniques that are based on exploiting sensor information. Since Data Stream Mining (DSM) algorithms are usually at the core of these CA approaches, this Section introduces the basis of DSM (Section 2.1). Subsequently, the subset of DSM algorithms considered in this paper are presented (Section 2.2). Finally, types of sensor injection in smartphones are presented (Section 2.3).

2.1. Data stream mining

Data stream mining techniques are a branch of machine learning procedures [18]. In machine learning, typically a pattern or feature is learned to classify inputs to classes afterwards. The dataset is divided into training and testing sets for this process, the former to build the model and the latter to verify the correctness of the classification.

However, most machine learning algorithms have limitations [18], specially when there is potentially infinite set of data at stake instead of a pre-known, fixed-size dataset. Moreover, due to limited computational resources and memory all managed elements cannot be stored. Data Stream Mining (DSM) techniques are proposed to address these limitations [22]. DSM is suitable for settings in which flow of data are received, storing in memory just some of the continuous flow. In this way, the system learns from elements that are processed in windows of a certain size and only once following their arrival order. Thus, for instance, a window of x bytes means that flows of data are processed in batches of x bytes.

DSM has the following requirements for processing continuous flow of data [18]:

- *Processing just one sample at a time*: data is processed as it arrives and retrieval is not allowed. In cases where re-analysis of a data-stream is practical, this restriction can be relaxed.
- *Limited amount of memory*: the processing of data is not limited to memory because DSM techniques can process infinite amount of data while just the current model and some useful statistics are stored in memory.
- *Work time limits*: to process data in real time, the algorithms need to work at a high throughput rate.
- *Instant predictions*: the system needs to provide predictions efficiently regardless of the amount of samples and without recomputations at any stage.

Sensor-based CA for smartphones should consider all the requirements mentioned above. This poses some practical limitations for their real-life implementation. First, smartphones do not have unlimited memory, so it is necessary to opt for computation-saving procedures. Second, as the authentication is continuous, the system should periodically provide results as fast as possible. Finally, the removal of the training phase is a preferred approach from the usability point of view.

2.2. Classification algorithms

Among existing classification algorithms, the following ones are applied herein [23] as they are the most prominent techniques in CA approaches leveraging IoT devices like smartphones [21]:

- K-Nearest Neighbour (KNN). The typical algorithm looks for instances in the training set that are more similar to the instance to classify. In particular, the used algorithm classifies an instance based on its k nearest neighbours according to a distance. In our case, the distance $d(x,y)$ is computed using the Euclidean distance. For two-dimensional points, this is computed following Eq. (1).

$$d(x,y) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{1}$$

The user is classified considering the predominant label of the k nearest points.

- Hoeffding Adaptive Trees (HAT). In order to understand HAT, the concept of Hoeffding Window Tree (HWT) is required. A HWT is a decision tree based on a sliding window keeping the last instances on the stream. They use the Hoeffding bound, which states that with probability 1- δ, the true mean of a random variable of range R will not differ from the estimated mean after n independent observations by more than ϵ (Eq. (2)).

$$\epsilon = \sqrt{\frac{R^2 \cdot (ln(\frac{1}{\delta}))}{2 \cdot n}} \tag{2}$$

Based on this concept, a HAT is a HWT that learns from data streams without a fixed size of sliding window. In this paper we focus on the ADWIN variant, which automatically and continuously estimates the rate of change in the data streams rather than using a priori guesses. With this estimation, ADWIN adapts the window size accordingly, using narrower windows when data changes at a higher pace.

In this work we opt for KNN and HAT as they are two widespread DSM alternatives. Moreover, they have been shown to be comparable to other alternatives, or even better. For instance, as compared to Concept-adapting Very Fast Decision Trees (CVFDT), HAT works quite similarly but offering better performance [18].

2.3. Sensor injection techniques in smartphones

Sensor injection attacks have been studied in numerous disciplines and devices but the way injection can be carried out has been little studied, specially in the IoT field and smartphones in particular.

Injection can be achieved through signal manipulation [24] or malware infection [15]. Besides, the attacker's knowledge of the victim's sensor data, before the injection attack, should be also considered. In the IoT field there are three works that study injection, though the focus of [15] is not related to the one proposed herein. [15] and [17] introduce the injection of sensor data in between the minimum and maximum values, thus assuming some knowledge to establish such limits. By contrast, [9] proposes two general types of injection, one based on corrupting data and assuming certain knowledge of the victim; and another one based on rewriting data, such that data of a user is injected in other user's data (victim), so no knowledge of the victim is assumed. However, there are no particular details given on how the injection is really carried out.

3. Model

This Section describes the model on top of which the experimental analysis is carried out. Section 3.1 gives a broad overview of the system settings. Section 3.2 describes the adversaries under consideration and their capabilities. Last but not least, the goals that are relevant in order to determine the effectiveness of any CA approach are described in Section 3.3.

3.1. Overview

The focus of this paper is to characterize the impact of injection attacks on the immediacy and accuracy of CA. For this purpose, several attacker types are considered depending on the previous knowledge on the target system, as it will be explained in Section 3.2. In particular, three sensors are taken into account, namely location, gyroscope and accelerometer.

An overview of the analysis to be carried out is depicted in Fig. 1. First, multi-sensor information is collected in a given smartphone. Each sensor provides several features or attributes. Afterwards, such data is preprocessed to choose the most representative features.

At this point in the process, data could be applied for CA purposes. However, in order to determine their resiliency against injection, the third step is data modification following different strategies. After this modification, two DSM techniques, namely KNN and HAT are applied to this data for CA purposes. Finally, the output of each DSM algorithm under different settings is analysed to conclude how injection affects the CA process in terms of the attacker window.

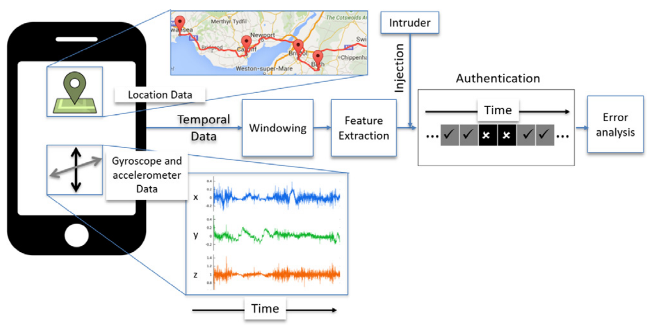


Fig. 1. Overview of the analysis steps.

3.2. Adversarial model and types of injection

The goal of an adversary A is to inject data in such a way that the smartphone authenticates A as the legitimate user U. Therefore, A is willing to impersonate U in order to inherit U's access rights to the smartphone functions and data.

In order to achieve this impersonation, A needs to modify the sensor information to defeat the CA mechanism at stake. In terms of location data, injection would be analogous to using the smartphone in a different place than that of U but forging spatial information in such a way that the system is unable to notice A. By contrast, injection in gyroscope and accelerometer data means that A could emulate a movement which is different from that of U but the system is unable to notice the presence of A.

Disregarding the type of sensor at stake, such an impersonation requires A controlling $U's$ device to some extent at some point in time. Thus, A may either control some environmental condition to cause sensor malfunction (e.g. presence of voltage), or inject fake data by means of malware (recall Section 2.3). In any case, the following pair of adversaries are considered depending on the prior knowledge that A has regarding U:

- **No-previous-knowledge adversary** (A_{NK}): A injects data without having any kind of previous information about U's legitimate sensor data.
- **Partial-previous-Knowledge adversary** (A_{PK}): A knows U's legitimate sensor data collected over a period of time before the attack. For example, this might be achieved by previously infecting U's smartphone with a spying application.

In order to carry out the experimental analysis, it is assumed that the adversary takes control of U's device and performs injection from that moment on. Concerning the types of injection, different patterns could be devised such as the addition of uniform or Laplace noise [25]. In this work we opt for a uniform distribution as it is more convenient when it comes to smartphone sensor data [15]. However, each adversary applies it in a different way.

In the following, let $R_U^T = \{f_{U,1}^T, \dots, f_{U,n}^T\}$ be a legitimate sensor reading of U at time T. Each one is formed by a set of features $f_{U,i}^T$. Thus, after injection they are transformed into a modified reading $\overline{R_U^T} = \{\overline{f_{U,1}^T}, \dots, \overline{f_{U,n}^T}\}$

- **Blind Rate (BR)**: A_{NK} injects data in a fixed way, that is modifying the original data a particular percentage referred to as ip.

In particular, for each R_U^T, all features $f_{U,i}^T$ are either increased or decreased at a fixed ratio Eq. (3).

$$\overline{R_U^T} = \{f_{U,1}^T \cdot (1 \pm ip), \dots, f_{U,n}^T \cdot (1 \pm ip)\} \tag{3}$$

- **Slogger (S)**: following the pattern proposed in [15] and assuming partial knowledge of U's data, A_{PK} computes the maximum and minimum values per $f_{U,i}$, to inject data randomly within such computed limits. More specifically, it is assumed that the attacker knows the victim between $T1$ and Tk, thus knowing R_U^{T1} to R_U^{Tk}. Depending on this background information, reference values are computed as $maxF_{U,i} = max(f_{U,i}^{T1}, \dots, f_{U,i}^{Tk})$ and $minF_{U,i} = min(f_{U,i}^{T1}, \dots, f_{U,i}^{Tk})$. Based on them, the injection percentage per feature $ip(f_{U,i})$ is computed following Eq. (4), where $random(a, b)$ produces a random value between a and b. Afterwards, readings are modified following Eq. (5).

$$ip(f_{U,i}) = random(minF_{U,i}, maxF_{U,i}) \tag{4}$$

$$\overline{R_U^T} = \{f_{U,1}^T \cdot (1 \pm ip(f_{U,1})), \dots, f_{U,n}^T \cdot (1 \pm ip(f_{U,n}))\} \tag{5}$$

3.3. Goals

In order to measure the degree of resilience of DSM techniques leveraging different smartphone sensor data, it is necessary to define the goals for such a CA system. In particular, the following two indicators are the main targets of the analysis:

- **Accuracy**: the system should distinguish U from A. It means that impersonation feasibility, the fact of A being similar to U, should be minimized.
- **Immediacy**: the system should detect the presence of A, that is the injection, as soon as possible. A threshold (TH) is introduced to determine how many data entries (also called records) are to be classified as A. The smaller the value of TH, the better for security because A is detected faster. However, small TH might be worse for usability because the chances of getting false positive responses will increase leading to frequently blocking of the phone even in the presence of U. The best TH value should be chosen to reach a balance between security and usability.

4. Use case

Alice, the victim, uses gyroscope and accelerometer or location sensor data for being continuously authenticated in her smartphone. Bob, the attacker, prepares a malicious application which is able to inject data in smartphone sensors [15]. The application is downloaded by Alice and once the malicious application is running, Bob robs the device and gets access to Alice's smartphone to exfiltrate, collect or visualize data. Considering that 5 min is the maximum time required for learning to use a mobile application [26], this is the time needed for Bob when unknown applications are at stake. However, 15 s. is enough for many common uses such as checking and dismissing a notification, quickly responding to a message or searching for something in the mobile phone browser [27]. We will refer to them as *quick interactions*. On the other hand, 107.6 s., 63.6 s. and 54.0 s. are the average usage times for social networking (Facebook, Twitter, Instagram, OkCupid), SMS/ Texting (built-in Messaging) and email (Gmail, Yahoo mail, SolMail) applications respectively [28]. These values are illustrative for the time Bob may spend in stealing information or misbehaving through these applications.

However, the time of the attacker is bounded by that of Alice to identify the robbery. According to a survey, 15 min is the time needed for a third of mobile users to notice the loss of their devices and 5 min for a quarter of young mobile users [29]. Then, time between such limits is the one Bob may use the stolen smartphone without being noticed. From then on, it is assumed that Alice applies some kind of countermeasure, such as remotely deleting all information stored in the device.[2]

5. Evaluation

This Section presents experimental results, starting by the description of the main implementation issues (Section 5.1). The assessment process is composed of five different parts (recall Fig. 1). The dataset is firstly prepared (Section 5.2). Secondly, data is analysed (Section 5.3). Then, authentication is enforced by applying different DSM with a set of configuration parameters. To measure the impact of injection attacks, different variables (recall Section 3.3) are studied (Section 5.4).

5.1. Implementation

Python is the programming language used to generate multiple scripts for doing data processing and injection, and evaluating results according to applied CA algorithms. Different standard Python libraries like cvs, os or numpy are used for this purpose.

In particular, for feature selection, the process performed just before injection, Weka,[3] a machine learning open source tool, is applied. Moreover, in terms of data injection, impersonating U involves controlling his/her device at some point in time. Therefore, each file is formed by a set $\{R_U^{T1}, \dots, R_U^{Tleg}\}$ containing leg legitimate sensor readings, followed by $\{\overline{R_U^{Tleg+1}}, \dots, \overline{R_U^{Tend}}\}$ injected records. The size of this second set, $w = end - (leg + 1)$ is the width of the DSM algorithm window. In this way, the maximum expected amount of True Positives (TP) and True Negatives (TN) are known beforehand which are leg and w, respectively. Algorithms 1 and 2 show how the injection is implemented according to Section 3.2.

```
Data: Operation (op), 1-addition/ 0-subtraction; Injection percentage
      (ip); $w$;$leg$
Result: New file with injected data ($\overline{File_U}$)
for each $File_U$ do
    counter=0
    while counter< $leg$ do
        $\overline{File_U}$=+$R_U^{Tj}$
        counter++
    end
    while counter< $w$ do
        for each reading $R_U^{Tj}$ in $File_U$ do
            for each feature $fuF_i$ in $R_U^{Tj}$ do
                if op==1 then
                    $\overline{R_U^{Tj}}$+=$fuF_i$+$fuF_i$*ip;
                else
                    $\overline{R_U^{Tj}}$+=$fuF_i$-$fuF_i$*ip;
                end
            end
            $\overline{File_U}$=+$\overline{R_U^{Tj}}$
        end
        counter++
    end
end
```

Algorithm 1: Blind rate injection algorithm

Finally, once injections are carried out, MOA,[4] an open source framework for data stream mining, is used to study the CA process, using existing KNN and HAT algorithms properly tuned with parameters w and k detailed in the following sections.

[2] https://support.google.com/accounts/answer/6160491?hl=en, last access June 2020.

[3] https://www.cs.waikato.ac.nz/ml/weka/ last access June 2020.

[4] https://moa.cms.waikato.ac.nz/ last access June 2020.

```
Data: Operation (op), 1-addition/ 0-subtraction; $Tk$; $w$; $leg$
        Result: New file with injected data ($\overline{File_U}$)
for each $File_U$ do
    counter=0
    while counter< $leg$ do
        $\overline{File_U}$=+$R_U^{Tj}$ counter++
    end
    attackerKnowledge =0 while attackerKnowledge<$Tk$ do
        for each $R_U^{Tj}$ in $File_U$ do
            $maxu_i f_j$=getMaxPerFeature($R_U^{Tj}$);
            $minu_i f_j$=getMinPerFeature($R_U^{Tj}$);
        end
        attackerKnowledge++
    end
    while counter< $w$ do
        for each $R_U^{Tj}$ in $File_U$ do
            for each feature $fuF_i$ in $lineuF_i$ do
                ip = randomBetween($maxu_i f_j$,$minu_i f_j$);
                if op==1 then
                    $\overline{R_U^{Tj}}$+=$fuF_i$+$fuF_i$*ip;
                else
                    $\overline{R_U^{Tj}}$+=$fuF_i$-$fuF_i$*ip;
                end
            end
            $\overline{File_U}$=+$\overline{R_U^{Tj}}$
        end
        counter++
    end
end
```

Algorithm 2: Slogger injection algorithm

For the sake of repeatability, the main scripts for running the experiments have been released in a GitHub repository.[5]

5.2. Data preparation

In order to ensure the real-world validity of the analysis, it is critical to use a realistic dataset. For this purpose, Sherlock dataset [30] is considered in these experiments. Sherlock is a comprehensive dataset collected from 47 users for a maximum of 36 months. For each user, sensor data was captured every minute (location sensor) or 15 s (gyroscope/accelerometer). Thus, after processing all users' data, each one counts on around 833.3 h. of location data and 208.3 h. of gyroscope and accelerometer data. To promote the realism of the data, these users were silently monitored (using a legitimate spyware app) while using their device in their daily routines.

Among all Sherlock data, this analysis focuses on three sensor sources — location, gyroscope and accelerometer. Within Sherlock, each location record is composed of 12 features, collected every minute. It must be noted that for privacy reasons, these features are obtained after applying a pseudoanonymisation technique (k-means in this dataset). This contributes to support the validity of our process even when privacy-preserving techniques are in place (for example to comply with legal regulations in force). On the other hand, gyroscope and accelerometer information is formed by 90 features collected every 15 s.

Considering existing CA approaches leveraging sensor data in smartphones, typically gyroscope and accelerometer are processed together. Therefore, each user U has two associated files — one for his/her location and another one containing gyroscope and accelerometer information.

Once this set of files has been created, feature selection is carried out. Three different algorithms are considered for this purpose:

- Correlation-based Feature Selection (CFS): selects a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. It outputs a subset of chosen features.
- Attribute correlation (AC): evaluates attributes measuring the correlation (Pearson's) between it and the class. It outputs a ranking of features.
- Information gain (IG): evaluates attributes measuring the information gain with respect to the class. It outputs a ranking of features.

In order to choose the most representative features, these three algorithms have been evaluated. In particular, the output of CFS gives a pre-filtered feature set. On the other hand, the top 40% of features of each of AC and IG rankings are computed. Thus, only those features selected by at least 2 of the aforementioned algorithms are chosen. As a result, 5 features are selected for location and 25 for gyroscope and accelerometer (see Appendix).

Moreover, to measure the impact of injection on CA techniques (more precisely, on their underlying DSM algorithms), it is necessary to prepare the files to be processed by each DSM. Thus, a set of files per sensorial source (location and the combination of gyroscope and accelerometer) are created per user U as explained in Section 5.1.

In our experiments, a pair of window sizes, $w_1 = 1000$ and $w_2 = 10{,}000$ are applied, considering that R_U^{Ti} refers to 223 bytes for accelerometer and gyroscope and 33 bytes for location, such that the maximum storage space refers to 2.2 MB and 330 KB in each case respectively. Since this information is to be stored in RAM memory, there are two practical limitations to consider. On the one hand, the total amount of RAM, which is usually 1 GB in many countries according to DeviceAtlas.[6] On the other hand, the maximum RAM share per app imposed by current smartphone operating systems, such as the Android memory management policy.[7] Therefore, these values for w ensure that this mechanism is suitable for the most constrained current devices. Moreover, leg is set to 25%, 50% and 75% of w_1 and w_2 values to do the analyses in greater depth.

Based on two adversarial attacks discussed in Section 3.2, injection strategies have been implemented as follows:

- Blind Rate (BR): percentage of injection is set to {−10%, 5%, 25%, 50%, 200%}, considering results presented later in Section 5.3.
- Slogger (S): the maximum and minimum values in R_o for each file are computed to inject random data in between such values.

In light of the above and to simulate that injection attacks can occur at any point in time of the usage period, experimental files are developed considering that attacks are carried out at the very beginning, as well as at 20%, 40%, 60% and 80% of the usage periods. Thus, each type of injection generates 150 files per user. In the case of BR such number bases on the use of 5 percentages of injections, w, leg and usage periods. While the amount of files for S injections is computed based on w, leg, usage periods and 5 repetitions for using a random operation. All in all, it results in a set of 14,100 experimental files.

5.3. Data suitability assessment for CA. preliminary exploration

Before analysing the effect of injection, it is important to determine whether the sensor data at hand is useful for CA purposes. For example, if all sensor data were the same for all users, CA could simply not be

[5] github.com/lgmanzan/impactSensorInjection

[6] https://deviceatlas.com/blog/most-common-smartphone-ram-by-country, last access June 2020.

[7] https://developer.android.com/topic/performance/memory, last access June 2020.

achieved — it would be impossible for any system to tell the two users apart.

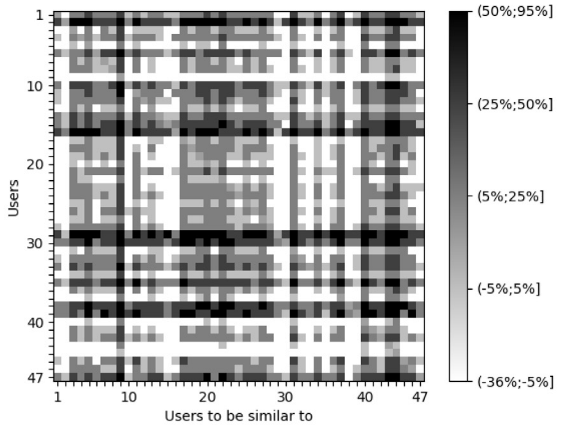
To assess the suitability of the considered data, the relative distance $dist(U_i, U_j, s)$ between two users U_i and U_j concerning sensor s is measured. Let Ts and Te be the start and end times of each recorded sensor. The average per user and feature is computed as $\widehat{f_{U,i}} = avg(f_{U,i}^{Ts}, \dots, f_{U,i}^{Te})$. Thus, the said distance is computed following Eq. (6), where nf is the number of features for sensor s.

$$dist(U_i, U_j, s) = \frac{(\sum_{x=1}^{nf} \frac{\widehat{f_{Ui,x}}}{\widehat{f_{Uj,x}}}) \cdot 100}{nf} \tag{6}$$

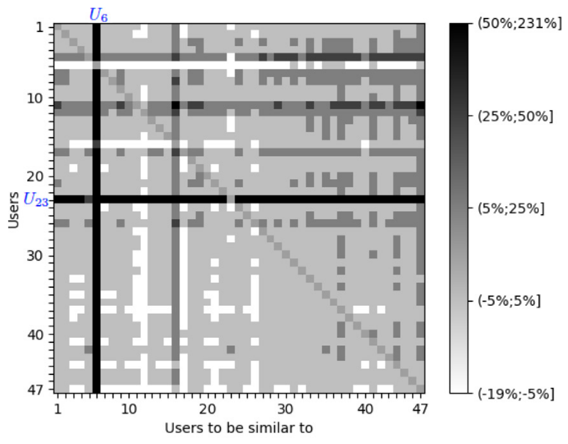
In terms of A_{NK}, which follows a BR strategy, the computed distances serve as an inspiration for defining injection rates. In other words, this gives us the amount of average injection rate that has to be applied for a user to resemble another.

Figs. 2(a) and 2(b) show the percentage of modification needed in the data of users in axis Y to become similar to users in axis X. In these figures, the scale is experimentally defined by observing the distribution of values for all $dist(U_i, U_j)$. These figures show that there is a lack of uniformity in the considered sensorial data. This is beneficial for the sake of distinguishability among users, as inter-user distances exhibit great variety. For instance, concerning Fig. 2(b), all users are very different from U_6, that is all users have to modify their data between (50%; 231%) to become similar to user U_6; and U_{23} is very different from the rest of users, that is U_{23} has to modify his/her data between (50%; 231%] to become similar to the remaining users. Note that the percentage of increase works in line with Eq. (3) related to BR strategy.

As a result, this analysis shows that some users can resemble others by modifying their sensorial inputs to some extent. It is remarkable that the mean of all features is considered herein and then, the same modification for all features is assumed. This decision has been taken considering that A_{NK} will inject data following the same pattern in every feature.



(a) Location



(b) Gyroscope and accelerometer

Fig. 2. Inter-users data similarity, in percentage.

5.4. Analysis on the impact of injection

This Section focuses on the results of the impact of the injection strategies on the prepared files. For this purpose, Section 5.4.1 defines which are the metrics that will be considered for the established goals, namely accuracy and immediacy. Section 5.4.2 discusses the global impact of injection into the system performance, focusing on accuracy. However, immediacy is essential in this system, being analysed in Section 5.4.3. Last but not least, Section 5.4.4 summarizes the main findings of the analysis.

5.4.1. Metrics

Both accuracy and immediacy are the variables under consideration in this analysis. However, it is necessary to determine which are the factors that will be considered to measure them. Four metrics are specially relevant in this study, accuracy, First Interval of False Positives (FIFP), First Interval of True Negatives (FITN) and Detection Time (DT). Each one is introduced below.

Accuracy is calculated as the amount of correctly predicted instances divided by the total amount of predictions, presented in Eq. (7). Then, it involves True Positive (TP), True Negative (TN), False Positives (FP) and False Negatives (FN). In this particular scenario, TP means that the system correctly predicts U and TN that the system correctly predicts A, while FP and FN represents the opposite respectively. Indeed, TP and TN offer a global metric on the accuracy of the DSM at stake even in the presence of injection.

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{7}$$

On the other hand, FIFP represents the period an attacker remains unnoticed until the first detection and thus, the smaller this value is, the lower the chances for the adversary A to succeed. FITN shows the period needed to detect an attacker and it is bounded by a given threshold TH, such that $FITN > TH$. TH is defined for the sake of usability — otherwise, any FN would cause the system to activate protective measures, thus imposing a non-negligible burden on the user.

The last metric, Detection Time (DT), bases on both FIFP and TH. Detecting A will take as much time as the amount of detection errors plus the time to determine A's presence, following Eq. (8):

$$DT = \alpha \times (FIFP + TH) \tag{8}$$

where α, the sampling interval, corresponds to 1 min in case of location and 15 s for gyroscope and accelerometer data for the Sherlock dataset (recall Section 5.2). Reasonable values for TH should not be higher than several minutes because A would take advantage of the situation otherwise. For the sake of generality, we consider the amount of records that are equivalent to 5 min, inspired by the most challenging limit imposed in the use case. This equals to 5 records for location and 20 for gyroscope and accelerometer. However, we adopt an intermediate value of 10 records to enrich the discussion.

Fig. 3 shows an example of FIFP, FITN and DT, where U means the data belongs to the user and A means an injection attack. Array (1) shows readings of the legitimate user U, followed by updated readings which the attacker (A) has injected. Array (2) shows predicted values, appearing FIFP after some initial predictions which correspond to data before injection. FITN appears afterwards, leading to protected measures when it exceeds TH. Thus, the challenge is to minimize FIFP and producing FITN at the earliest to reduce the changes of the attacker for going unnoticed.

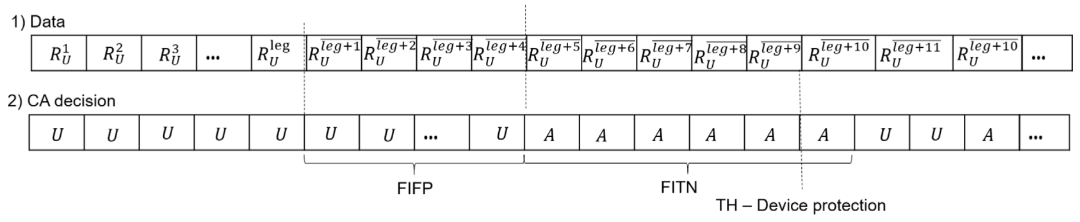


Fig. 3. Example of FIFP and FITN considering TH = 5.

5.4.2. Global impact of injection

According to the established goals, *accuracy* requires maximizing TP and minimizing FP. Besides, *immediacy* is achieved reducing TN in terms of an established TH.

Table 1 presents the mean of TP and TN rates per DSM technique, w and type of sensor data. Generally speaking, injection attacks have a very discrete impact into TPs, which is beneficial for usability — each time U is present, it is successfully authenticated as such. The situation is rather similar for TN, except for HAT and w = 10,000, where the CA system makes an error 13.2% of the times. Therefore, injection attacks introduce some degree of distortion into the performance of CA mechanisms, being immediacy a complementary key element to study.

5.4.3. Impact on immediacy

This analysis focuses on immediacy, studying FIFP, FITN and DT. The following graphical elements are applied:

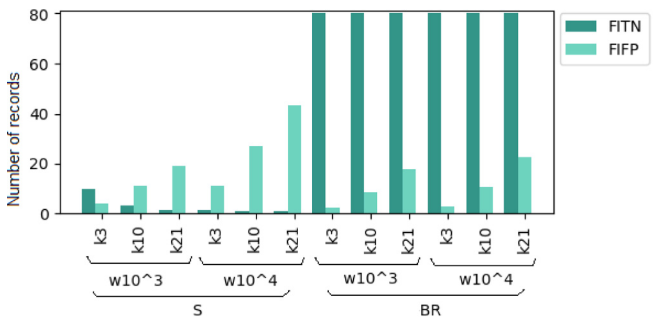
- Plots depicting FITN and FIFP: for readability purposes, the maximum number of presented registers is 80, as it is regarded as an extreme value for TH.
- Plots presenting DT: as a trade-off between security and usability and considering that DT bases on TH and that it should be within time limits set in the use case (Section 4).

5.4.3.1. Effect of k value in KNN. The use of KNN algorithm requires the specification of parameter k. There is no single, globally accepted optimum value for k, so typically a trial-and-error approach is followed [31]. However, multiple authentication approaches commonly use small k values, e.g. [32,33], and [34] studies the result from k = 1 to k = 21. Considering these studies and the fact that several tests confirm that results get worse when k increases, in these experiments k has been set to 3, 10 and 21 for comparison purposes. Figs. 4(a) and 4(b) show FIFP and FITN for each type of injection and w. The system is able to identify the attacker in both injection types but there are relevant differences.

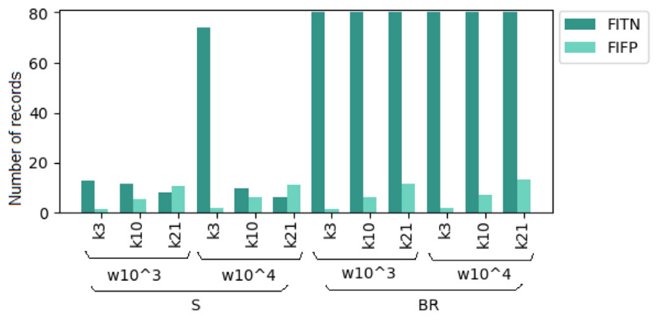
In terms of k value, the smaller k, the better in all cases. Usually a small value of k means that noise will have a higher influence on results. However, for location and inertial sensor data, the training set might have a wide variation and therefore considering more samples would defeat the basic philosophy behind KNN that the points near might have similar density or classes. It is in line with our previous work [20], so it seems to be a typical issue. Results are acceptable in BR injection for small k values and w=1000 specially. The best results are achieved for w = 1000 and k = 3, such that FITN = 152.11 and FIFP = 2.27 for location; and FITN = 132.13 and FIFP = 1.38 for gyroscope and accelerometer.

S injection leads to adequate results if TH is carefully chosen, for instance, for location, when k = {10, 21}, TH should be set to 1 to detect A_{PK} in the first interval, except for k = 10 and w = 1000 in which TH should be set to 3. Nonetheless, TH = 1 is not a reasonable value for usability because in that setting one mistake of the CA mechanism would lead to the user being rejected from his/her own smartphone. In a better scenario, like k = 3 and w = 1000, FITN = 9.69 and 12.99 (a high but sensible TH) and FIFP = 3.81 and 1.28 for location, and gyroscope and accelerometer, respectively.

Table 2 shows DT for different values of k and injection types, where omitted values (-) mean that FITN< TH and thus, we consider



(a) Location



(b) Gyroscope and accelerometer

Fig. 4. FIFP and FITN for k value analysis.

Table 1
Mean TP, TN and accuracy rates for each sensor and DSM technique.

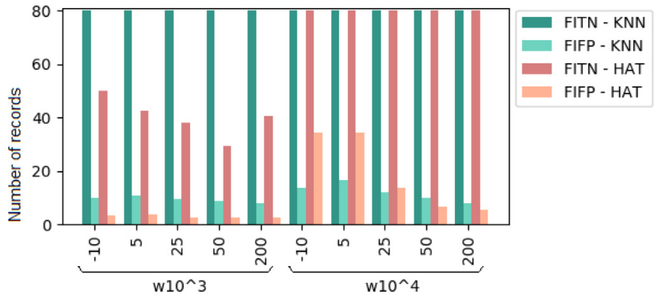
		w	TP rate	TN rate	Accuracy rate
Location	KNN	1000	99.8	95.0	96.65
		10,000	100.0	96.0	97.67
	HAT	1000	99.7	94.1	96.05
		10,000	100.0	86.8	91.51
Gyroscope and Accelerometer	KNN	1000	100.0	97.9	98.45
		10,000	100.0	98.4	98.92
	HAT	1000	99.6	97.8	98.51
		10,000	100.0	97.5	98.34

the system is unable to detect A_{PK} in these cases. Though a small k produces smaller DT, accelerometer and gyroscope lead to nice results in most cases, being DT = 8.32 min in the worst case (BR, w = 10,000 and TH = 10) and DT = 1.32 min in the best one (S, w = 1000 and TH = 5). Indeed, inertial sensors are not significantly affected by the injection type, just around 50 s at most. By contrast, results for location are worse in S injection, the attack can only be detected when k = 3, getting DT = 8.81 min when $FITN$ >5. This sensor type produces acceptable results for small k values, getting DT = 7.27 min in the best case (BR, w = 1000, TH = 5).

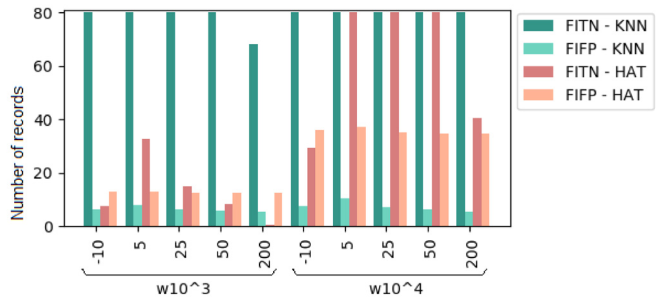
5.4.3.2. Effect of BR injection strategy. The FIFP and FITN for the BR injection strategy are presented in Figs. 5(a) and 5(b). If the right

Table 2
Detection time for k value analysis in minutes.

		Location				Gyroscope and accelerometer					
		S		BR		S			BR		
w	k	TH=5	TH=10	TH=5	TH=10	TH=5	TH=10	TH=20	TH=5	TH=10	TH=20
1000	3	8.81	13.81	7.27	12.27	1.32	2.32	5.32	1.35	2.34	5.34
	10	–	–	13.50	18.50	2.37	3.37	6.36	2.50	3.50	6.50
	21	–	–	22.60	27.60	3.61	4.61	7.61	3,90	4.90	7,90
10,000	3	–	–	7.79	12.79	1.41	2.41	5.41	1.43	2.43	5.43
	10	–	–	15.80	20.80	2.51	3.51	6.50	2.73	3.73	6.73
	21	–	–	27.56	32.56	3.80	4.80	7.80	4.32	5.32	8.32



(a) Location



(b) Gyroscope and accelerometer

Fig. 5. FIFP and FITN for Blind Rate injection.

DSM technique is chosen, injection attacks can be identified even when small injections are carried out. In all cases FIFP is slightly better (thus smaller) when higher injection rates are considered and FITN remains acceptable, though results differ between sensor types. In terms of location, HAT works better in most cases and particularly for $w = 1000$. For instance, using HAT, FIFP is between 3.77 and 2.45 for $w = 1000$ and FITN high enough. By contrast the KNN algorithm leads to much better results for accelerometer and gyroscope data, being FIFP between 5.44 and 6.1 for $w = 1000$ and between 5.51 and 10.22 for $w = 10{,}000$, while FITN remains under sensible limits.

Concerning DT, Table 3 shows the results per algorithm. Though the attack is detected in all cases, there are some challenging situations. Using gyroscope and accelerometer attacks can be detected significantly fast, specially if KNN is applied. In this case w does not affect DT, while it is just slightly affected by injection rates, for example but under a similar reasoning in all cases, DT = 2.46 min for 50% of injection and DT = 2.36 min when injection increases to 200%, TH = 5 and w = 1000. Location provides opposite results, being HAT and $w = 1000$ the most appropriate solution for being DT = 8.26 min and 13.26 min in the worst case and DT = 7.45 min and 12.45 min in the best case for TH = 5 and 10 respectively.

5.4.3.3. Effect of S injection strategy. Having knowledge of the victim and thus, doing S injection would presumably make detection more difficult. Fig. 6 presents results of the analysis. KNN does not properly work for location because FITN is very small (4.8 for $w = 10{,}000$ and 1.13 for w = 1000). Moreover, HAT and w = 1000 seem to be the only alternative for this sensor type leading to FIFP = 7.66 and FITN = 670.79. Gyroscope and accelerometer produce better results for KNN getting FIFP = 5.73 and 6.29 and FITN = 10.79 and 30.12 for $w = 1000$ and $w = 10{,}000$ respectively.

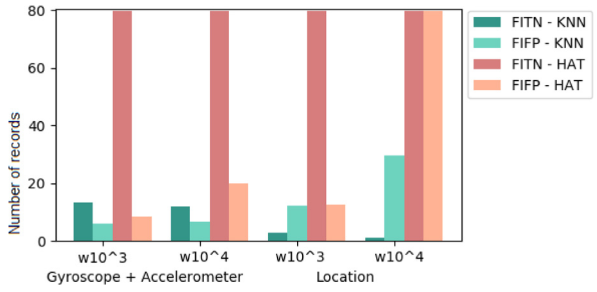


Fig. 6. FIFP and FITN for Slogger injection.

Table 4 depicts DT for S injection and the considered values for TH. Again, there are some cases in which A_{PK} is not detected. For location, the best case is reached with HAT, $w = 1000$ and TH = 5. Even in that case, DT = 12.66 min which is a poor result. By contrast, for gyroscope and accelerometer HAT leads to acceptable results when w = 1000, for instance, DT = 3.76 min for TH = 5. Nonetheless, KNN is the best alternative of this type of sensors, leading to DT = 2.43 min in the best case (TH = 5 and w = 1000) and DT = 6.57 min in the worst one (TH = 20 and w = 10,000).

5.4.4. Summary of the analysis

In light of this study different results are observed. First, injection affects CA systems but the type of injection is a matter of vital concern, together with the type of sensorial data at stake. Results of injection types and most appropriate algorithms are as follows:

- Blind Rate injection: injection rates do not significantly affect results, though they differ between sensor data types. In terms of location, HAT with w = 1000 is the best choice as the detection takes around 7–9 min if TH = 5. According to the proposed use case, an attacker would have enough time to use common applications and even to learn how to use an unknown one, but the system would be useful as DT is less than that of the victim to notice the loss of his/ her device. On the contrary, if using gyroscope and accelerometer data KNN is preferable, as DT is between 2.36 and 4.56 min when TH<20 and between 8.14 and 13.84 if TH = 20. In this situation the attacker has time to do common activities, for instance, in the best case (DT = 2.36) 9 quick interactions could be carried out, or an average use of social networking applications, though the attacker would still be detected before the victim realizes (recall Section 4).
- Slogger injection: the best results are achieved for gyroscope and accelerometer when using KNN and TH is the smallest (1.32 min). Indeed, this type of attack could be detected in a few minutes (between 2 and 4). However, A_{PK} cannot be detected using KNN when using location. HAT and w = 1000 is the only choice for this sensor data, though there is room for significant improvement because more than 12 min are required to detect an attack in the best studied case. For the sake of illustration, the

Table 3
Detection time of Blind Rate injection in minutes.

		Location				Gyroscope and accelerometer					
		KNN		HAT		KNN			HAT		
w	Injection	TH=5	TH=10	TH=5	TH=10	TH=5	TH=10	TH=20	TH=5	TH=10	TH=20
1000	−10%	15.09	20.09	8.26	13.26	2.53	3.53	6.52	4.19	5.19	8.18
	5%	15.87	20.87	8.77	13.77	3.00	4.00	6.99	4.25	5.25	8.24
	25%	14.56	19.56	7.75	12.75	2.56	3.56	6.56	4.14	5.14	8.14
	50%	13.88	18.88	7.55	12.55	2.46	3.46	6.46	4.14	5.14	8.14
	200%	12.87	17.87	7.45	12.45	2.36	3.36	6.36	4.14	5.14	8.14
10,000	−10%	18.81	23.81	39.51	44.51	2.90	3.90	6.89	9.99	10.99	13.99
	5%	21.39	26.39	39.25	44.25	3.56	4.56	7.55	10.34	11.34	14.34
	25%	16.92	21.92	18.84	23.84	2.74	3.74	6.74	9.81	10.81	13.81
	50%	15.02	20.02	11.91	16.91	2.57	3.57	6.56	9.73	10.73	13.73
	200%	13.12	18.12	10.67	15.67	2.38	3.38	6.38	9.73	10.73	13.73

Table 4
Detection time of Slogger injection in minutes.

		KNN			HAT		
	w	TH=5	TH=10	TH=20	TH=5	TH=10	TH=20
Gyr. & Acc.	1000	2.43	3.43	6.43	3.76	4.76	7.76
	10,000	2.57	3.57	6.57	9.17	10.17	13.17
Loc	1000	–	–	■	12.66	17.66	■
	10,000	–	–	■	99.75	104.75	■

Table 5
Related work comparison. √ means addressed.

	RQ1	RQ2	RQ3	Sensors	Algorithms	# injection types	Dataset
[9]	–	–	–	Gyroscope and magnetometer	KNN, SVM, Tree, Regression, Ensemble	2	5 devices, 500 traces each one (duration not specified)
[15]	–	–	–	Accelerometer and gyroscope	Random forest	1	1200 key presses and each key 20 presses
[8]	–	–	–	accelerometer, gyroscope, light sensor, proximity sensor, location, audio, camera and headphone	Markov chain, Naive bayes and PART, Logistic Function, J48, LMT, Hoeffding Tree, and Multilayer Perception	1	50 users, 25 h for each sensor data
[35]	–	–	–	Audio	Proprietary development	1	1 device
[36]	–	–	–	Audio	Neuronal networks	1	2 devices
[17]	√	–	–	Gyroscope and accelerometer	SVM, Regression	1	21 users, 2 sessions
Our proposal	√	√	√	Gyroscope, accelerometer and location	DSM KNN, DSM HAT-ADWIN	2	47 users, 833.3h for location/ 208.3h for gyracc

attacker has time to do 48 quick interactions; or to learn how to use an unknown e-mail application (e.g., SolMail), carry out an average usage session and also doing 24 quick interactions (recall Section 4). Nonetheless, considering 15 min the maximum for a victim to identify the loss of his/her device, DT is within the limits.

In sum, the results show that gyroscope and accelerometer are not significantly affected by the injection type and then, having knowledge on the victim does not affect the detection. Nonetheless and in line with expectations, location data works differently, as detection is faster when the victim is unknown for the attacker (BR injection).

Second, the collection rate may affect detection. On the one hand, location collection rate is 4 times lower than gyroscope and accelerometer, and it inherently affects DT. On the other hand, users may change location less frequently than moving the smartphone while using it. Thus, even if location data were collected at a higher rate (e.g., every second), results could be similar to the one achieved in this analysis.

Third, concerning parameter k of KNN, this study shows that smaller k values (e.g. 3) lead to better results. For larger values like 10 or 21, results are reasonable for gyroscope and accelerometer data but unacceptable in most cases for location data.

Fourth, DSM techniques have been shown to be suitable as a basis for CA systems. However, the value of w affects the system performance. In general, the use of a smaller w (e.g. 1000) produces better results, as previous results suggested [20]. This is an intrinsic property of how DSM works, the system stores less users' information in memory and thus when the adversary A appears, the detection can be faster. An additional benefit is that smaller w values are better in terms of storage because less data has to be kept in memory for the system to work properly.

Finally, it is important to emphasize that sensors should not be considered as trusted sources. An IoT device, like a smartphone, can be compromised and its sensors attacked. To solve this problem the design of the system should consider this attack from the very beginning. For instance, in the design of CA systems leveraging sensorial data through DSM, the management of TH, the choice of the DSM technique and its configuration might have a dramatic impact on the burden of injection attacks.

Table 6
Applied features.

Location	
location_spatio_5means [int]	The cluster ID of the device's location, from a 5-means clustering of the longitude and latitude (received from the Google Play Location API).
location_spatio_10means [int]	The cluster ID of the device's location, from a 10-means clustering of the longitude and latitude (received from the Google Play Location API).
location_spatio_25means [int]	The cluster ID of the device's location, from a 25-means clustering of the longitude and latitude (received from the Google Play Location API).
location_spatio_50means [int]	The cluster ID of the device's location, from a 50-means clustering of the longitude and latitude (received from the Google Play Location API).
location_spatio_100means [int]	The cluster ID of the device's location, from a 100-means clustering of the longitude and latitude (received from the Google Play Location API).
Gyroscope and accelerometer	
accelerometerstat_x_fourth_idx_fft [int]	The index (frequency) of the FFT with the fourth most energy on the accelerometer x-axis.
accelerometerstat_y_first_idx_fft [int]	The index to the component (frequency) of the FFT with the most energy on the accelerometer y-axis.
accelerometerstat_y_fourth_idx_fft [int]	The index (frequency) of the FFT with the fourth most energy on the accelerometer y-axis.
accelerometerstat_y_mean_fft [float]	The average energy across the FFT components on the accelerometer y-axis.
accelerometerstat_y_second_idx_fft [int]	The index (frequency) of the FFT with the second most energy on the accelerometer y-axis.
accelerometerstat_y_third_idx_fft [int]	The index (frequency) of the FFT with the third most energy on the accelerometer y-axis.
accelerometerstat_y_var_fft [float]	The variance of the FFT values obtained from accelerometer y-axis frequencies.
accelerometerstat_z_dc_fft [float]	The DC component of the FFT on the accelerometer z-axis.
accelerometerstat_z_fourth_idx_fft [int]	The index (frequency) of the FFT with the fourth most energy on the accelerometer z-axis.
accelerometerstat_z_fourth_val_fft [float]	The energy of the fourth strongest FFT component on the accelerometer z-axis.
accelerometerstat_z_mean [float]	The average acceleration across the sampled accelerometer z-axis values.
accelerometerstat_z_mean_fft [float]	The average energy across the FFT components on the accelerometer z-axis.
accelerometerstat_z_median_fft [float]	The median FFT value (energy) across the z-axis frequencies.
accelerometerstat_z_second_idx_fft [int]	The index (frequency) of the FFT with the second most energy on the accelerometer z-axis.
accelerometerstat_z_third_idx_fft [int]	The index (frequency) of the FFT with the third most energy on the accelerometer z-axis.
accelerometerstat_z_var_fft [float]	The variance of the FFT values obtained from accelerometer z-axis frequencies.
accelerometerstat_cov_y_x [float]	The y-x covariance of the sampled accelerometer values.
gyroscopestat_x_mean [float]	The average acceleration across the sampled gyroscope x-axis values.
gyroscopestat_x_median [float]	The median acceleration across the sampled gyroscope x-axis values.
gyroscopestat_y_dc_fft [float]	The DC component of the FFT on the gyroscope y-axis.
gyroscopestat_y_mean [float]	The average acceleration across the sampled gyroscope y-axis values.
gyroscopestat_y_median [float]	The median acceleration across the sampled gyroscope y-axis values.
gyroscopestat_z_dc_fft [float]	The DC component of the FFT on the gyroscope z-axis.
gyroscopestat_z_mean [float]	The average acceleration across the sampled gyroscope z-axis values.
gyroscopestat_z_median [float]	The median acceleration across the sampled gyroscope z-axis values.

6. Related work

The existence and detection of data injection attacks has been studied for years. Given that sensors are at stake in a vast array of settings, it has attracted research interest in a variety of scenarios, such as power grids [37], Micro Electro-Mechanical Systems (MEMS) [38], generic sensor-equipped systems [24], cyber–physical systems [39] or internet of things [11].

In what comes to resource-constrained devices such as smartphones, several works focus on the identification of these attacks in sensor networks. Some of them apply authentication schemes to help in injection detection [40,41]. In [42] a hash chain of authentication keys for each node is used to detect malicious nodes injecting data. In [43] keys are bound to geographical locations of sensors to verify the legitimacy of data. By contrast, [44] presents a statistical en-route filtering mechanism which uses message authentication codes attached to sensor data. More general techniques are mentioned in [45], which describes the use of anomaly detection and trust management techniques for detecting malicious sensor nodes. Similarly, [46] presents an Adaptive algorithm to identify compromised nodes by anticipating the right measurement in case of a false data injection attack. By contrast, [47] studies the amount of data that an attacker can inject in a sensor network without being detected, but it is only analysed from a theoretical perspective, so it cannot be regarded as a practical approach.

Focusing on smartphones, [15] proposes leveraging additional sensor readings while users touch screens to prevent from touchstroke leakage. An intrusion detection system based on sensor changes is introduced in [8]. From the perspective of the signals emitted by the device itself, [35] calculates the security level of real systems doing injection attacks, in this case, in the smartphone microphone. [9] focused on how to detect spoofing attacks against the magnetometer and the gyroscope. For this purpose, they combine batch learning artificial intelligence techniques with sensor fusion in order to distinguish between attack and no-attack situations apart. [36] presents a spoof-resistant sensor-based device fingerprinting method to address the challenge of getting strong mobile device authentication. However, an upcoming trend in authentication is the CA variant in which decisions are taken on a rolling basis. In this scenario, [17] is the only work that points out the impact of sensor injection in CA schemes. It shows that injection can affect CA systems accuracy, as well as it proposes the use of sensor data injection as a countermeasure against privacy attacks.

Table 5 compares proposals which deal with sensor data injection attacks on smartphones. It is analysed, per proposal, the satisfaction of established research questions (recall Section 1), sensors involved, algorithms, number of injection types and datasets applied in the evaluation process. First of all, our approach is the only one that addresses proposed research questions. Indeed, just [17] tries to address RQ1. In relation to RQ1, [36] focuses on authentication (no CA) without being directly interested in the analysis of injection attacks but in their prevention. In terms of sensors, the use of gyroscope and accelerometer are a common practice [8,9,15,17] because they lead to high accuracy rates for authentication purposes [48,49]. Audio sensor is also applied in several proposals [8,35,36] and just [8] works with location, in spite of being commonly use for authentication purposes in general [50] and also considered for CA [51,52]. Regarding algorithms, the presented proposal differs in the use of DSM techniques, which are really appropriate in a CA system [21] (recall Section 2). The type of injection is barely described in most of the cases, just [15] proposes the technique that we called "Slogger injection" in this paper and [9] presents a pair of attacks, one to replace sensor data with arbitrary corrupted values and other to do the replacement with chosen values, but no concrete details about the injection procedure are outlined. Finally, our dataset is the biggest one in terms of size, which is specially appropriate when evaluating a CA system where continuous flows of data should be tested.

7. Conclusion and future research issues

The use of Internet of Things (IoT) devices and smartphones in particular is becoming more demanding day by day. Given such extensive use, these devices are becoming useful tools for CA purposes, in which, the identity of a user is periodically verified reducing impersonation possibilities. However, the management of injection attacks in this scenario has been barely addressed and no previous work has jointly focused on the timeliness and effectiveness of this process. This paper presents an analysis of CA systems leveraging DSM techniques, using location, gyroscope and accelerometer sensor data, under the existence of different types of injection attacks. Results show that the type of sensor data, the CA algorithm and some relevant parameters have a dramatic impact on detection accuracy and immediacy. Thus, while in many cases attackers can be identified within a few minutes (between 2 and 4) using gyroscope and accelerometer data, location requires around 8 min in the best case.

As for future directions four different approaches are suggested.

(1) The resiliency analysis of injection in more and different features is an interesting open issue, for instance the use of audio sensors could be a challenging line of research.
(2) The development of a dataset including real injection attacks would help in the study of injection management to encourage future research. Indeed, the different types of devices and user activities while using the smartphone should be taken into account for the dataset generation.
(3) A smart city is full of smart devices, like smart thermometers [53], which are equipped with sensors that could be attacked. Novel scenarios should be devised and tested against injection attacks in this context.
(4) The types of injection attacks should be linked to each particular scenario and threat model. An analysis of types of injection attacks in different scenarios, for instance, in a smart city, would help in the study of their impact.

CRediT authorship contribution statement

Lorena Gonzalez-Manzano: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Software, Validation, Writing - original draft. **Upal Mahbub:** Conceptualization, Investigation, Methodology, Supervision, Writing - review & editing. **Jose M. de Fuentes:** Conceptualization, Investigation, Visualization, Methodology, Supervision, Validation, Writing - review & editing. **Rama Chellappa:** Conceptualization, Methodology, Supervision, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the Spanish Ministry of Science, Innovation and Universities grants TIN2016-79095-C2-2-R (SMOG-DEV), PID2019-111429RBC21(ODIO); by Comunidad de Madrid (CAM) grant P2018/TCS4566 (CYNAMON-CM) funded with European FEDER funds; and CAVTIONS-CM-UC3M funded by UC3M and CAM.

Appendix

Table 6 presents features applied for location and gyroscope and accelerometer data.

References

[1] A. Solanas, C. Patsakis, M. Conti, I.S. Vlachos, V. Ramos, F. Falcone, O. Postolache, P.A. Pérez-Martínez, R. Di Pietro, D.N. Perrea, et al., Smart health: a context-aware health paradigm within smart cities, IEEE Commun. Mag. 52 (8) (2014) 74–81.
[2] M. Alizadeh, S. Abolfazli, M. Zamani, S. Baharun, K. Sakurai, Authentication in mobile cloud computing: A survey, J. Netw. Comput. Appl. 61 (2016) 59–80.
[3] Z. Sitová, J. Šeděnka, Q. Yang, G. Peng, G. Zhou, P. Gasti, K.S. Balagani, HMOG: New behavioral biometric features for continuous authentication of smartphone users, IEEE Trans. Inf. Forensics Secur. 11 (5) (2016) 877–892.
[4] L. Fridman, S. Weber, R. Greenstadt, M. Kam, Active authentication on mobile devices via stylometry, application usage, web browsing, and GPS location, IEEE Syst. J. 11 (2) (2017) 513–521.
[5] U. Mahbub, S. Sarkar, V.M. Patel, R. Chellappa, Active user authentication for smartphones: A challenge data set and benchmark results, in: 2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems, BTAS, IEEE, 2016, pp. 1–8.
[6] Y. Liu, P. Ning, M.K. Reiter, False data injection attacks against state estimation in electric power grids, ACM Trans. Inf. Syst. Secur. 14 (1) (2011) 13.
[7] L. Hu, Z. Wang, Q.-L. Han, X. Liu, State estimation under false data injection attacks: Security analysis and system protection, Automatica 87 (2018) 176–183.
[8] A.K. Sikder, H. Aksu, A.S. Uluagac, 6thsense: A context-aware sensor-based attack detector for smart devices, in: 26th $\{USENIX\}$ Security Symposium, $\{USENIX\}$ Security 17, 2017, pp. 397–414.
[9] K.S. Tharayil, B. Farshteindiker, S. Eyal, N. Hasidim, R. Hershkovitz, S. Houri, I. Yoffe, M. Oren, Y. Oren, Sensor defense in-software (SDI): Practical software based detection of spoofing attacks on position sensor, 2019, arXiv preprint arXiv:1905.04691.
[10] S. Maruyama, S. Wakabayashi, T. Mori, Tap'n ghost: A compilation of novel attack techniques against smartphone touchscreens, in: 2019 IEEE Symposium on Security and Privacy, SP, IEEE, 2019, pp. 620–637.

[11] M. Mohamed, B. Shrestha, N. Saxena, Smashed: Sniffing and manipulating android sensor data for offensive purposes, IEEE Trans. Inf. Forensics Secur. 12 (4) (2016) 901–913.

[12] A.K. Sikder, G. Petracca, H. Aksu, T. Jaeger, A.S. Uluagac, A survey on sensor-based threats to internet-of-things (iot) devices and applications, 2018, arXiv preprint arXiv:1802.02041.

[13] T. Giannetsos, T. Dimitriou, Spy-Sense: spyware tool for executing stealthy exploits against sensor networks, in: Proceedings of the 2nd ACM Workshop on Hot Topics on Wireless Network Security and Privacy, 2013, pp. 7–12.

[14] F. Wei, Y. Li, S. Roy, X. Ou, W. Zhou, Deep ground truth analysis of current android malware, in: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2017, pp. 252–276.

[15] P. Shrestha, M. Mohamed, N. Saxena, Slogger: Smashing motion-based touchstroke logging with transparent system noise, in: Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks, ACM, 2016, pp. 67–77.

[16] M.H. Au, K. Liang, J.K. Liu, R. Lu, J. Ning, Privacy-preserving personal data operation on mobile cloud—Chances and challenges over advanced persistent threat, Future Gener. Comput. Syst. 79 (2018) 337–349.

[17] R. Matovu, A. Serwadda, D. Irakiza, I. Griswold-Steiner, Jekyll and Hyde: On the double-faced nature of smart-phone sensor noise injection, in: 2018 International Conference of the Biometrics Special Interest Group, BIOSIG, IEEE, 2018, pp. 1–6.

[18] A. Bifet, R. Kirkby, Data Stream Mining a Practical Approach, Citeseer, 2009.

[19] M.H. ur Rehman, C.S. Liew, T.Y. Wah, M.K. Khan, Towards next-generation heterogeneous mobile data stream mining applications: Opportunities, challenges, and future research directions, J. Netw. Comput. Appl. 79 (2017) 1–24.

[20] J. de Fuentes, L. Gonzalez-Manzano, A. Ribagorda, Secure and usable user-in-a-context continuous authentication in smartphones leveraging non-assisted sensors, Sensors 18 (4) (2018) 1219.

[21] L. Gonzalez-Manzano, J.M.D. Fuentes, A. Ribagorda, Leveraging user-related internet of things for continuous authentication: A survey, ACM Comput. Surv. 52 (3) (2019) 53.

[22] M.M. Gaber, A. Zaslavsky, S. Krishnaswamy, Data stream mining, in: Data Mining and Knowledge Discovery Handbook, Springer, 2009, pp. 759–787.

[23] J.J. Tsai, S.Y. Philip, Machine Learning in Cyber Trust: Security, Privacy, and Reliability, Springer Science & Business Media, 2009.

[24] Y. Zhang, K. Rasmussen, Detection of electromagnetic interference attacks on sensor systems, in: IEEE Symposium on Security and Privacy, S&P2020, 2020.

[25] A. Das, N. Borisov, M. Caesar, Tracking Mobile Web Users Through Motion Sensors: Attacks and Defenses, NDSS, 2016.

[26] E. Bertini, T. Catarci, A. Dix, S. Gabrielli, S. Kimani, G. Santucci, Appropriating heuristic evaluation for mobile computing, Int. J. Mob. Hum. Comput. Interact. (IJMHCI) 1 (1) (2009) 20–41.

[27] D. Ferreira, J. Goncalves, V. Kostakos, L. Barkhuus, A.K. Dey, Contextual experience sampling of mobile application micro-usage, in: Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices & Services, 2014, pp. 91–100.

[28] J.P. Carrascal, K. Church, An in-situ study of mobile app & mobile search interactions, in: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, 2015, pp. 2739–2748.

[29] D. Reading, National survey finds 1 in 3 mobile phone owners would know they've lost their phone within 15 minutes, 2011.

[30] Y. Mirsky, A. Shabtai, L. Rokach, B. Shapira, Y. Elovici, Sherlock vs moriarty: A smartphone dataset for cybersecurity research, in: Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security, ACM, 2016, pp. 1–12.

[31] J. Hu, D. Gingrich, A. Sentosa, A k-nearest neighbor approach for user authentication through biometric keystroke dynamics, in: 2008 IEEE International Conference on Communications, IEEE, 2008, pp. 1556–1560.

[32] S. Choi, I.-H. Youn, R. LeMay, S. Burns, J.-H. Youn, Biometric gait recognition based on wireless acceleration sensor using k-nearest neighbor classification, in: 2014 International Conference on Computing, Networking and Communications, ICNC, IEEE, 2014, pp. 1091–1095.

[33] Y. Zhong, Y. Deng, A.K. Jain, Keystroke dynamics for user authentication, in: 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, IEEE, 2012, pp. 117–123.

[34] R.S. Zack, C.C. Tappert, S.-H. Cha, Performance of a long-text-input keystroke biometric authentication system using an improved k-nearest-neighbor classification method, in: 2010 Fourth IEEE International Conference on Biometrics: Theory, Applications and Systems, BTAS, IEEE, 2010, pp. 1–6.

[35] I. Giechaskiel, Y. Zhang, K.B. Rasmussen, A framework for evaluating security in the presence of signal injection attacks, 2019, arXiv preprint arXiv:1901.03675.

[36] Y. Lee, J. Li, Y. Kim, MicPrint: acoustic sensor fingerprinting for spoof-resistant mobile device authentication, in: Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, 2019, pp. 248–257.

[37] L. Xie, Y. Mo, B. Sinopoli, False data injection attacks in electricity markets, in: 2010 First IEEE International Conference on Smart Grid Communications, IEEE, 2010, pp. 226–231.

[38] T. Trippel, O. Weisse, W. Xu, P. Honeyman, K. Fu, WALNUT: Waging doubt on the integrity of mems accelerometers with acoustic injection attacks, in: 2017 IEEE European Symposium on Security and Privacy, EuroS&P, IEEE, 2017, pp. 3–18.

[39] A. Chattopadhyay, U. Mitra, Security against false data injection attack in cyber-physical systems, 2018, CoRR abs/1807.11624.

[40] R. Lu, X. Lin, H. Zhu, X. Liang, X. Shen, BECAN: A bandwidth-efficient cooperative authentication scheme for filtering injected false data in wireless sensor networks, IEEE Trans. Parallel Distrib. Syst. 23 (1) (2012) 32–43.

[41] S. Zhu, S. Setia, S. Jajodia, P. Ning, An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks, in: Null, IEEE, 2004, p. 259.

[42] Z. Yu, Y. Guan, A dynamic en-route scheme for filtering false data injection in wireless sensor networks, in: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, ACM, 2005, pp. 294–295.

[43] J. Wang, Z. Liu, S. Zhang, X. Zhang, Defending collaborative false data injection attacks in wireless sensor networks, Inform. Sci. 254 (2014) 39–53.

[44] F. Ye, H. Luo, S. Lu, L. Zhang, Statistical en-route filtering of injected false data in sensor networks, IEEE J. Sel. Areas Commun. 23 (4) (2005) 839–850.

[45] V.P. Illiano, E.C. Lupu, Detecting malicious data injections in wireless sensor networks: A survey, ACM Comput. Surv. 48 (2) (2015) 24.

[46] Y. Hua, F. Chen, S. Deng, S. Duan, L. Wang, Secure distributed estimation against false data injection attack, Inform. Sci. 515 (2020) 248–262.

[47] Y. Mo, E. Garone, A. Casavola, B. Sinopoli, False data injection attacks against state estimation in wireless sensor networks, in: 49th IEEE Conference on Decision and Control, CDC, IEEE, 2010, pp. 5967–5972.

[48] D. Crouse, H. Han, D. Chandra, B. Barbello, A.K. Jain, Continuous authentication of mobile user: Fusion of face image and inertial measurement unit data, in: 2015 International Conference on Biometrics, ICB, IEEE, 2015, pp. 135–142.

[49] R. Murmuria, A. Stavrou, D. Barbará, D. Fleck, Continuous authentication on mobile devices using power consumption, touch gestures and physical movement of users, in: International Symposium on Recent Advances in Intrusion Detection, Springer, 2015, pp. 405–424.

[50] F. Zhang, A. Kondoro, S. Muftic, Location-based authentication and authorization using smart phones, in: 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, IEEE, 2012, pp. 1285–1292.

[51] L. Fridman, S. Weber, R. Greenstadt, M. Kam, Active authentication on mobile devices via stylometry, application usage, web browsing, and GPS location, IEEE Syst. J. 11 (2) (2016) 513–521.

[52] H. Saevanee, N. Clarke, S. Furnell, V. Biscione, Text-based active authentication for mobile devices, in: IFIP International Information Security Conference, Springer, 2014, pp. 99–112.

[53] H. Habibzadeh, Z. Qin, T. Soyata, B. Kantarci, Large-scale distributed dedicated- and non-dedicated smart city sensing systems, IEEE Sens. J. 17 (23) (2017) 7649–7658.