J Netw Syst Manage (2012) 20:561–578
DOI 10.1007/s10922-012-9246-0

Online Randomization Strategies to Obfuscate User Behavioral Patterns

Juan E. Tapiador · Julio C. Hernandez-Castro · Pedro Peris-Lopez

Received: 15 June 2011 / Revised: 21 June 2012 / Accepted: 23 June 2012 /
Published online: 7 July 2012

Abstract When operating from the cloud, traces of user activities and behavioral patterns are accessible to anyone with enough privileges within the system. This could be, for example, the case of dishonest technical staff who may well be interested in selling user logs to competitors. In this paper, we investigate some of the security and privacy leakages derived from the analysis of user activities. We show that the working behavioral patterns exhibited by users can be easily captured into computationally useful representations that would allow an adversary to predict future activities, detect the occurrence of events of interest, or infer the organization's internal structure. We then introduce the idea of obfuscating user behaviour through Online Action Randomization Algorithms. In doing so, we introduce an indistinguishability-based definition for perfectly obfuscated actions and a concrete scheme to randomize user traces in an incremental way. We report experimental results confirming the obfuscation quality and other properties of the proposed schemes.

J. E. Tapiador (✉)
Department of Computer Science, University of York, Deramore Lane, York YO10 5GH, UK
e-mail: jet@cs.york.ac.uk; jestevez@inf.uc3m.es

Present Address:
J. E. Tapiador
Deptartment of Computer Science, Universidad Carlos III de Madrid, Avda. Universidad 30, 28991 Leganes, Madrid, Spain

J. C. Hernandez-Castro
School of Computing, University of Portsmouth, Buckingham Building, Lion Terrace, Portsmouth PO1 3HE, UK
e-mail: Julio.Hernandez-Castro@port.ac.uk

P. Peris-Lopez
Information Security and Privacy Lab, Delft University of Technology (TU-Delft), P.O. Box 5031, 2600 GA Delft, The Netherlands
e-mail: P.PerisLopez@tudelft.nl

Keywords Cloud computing security · Insider threats · User modeling · Anonymity · Privacy

1 Introduction

One major danger of externalizing an organization's computing activities is the potential loss of some of the security and privacy properties intrinsically provided by an owned, perimetered network. Operating from the cloud is undoubtedly very attractive from an economic perspective due to its flexibility, technical advantages, and cost efficiency, but it also entails a number of security concerns. It has been pointed out that many such security issues are essentially old problems in a new setting, although they may be more acute [1]. This is precisely the case of the so-called insider threats, i.e., security risks posed by authorized users who enjoy enough privileges to compromise some security properties of the organization's assets. Yet in a cloud computing environment, the concept of 'insider' acquires a renewed relevance, since the technical staff employed by the cloud operator may enjoy the same, or even more privileges, than any truly internal user. In fact, the malicious insider problem is systematically found in the top seven security threats of cloud computing published by the Cloud Security Alliance [2], as many providers do not reveal how they hire staff, how they grant them access to computing assets, or how they monitor them.

These and other concerns have encouraged organizations to heavily rely on different cryptographic primitives to securely store and process data in the cloud. Appropriate use of these techniques can minimize the risk exposure to, for example, privacy leakages, integrity breaches, or data theft. Yet what these techniques cannot offer so far is protection against an insider attempting to gain information about the organization's assets and ongoing operations by simply observing user behavior. The situation is somewhat similar to traffic analysis attacks in communication networks [3], where the mere observation of traffic flows (or the lack of them) enables an observer to infer sensitive information about what it is going on.

Ideally, users should be provided with guarantees that their logs and traces are conveniently anonymized or, at least, that they are processed in a privacy-preserving way [4]. Work in privacy-preserving data processing has proliferated over the last decade in many domains, and the cloud computing setting poses some interesting challenges. In our opinion, cloud users must be aware that any activity performed in the cloud can be potentially monitored, recorded, and analyzed by an attacker with enough privileges in the cloud infrastructure. Consequently, not only data but also *actions* (or the lack of them) must be conveniently processed in order to preserve some privacy properties.

1.1 Contributions

In this work, we investigate some of the security and privacy threats posed by any passive observer with access to users' activities on a computing infrastructure. As discussed above, this is the case of cloud operators and others insiders with enough privileges to monitor users accessing the cloud remotely, either in real time or offline, through logs and other historical records.

In particular, we make the following contributions:

1. We show through the analysis of real-world user traces that the behavioral patterns exhibited by users can be easily captured into a computationally useful representation by anyone with access to the raw data.
2. We argue that these models can be subsequently used to carry out activities that violate some security and privacy properties of the organization. This includes predicting future activities to be executed by users, even in the absence of complete information; detecting unusual work patterns that in turn could reveal the occurrence of events of interest; or deducing the organization's internal structure by inferring the role played by different users. We evaluate and discuss some of these attacks.
3. We describe a potential countermeasure based on obfuscating user activities to prevent profiling attacks. To that end, we introduce an indistinguishability-based definition for perfectly obfuscated actions and a concrete scheme to randomize user traces in an incremental way.
4. We propose and explore the performance of four different heuristic procedures to randomize users activities. All of them constitute practical schemes that could be deployed with little extra cost on the organization. We discuss their obfuscation quality and other properties related to the influence on the adversary's perceived state.

2 Related Work

Security and privacy concerns constitute strong obstacles to wide adoption of cloud computing. Most classic security problems translate easily into equivalent, or even more acute, threats in the cloud domain, while others become particularly worrisome. Thus, for example, recent research has shown that multitenancy opens some doors for potential privacy leaks through traffic patterns and other side channels, making it possible for an attacker to infer private information about a victim whose virtualized application is on the same physical machine [5].

In most current cloud computing frameworks, users do not have access to the cloud's internal operational details. Consequently, both users' data and activities could be examined and recorded without the users' consent and, more importantly, in an undetectable way. Even if the cloud service provider is trusted in the sense that it guarantees the privacy of its users, it is unclear whether a malicious insider will do the same. Apart from stealing valuable data stored in the cloud, the traces of activity left by users contain sensitive information about the users' identities and preferences, and the role played in the organization, etc. Several decades of research in user modeling techniques have revealed that users tend to display patterns of actions throughout their normal working activities [6]. Such patterns are extremely regular and often have very low complexity. Thus, for example, experiments conducted by Davison and Hirsh [7, 8] showed that in most cases, the last two commands executed by a user suffice to predict the next action with an accuracy of around 38 %.

In a recent experiment, the MIT Reality Mining Project [9] provided further evidence about the risks of user profiling. In this case, the location and communication activities of 100 users chosen among students and staff were monitored during several days. The logs were used to construct classifiers that revealed huge amounts of sensitive information about each user's preferences, intentions, relations with others, etc. For example, it was found that staff and students have very different behavioral patterns, so it is easy to classify them. Furthermore, only a few parameters could serve to guess the relationship between two users, and predicting future actions and locations is extremely easy in almost all cases.

The interested reader can find in [10] a more detailed discussion about challenges in cloud computing security and privacy.

3 Background

3.1 Definitions and Notation

We first introduce and discuss some definitions and notation that will be extensively used throughout this paper.

Definition 1 (*Actions*) For each system user U, let $\texttt{Act(U)} = \{\mathtt{a}_1, \ldots, \mathtt{a}_K\}$ be the set of possible actions available to U. For the purposes of this paper, we will assume each action to be atomic, i.e. with no duration.

This is a quite simplified model of user actions. Nonetheless, for the purposes of this work, we attempt to keep behavioral profiles as simple as possible, and therefore we choose not to use a richer data model, e.g., including action modifiers and attributes. We also will not use a fine-grained data model for actions related to communications. Thus, for example, one action could represent the fact that the user sends an email, but the recipient and other message attributes will be omitted, effectively making all email actions indistinguishable. Later on, we show how such a simplistic model suffices to obtain reliable behavioral profiles of most users, even though we acknowledge that a richer data model could provide us with more interesting information.

Definition 2 (*Trace*) A trace $\texttt{Tr(U)} = \langle \mathtt{o}_1, \ldots, \mathtt{o}_\ell \rangle$, with $o_i \in \texttt{Act(U)}$, is an ordered sequence of actions executed by user U. We assume the order is implicit in the sequence. Thus, o_1 was executed before o_2, which was executed before o_3, and so on. We will denote by $|\texttt{Tr(U)}| = \ell$ the length of $\texttt{Tr(U)}$.

Here we deliberately avoid the use of timestamps and other contextual information associated with each action executed. Even though such information is generally available in most log formats, we will make no use of it in this work. Only the temporal relation among actions will be explicitly used in our model.

Definition 3 (*N-grams*) An N-gram $\xi = \langle o_{i_1,\ldots,i_N} \rangle$ extracted from a trace $\mathtt{Tr(U)} = \langle \mathtt{o}_1, \ldots, \mathtt{o}_\ell \rangle$ is a sequence of N consecutive actions contained in $\mathtt{Tr(U)}$. We will denote by $[\![\mathtt{Tr(U)}]\!]_\mathtt{N} = \{\xi_1, \ldots, \xi_{\mathtt{N}-|\mathtt{Tr(U)}|+1}\}$ the set of all possible N-grams in $\mathtt{Tr(U)}$.

N-grams capture temporal relations and dependencies among actions. As we will show later, the majority of tasks performed by users are composed of sequences of repetitive actions. Such routines can be reliablly represented by a few, highly probable short sequences.

3.2 Evaluation Framework

The proposal and empirical work introduced in this paper will be evaluated using a dataset of user traces introduced by Schonlau et al. (SEA) [11]. The dataset[1] consists of sequences of truncated UNIX commands corresponding to the normal activity of 70 users and collected over a period of several months. These user traces were originally released to provide a framework for the evaluation of different masquerade detection methods. Roughly speaking, a masquerader is an attacker who succeeds in obtaining a legitimate user's credentials and attempts to use the stolen identity to carry out malicious actions (e.g. credit card fraud). Masquerade detection algorithms construct models of user behavior and then measure departures from them to identify if someone other than the authorized user is impersonating him. In the SEA dataset, 50 users were chosen as intrusion targets and the remaining 20 as masqueraders. The dataset provides 15,000 commands for each one of the target users, organized as follows. The first 5,000 commands contain actions actually executed by the user. The remaining 10,000 commands are split in blocks of 100 commands, each block being either self (i.e., corresponding to the user) or non-self. Non-self blocks represent masquerade attempts and contain commands issued by a randomly chosen masquerader user.

As introduced in [11], the main task for a masquerade detection algorithm is to accurately identify non-self blocks as anomalous (and, therefore, implicitly marked as masquerade attempts), while correctly classifying the self blocks as belonging to the user. The work in [11] explores the performance of six different machine learning algorithms for this task. Work on masquerade detection (and, more generally, on profiling user behavior for security purposes) has proliferated over the last decade, especially concerning the study of different detection strategies. Some of the proposals include the use of Bayes classifiers and Support Vector Machines [12–16]; information-theoretic approaches [17, 18]; hidden Markov models [19]; or sequence- and text-mining [20–23] schemes, among others. Despite the diversity of principles behind these methods, the reported results show that they all perform similarly in terms of accuracy.

In this paper, we will first use the SEA dataset to evaluate the extent to which users are generally susceptible to profiling attacks. As we will have no interest in the detection of masquerade attempts, all the data blocks labeled as such in the dataset

[1] Publicly available at http://www.schonlau.net.

will be simply ignored. Subsequently, we will use the same dataset to study the performance of different obfuscation strategies.

4 User Profiling Attacks

In this section, we motivate our work by showing how an adversary, who can observe the activities carried out by a user during a prolonged period of time, is able to obtain a computationally useful profile of his behavior. Such profiles can be used for a number of purposes, including predicting the user's future activities, detecting unusual work patterns, which could reveal the occurrence of events of interest, or inferring users with a similar role in the organization by clustering similar behavioral profiles.

4.1 Experiment 1: Measuring the Complexity of User Behavioural Patterns

One reason why user behaviour can be efficiently profiled into a computationally useful model is its low complexity. This fact has been repeatedly pointed out in works devoted to modeling user activities and reveals the intrinsically repetitive nature of most computer tasks. In order to obtain some numerical estimates of such a phenomenon, we have computed several metrics for each one of the 50 users in the SEA dataset. Figure 1 shows the fraction of used commands and each user's entropy. The former is simply computed by dividing the number of observed commands into 856, which is the total number of actions available. The entropy, on the other hand, is given in relative terms, i.e. divided by the maximum possible attainable entropy:

$$H(\mathtt{Tr(U)}) = -\log_2 856 \sum_{\mathtt{i=1}}^{\mathtt{856}} \mathtt{P(a_i)} \log_2 \mathtt{P(a_i)} \tag{1}$$

where $P(a_i)$ is the probability of action a_i, as computed from $\mathtt{Tr(U)}$.

Both charts confirm the hypothesis that users tend to repeat themselves by using only a reduced set of actions, generally less than 20 % of those available in the system. As a result, user entropy oscillates roughly around 50 % of its maximum and, in some cases, notably lower. Although not shown in this graph, the subset and frequencies of used actions is relatively unique to each user, a fact that facilitates the construction of user-specific profiles and therefore, can serve to distinguish among them.

In order to further explore the low complexity of user behavior, we have computed the number of different N-grams appearing in each user trace. Figure 2 shows the curves obtained for different values of N. We note here that a perfectly random process will yield the maximum (856^N in this case) number of possible N-grams over a statistically significant sample. The case here is quite the contrary. Firstly, for each value of N the number of different N-grams is considerably lower than the maximum; and secondly, the number of different N-grams observed is

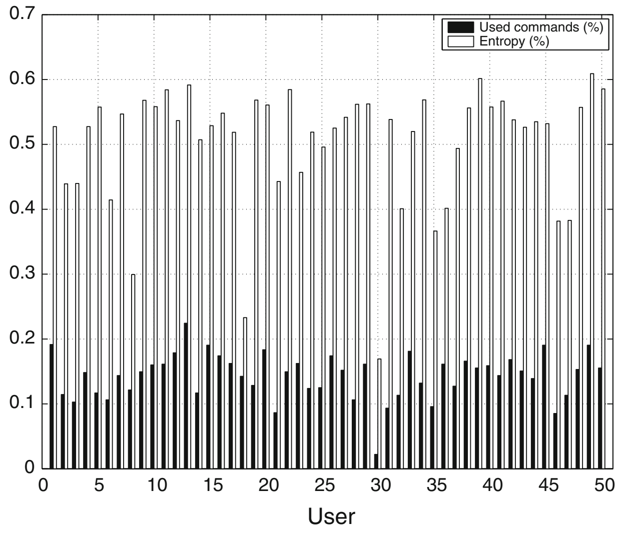

Fig. 1 Relative entropy and fraction of used commands of the 50 users in the SEA dataset

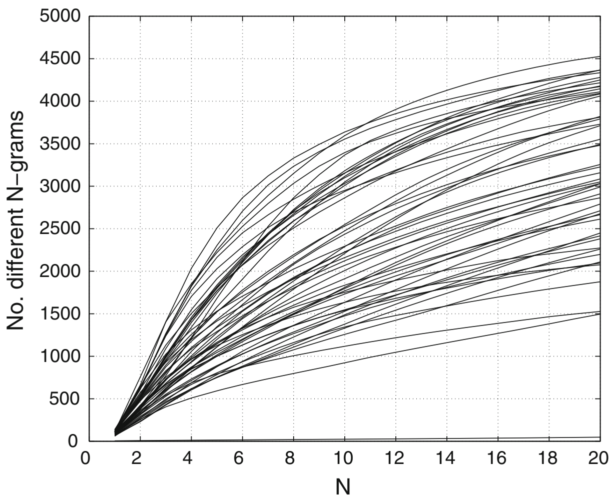

Fig. 2 Number of different *N*-grams found in the user traces contained in the SEA dataset. Each curve corresponds to one of the 50 users

sublinear in *N*, quite different from the exponential behavior expected from a truly random process.

4.2 Experiment 2: User Predictability

The results obtained in the previous section indicate that users exhibit a fairly regular behavior susceptible to modeling. Such models could be used to anticipate, in a probabilistic sense, the actions that users will execute in the near future. Several machine-learning techniques can be applied here, especially those developed in the context of user modeling for security purposes (e.g., [16]) or plan recognition and task prediction [6, 24–27].

To illustrate the extent to which user behavior can be predicted, we conducted a series of experiments based on the use of an incremental probabilistic action modeling (IPAM) algorithm similar to that presented by Davison and Hirsh in [25]. The algorithm relies on a low-order Markov assumption that each action depends only on the few preceding actions. Contrary to the approach followed in [25], where only the last executed action is used to predict the next one, we use a more general algorithm, denoted IPAM-*N*, in which the order *N* of the model will be a user-specified parameter.

The algorithm relies on a data structure called `state` to keep track of the occurrences of all *N*-grams observed so far. The `state` is represented as a tree where each node has degree $|\mathtt{Act(U)}|$. Each node n_i in the tree is associated with one action a_i and maintains a counter initially set to zero. For each node n_i other than the root r, the path $r \rightarrow n_1 \rightarrow \cdots \rightarrow n_i$ represents the *N*-gram $\langle a_1, \ldots, a_i \rangle$. We denote by $\mathtt{state}\langle \mathtt{a}_1, \ldots, \mathtt{a}_\mathtt{i} \rangle$ the counter kept at the last node, indicating how many times the *N*-gram $\langle a_1, \ldots, a_i \rangle$ has been observed so far. This is a very compact and efficient representation of all the *N*-grams seen so far and their frequencies. In fact, nodes can (and should) be added dynamically. As discussed in Fig. 2, not all possible *N*-grams will appear in a user trace, and the actual state will be far smaller than the one theoretically expected.

The two main operations performed over the `state` are:

1. $\mathtt{update}(\mathtt{state}, a_{t-N}, \ldots, a_t)$
   When called after the *t*-th action, this function simply increments $\mathtt{state}\langle \mathtt{a}_{\mathtt{t-N}}, \ldots, \mathtt{a}_\mathtt{t} \rangle$. Note that the arguments are an *N*-gram and the next action.
2. $a_t \leftarrow \mathtt{predict}(\mathtt{state}, a_{t-N}, \ldots, a_{t-1})$
   Given an *N*-gram of previously observed actions, the function returns

$$a_t = \arg\max_{x_t} \mathtt{state}\langle \mathtt{a}_{\mathtt{t-N}}, \ldots, \mathtt{a}_{\mathtt{t-1}}, \mathtt{x}_\mathtt{t} \rangle \tag{2}$$

i.e., the most likely next action according to the observations so far.

We implemented IPAM-*N* and explored its performance on the SEA dataset. For each user trace, we incrementally learn by using the first 5,000 actions, and then use the model to predict the remaining commands in the trace. We deliberately chose to stop the learning process after 5,000 actions, as this proved to be enough to capture the behavioral patterns exhibited by users.

The *prediction accuracy*, defined as the fraction of correctly predicted actions, is shown in the boxplots given in Fig. 3 for different values of *N*. Each boxplot represents the distribution of accuracies computed for the 50 users. In general terms, the prediction error is much lower for 2- and 3-grams than the one obtained by simply relying on the last action. This fact has been reported before in experiments conducted in other contexts (see e.g. [7, 8]). Furthermore, larger values of *N* result in a loss of accuracy, mainly due to the overly complicated model and the spread of probabilities over too many and too long *N*-grams.

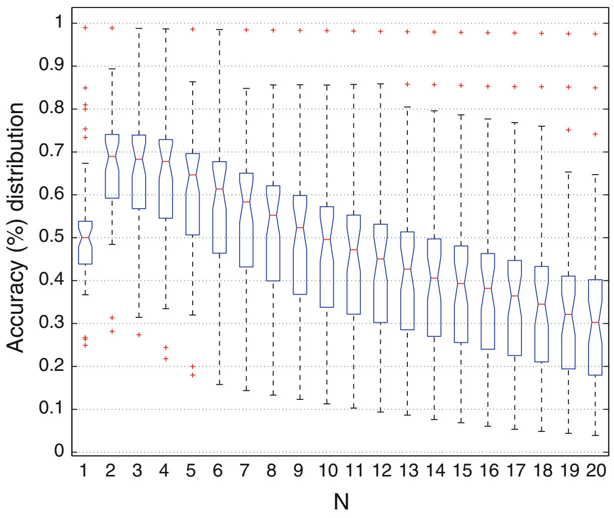

Fig. 3 Distribution of the prediction accuracy for all users and different values of N

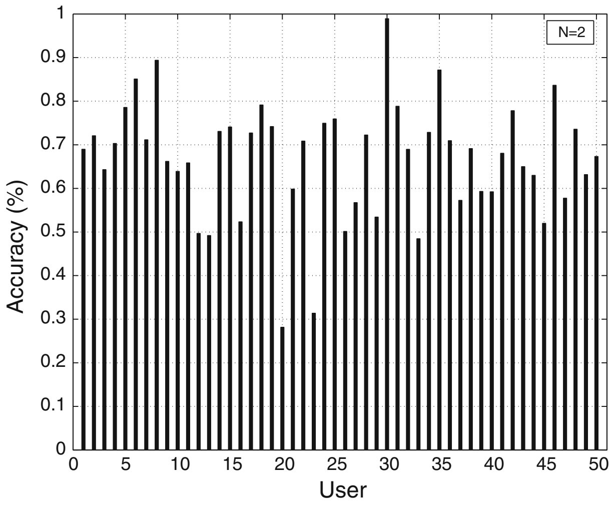

Fig. 4 Accuracy for $N = 2$

For illustration purposes, Fig. 4 shows the prediction accuracy for all users using IPAM-2. In most cases, it is possible to correctly predict the next action with a probability higher than 0.5, which constitutes a very significant gain over the a-priori 1/856 likelihood. The variability among users (prediction-wise) is high, and generally depends on the user role within the organization, the tasks he generally carries out, etc. As discussed before, these and other factors make each user profile relatively unique, a fact that would entail other security risks.

4.3 Further Attacks

In this section, we have given practical evidence about the low complexity of behavioral patterns exhibited by users and their intrinsic susceptibility to modeling. These are by no means the only class of attacks that users under surveillance may experience. As indicated before in this paper, a number of related threats can be demonstrated, including:

- *Clustering users with similar profiles*. Such clusters could represent behaviors exhibited by users who are *functionally* similar, e.g. because they play a similar role within the organization. This information may be useful to an adversary who attempts to reconstruct a picture of the whole structure of the organization and the position of each user within it.
- *Detecting abnormal working patterns*, e.g., by using an anomaly detector. These could indicate a change in the user role (for example, due to an internal promotion) or his assignment to a new task (e.g., a new project).

In this paper, we have omitted results related to these attacks due to space reasons. They will be covered in future work. Nonetheless, the countermeasures we introduce in the next section should suffice to prevent them as well.

5 Obfuscating Behavioral Patterns with Online Action Randomization Algorithms (OARA)

We next introduce and evaluate practical countermeasures to prevent an observer from learning anything useful from a user trace of actions. We will first introduce the problem in more precise terms and then describe and evaluate concrete schemes.

5.1 Online Action Randomization Algorithms

One classical way of reducing the amount of information a process gives to an observer is by obfuscating its observable outcome. Thus, for example, traffic analysis prevention measures rely on primitives such as rerouting network flows through different paths and padding them with dummy traffic (see e.g. [3]). We will make use of an analogous idea here, but first we provide a more formal treatment of the problem.

Definition 4 (*Perfectly obfuscated trace*) A trace $\mathtt{Tr(U)}$ is perfectly obfuscated if there is no algorithm $\mathcal{A}$ such that, for all N-grams $\xi \in [\![\mathtt{Tr(U)}]\!]_{\mathtt{N}}$ and all values of N, $\mathcal{A}(\xi)$ cannot predict the action next to ξ in $\mathtt{Tr(U)}$ with a probability greater than $1/|\mathtt{Act(U)}|$.

Roughly speaking, a perfectly obfuscated trace is one such that at any given point, all future histories of user actions are equally probable. Naturally, any sequence of truly random actions will satisfy this. However, in practice, the problem we deal with is to randomize a sequence of user actions over which we have no

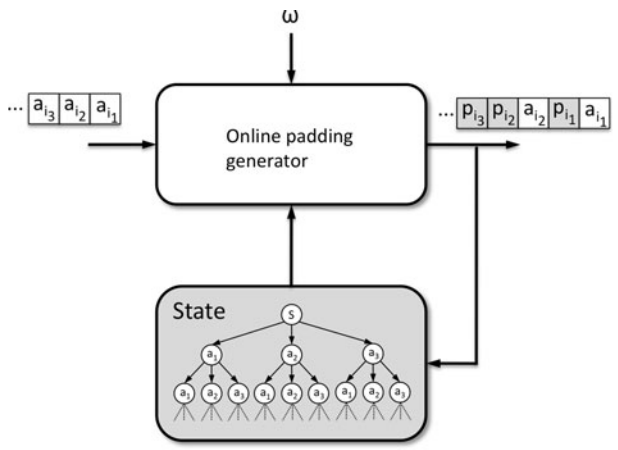
ω
... a_{i_3} a_{i_2} a_{i_1}
Online padding generator
... p_{i_3} p_{i_2} a_{i_2} p_{i_1} a_{i_1}
State
s
a1 a2 a3

Fig. 5 Basic architecture of an Online Action Randomization Algorithm (OARA) controled by a padding intensity parameter ω

influence, i.e., the actions executed by users *must* be executed and in the same order provided.

Our approach consists in adding *padding* to the sequence of user-provided actions. Such padding will consist of dummy actions generated by an automated process, preferably operating in an incremental fashion rather than in offline mode. Ideally the result of such dummy actions (opened windows, text output, etc.) should be filtered out, in such a way that the overall operation of the obfuscation process becomes transparent to the final user. We believe that implementing such a filter is technically feasible, although we do not underestimate the technical complexities involved.

Figure 5 shows the general architecture of an Online Action Randomization Algorithm (OARA). The main building block is a padding generator function that will be responsible of choosing what action(s) should be executed next. The process is controled by a *padding intensity* parameter $\omega \in [0, 1]$ and is invoked after each action provided by the user. The output is given by the following expression:

$$\mathtt{OARA}(\mathtt{a_i}, \mathtt{state}) = \begin{cases} \mathtt{a_i} & \mathtt{with\ prob.}\ 1 - \omega \\ \mathtt{p} \circ \mathtt{a_i} & \mathtt{with\ prob.}\ \omega \end{cases} \qquad (3)$$

where $x \bigcirc y$ indicates concatenation, i.e., action x is executed before action y. The variable `state` will be used by the function to maintain any information necessary to produce the next padding action p. We emphasize that the padding insertion process is performed *locally*, i.e., at the user's terminal before being sent to execution. Thus, the adversary has only access to the (presumably) randomized sequence, but not to the original one.

By following this scheme, a natural consequence of obfuscating a user trace $\mathtt{Tr(U)}$ is that its length will increase to $(1 + \omega)|\mathtt{Tr(U)}|$. We deliberately choose this model as it allows the user to specify the amount of padding that can be afforded. The reason for that is that many cloud computing providers establish a pricing model based on usage time. If we assume that there is a proportional relation between the amount of actions executed by a user and their economic cost, then ω admits a simple interpretation as a budget.

A final but important remark is that we will assume that the entire OARA algorithm is *public*. This implies that the adversary will have access to it, so we cannot rely on trivially breakable schemes such as those based on precomputed tables whose effects the adversary can easily undo.

5.2 Randomization Heuristics

We now describe various heuristic strategies to generate padding actions within an online action randomization algorithm. Even though the overall goal is common to all methods (i.e., to increase the prediction error by introducing some distortion into the perceived behavior), each one attempts to maximize a different property of the internal state maintained by the generation process, and presumably by a learner. For clarity, we group them into two classes:

(i) *Indistinguishability* heuristics that attempt to make the user history indistinguishable from a perfectly random process; and
(ii) *Concept drift* heuristics that attempt to force the learner into learning a concept (i.e., a behavioral pattern) different from the actual one exhibited by the user. Even though these strategies will not satisfy the security notion given in Definition 4, we believe they may be useful in many contexts.

In the following, p_t denotes the action produced by the padding generator when adding padding to action a_t:

1. RAND—Randomly choose one action in $\mathtt{Act(U)}$:

$$p_t \leftarrow_R \mathtt{Act(U)} \tag{4}$$

The goal pursued here is to approximate a flat distribution over the set of possible N-grams $[\![\mathtt{Tr(U)}]\!]_{\mathtt{N}}$, for every N.

2. MAX—Choose the most likely next action for the last observed N-gram:

$$p_t \leftarrow \arg\max_{x_t} \mathtt{state}\langle \mathtt{a_{t-N}}, \ldots, \mathtt{a_{t-1}}, \mathtt{x_t}\rangle \tag{5}$$

No specific shape of the distribution over $[\![\mathtt{Tr(U)}]\!]_{\mathtt{N}}$ is attempted. Rather, by always repeating the so-far most likely next action, the heuristic tries to force the profiling process into learning a wrong behavioral pattern.

3. 2MAX—Choose the second most likely next action for the last observed N-gram:

$$\begin{aligned} m &\leftarrow \arg\max_{x_t} \mathtt{state}\langle \mathtt{a_{t-N}}, \ldots, \mathtt{a_{t-1}}, \mathtt{x_t}\rangle \\ p_t &\leftarrow \arg\max_{x_t} \{\mathtt{state}\langle \mathtt{a_{t-N}}, \ldots, \mathtt{a_{t-1}}, \mathtt{x_t}\rangle \backslash \\ &\qquad \mathtt{state}\langle \mathtt{a_{t-N}}, \ldots, \mathtt{a_{t-1}}, \mathtt{M}\rangle\} \end{aligned} \tag{6}$$

This heuristic can be seen as an intermediate point between the previous ones. By increasing the second most likely next action, the generator attempts to produce a flat distribution, but only among those N-grams seen so far.

4. NZMIN—Choose the most unlikely next action with a strictly positive probability for the last observed N-gram:

$$p_t \leftarrow \arg\min_{x_t}\{\mathtt{state}\langle \mathtt{a}_{\mathtt{t-N}}, \ldots, \mathtt{a}_{\mathtt{t-1}}, \mathtt{x}_{\mathtt{t}}\rangle | \\ \mathtt{state}\langle \mathtt{a}_{\mathtt{t-N}}, \ldots, \mathtt{a}_{\mathtt{t-1}}, \mathtt{x}_{\mathtt{t}}\rangle > 0\} \tag{7}$$

Here we attempt to achieve the same goal as with RAND but but only among those N-grams seen so far.

RAND and NZMIN are indistinguishability heuristics, whereas MAX and 2MAX belong to the concept drift class. In the case of MAX, 2MAX, and NZMIN, the generator must maintain a `state` structure equivalent to that described for the IPAM-N algorithm. The parameter N must be chosen by the user. In general, higher values will obfuscate better, but this will depend on the intrinsic complexity of user behavior.

5.3 Empirical Evaluation I: Obfuscation Metrics

In this section, we report results obtained after the evaluation of the previously described randomization heuristics on the SEA dataset. The experimental methodology is equivalent to that used in Section 4.2: each user's first 5,000 actions are used to train IPAM-N, and the prediction accuracy is computed over the remaining elements in the trace. Here, however, we provide each user with an online action randomization algorithm, so the learner observes 'actual' actions interleaved with padding.

We first evaluate the amount of *obfuscation* introduced by the OARA, which we measure as the reduction in prediction accuracy. Thus, a higher obfuscation value translates into a more successful strategy to avoid profiling. Figure 6 shows the distribution of obfuscation obtained for the 50 users with the 4 randomization heuristics and different values of the padding intensity ω. These plots correspond to IPAM-2, though the figures for other values of N are completely equivalent.

In terms of obfuscation, RAND and NZMIN are significantly poorer than both MAX and 2MAX. This admits a simple explanation; for each possible N-gram, a prediction error is induced only when a padding action becomes more probable than the actual one. After padding, each possible next action is uniformly chosen and its probability is increased by a small amount. Consequently, prediction errors will only be forced over long periods of time and at high padding intensities, as confirmed by the plots. Restricting the randomization to only those N-grams actually executed by the user (as in NZMIN), translates into better obfuscation quality. Nevertheless, padding intensities over $\omega = 0.8$ are required to introduce more than 20 % of the prediction errors.

MAX and 2MAX, on the other hand, perform quite well obfuscation-wise. As expected, 2MAX obfuscates much better due to its ability to flatten some parts of the distribution of N-grams. In both cases, it is possible to achieve high amounts of obfuscation with relatively little padding overhead.

A different picture of how each method acts over the learner's internal state is given in Fig. 7. Here we plot how the distance between the distribution given by the

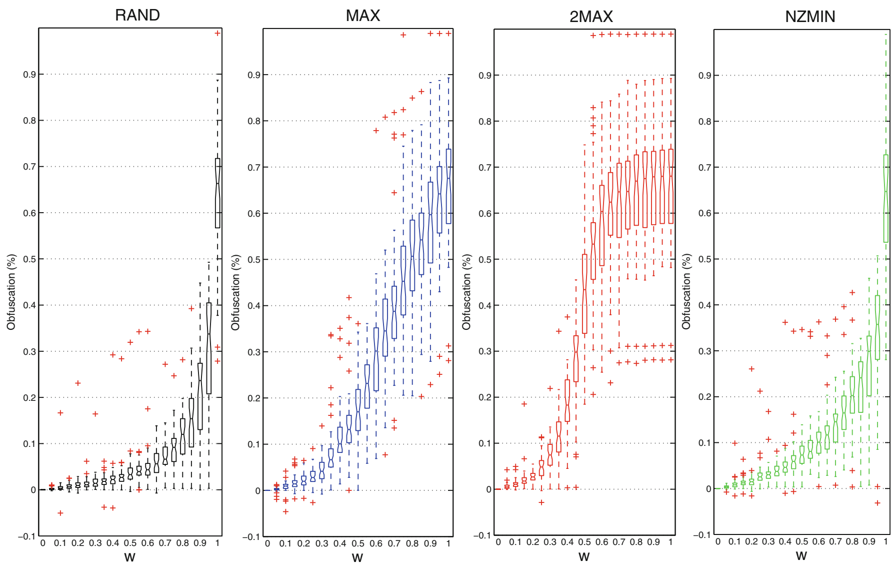

Fig. 6 Amount of obfuscation induced by different randomization heuristics on IPAM-2. For each padding intensity ω, the boxplot shows the distribution computed over the 50 users in the SEA dataset

state and a flat one over all possible N-grams of length $1, 2, \ldots, N$ evolves over time. We use the Kullback-Leibler divergence [28] to measure such deviation, and then average the distance over all possible N-grams:

$$\text{AvgKL} = \frac{1}{N}\sum_{i=1}^{N}\sum_{\xi \in [\![\mathtt{Tr(U)}]\!]_{\mathtt{i}}} P(\xi)\log_2(|\mathtt{Act(U)}|)^{\mathtt{i}}\mathtt{P}(\xi) \tag{8}$$

The behavior shown in Fig. 7 conforms to our intuition. RAND and NZMIN significantly decrease the average KL distance to an ideal distribution; compare both curves with the baseline one that corresponds to not using padding at all. On the other hand, MAX and 2MAX do not attempt to increase the indistinguishability of the state, so KL values are similar to, or even greater than, those intrinsic to the user. In a way, these figures confirm the intuitive idea that indistinguishability heuristics decrease the *information content* of a behavioral pattern, whereas concept drift heuristics simply changes it to something different but with a similar amount of information[2].

5.4 Empirical Evaluation II: Overheads

The introduction of padding sequences by each randomization heuristic can affect the size of the state required both by the padding generator and the learner. Here we focus on the latter. Figure 8 shows the state size, measured in the number of

[2] An alternative method to visualize this would be to *compress* the user traces generated by each heuristic. Whereas those generated by indistinguishability heuristics will be barely compressible, concept drift methods will produce traces with a compression rate similar to the unpadded ones.

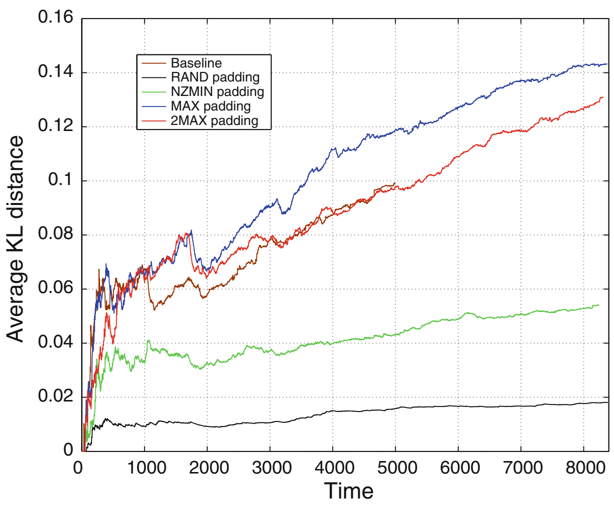

Fig. 7 Time evolution of the average Kullback-Leibler distance between the state and the ideal case

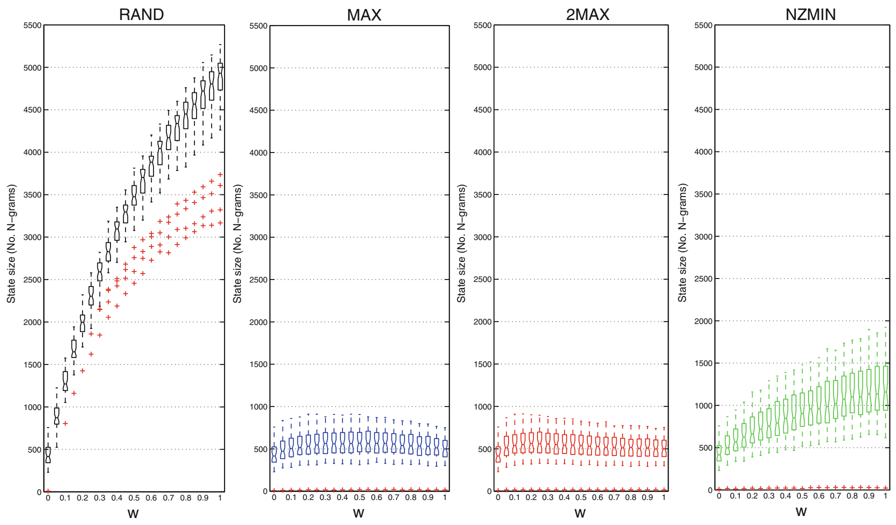

Fig. 8 State size, measured in number of *N*-grams, for IPAM-2 with various randomization heuristic and different padding intensities

different *N*-grams, for different padding methods. Again, we provide boxplots of the distribution computed over the 50 users for different values of ω.

Both MAX and 2MAX do not involve any significant increment of the number of different *N*-grams observed, regardless of the padding intensity. NZMIN, on the contrary, requires both the padding generator and the learner to maintain a bigger internal state. The explanation for this comes from the attempt of the heuristic to maintain a flat distribution over all seen N-grams. Finally, RAND generates as many different N-grams as possible, and this will translate into a continuously growing state. In this last case, we note that this fact is only applicable to the learner, as the padding generator does not need to maintain any internal state for this heuristic.

These overheads are also intimately related to the intrinsic goal of each heuristic. RAND and NZMIN will increase the perceived complexity of user behavior. This will translate in a much higher number of different N-grams. On the contrary, MAX and 2MAX will alter the distribution of N-grams, but the overall complexity remains approximately the same.

6 Concluding Remarks and Future Work

In this paper, we have investigated the threat posed by cloud operators and other 'passive insiders' on the security and privacy of users and organizations that choose to externalize their infrastructure and applications. In particular, we have shown that the regularity and behavioral patterns associated with users' workflows and daily routines can be easily captured by anyone with access to their traces. We argue that this will be generally the case in most anything-as-a-service environment, and we have discussed some of the inherent risks that an organization faces by giving full access to a record of its activities.

Following an approach loosely inspired by that used to counteract traffic analysis attacks, we have proposed to obfuscate user activities as a general measure to prevent profiling attacks. To that end, we have introduced an indistinguishability-based definition of perfectly obfuscated actions and a concrete scheme to randomize user traces in an online fashion. Finally, we have empirically evaluated four different obfuscation heuristics and discussed their performance and tradeoffs in terms of obfuscation quality, influence on the adversary's perceived state, etc. All of them constitute practical schemes that could be deployed with little extra cost on the organization.

The work presented here can be refined and improved in a number of directions. For simplicity, we have assumed that all actions have the same cost. This is unlikely to be the case in a real-world environment. In this regard, it would be interesting to extend our proposal towards minimum cost padding schemes. A priori it is unclear to us what strategy will perform the best under such constraints. Similarly, the notion of a padding intensity is motivated by a pricing model based on cloud usage time. This could be clearly suboptimal under other assumptions.

In this work, we have not discussed the possibility of a more powerful insider who, being aware of the presence of a randomization scheme on the user's side, attempts to separate real from dummy actions. A similar problem was attacked in [16] for mimicry masqueraders. In this setting, some external information could be of help to the attacker (e.g., a priory unlikely actions). We believe this is an interesting direction to evaluate the practical strength of randomization heuristics.

Finally, a richer trace model could enable the construction of behavioral patterns more sophisticated than those analyzed here; for example, the inclusion of time, location, and other contextual atributes. Modeling communication patterns explicitly would be particularly interesting, as it may give a clear picture of who communicates with whom within the organization and the correlations, if any, among their activities. We anticipate that these problems could lead to privacy-protection schemes combining classical traffic analysis prevention measures with the sort of techniques introduced in this work.

Acknowledgments We thank the anonymous reviewers for their insights and comments that have greatly contributed to improve the quality of this work.

References

1. Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R., Molina, J.: Controlling data in the cloud: Outsourcing computation without outsourcing control. In: Proceedings of the 2009 ACM Workshop on Cloud Computing Security—CCSW 2009, pp. 85–90. (2009)
2. Cloud Security Alliance: Security Guidance for Critical Areas of Focus in Cloud Computing v1.0. April 2009. Available at http://www.cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf
3. Newman, R.E., Moskowitz, I.S., Syverson, P., Serjantov, A.: metrics for traffic analysis prevention. In: Privacy Enhancing Technologies Symposium - PET 2003, LNCS 2760, pp. 48–65. Springer, Berlin (2003)
4. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proc. 2000 ACM SIGMOD International Conference on Management of Data - SIGMOD 2000, pp. 439–450. (2000)
5. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud! Exploring information leakage in third-party compute clouds. In: Proceedings of the 16th ACM Conference on Computer and Communications Security—CCS 2009, pp. 199–212. (2009)
6. Zukerman, I., Albrecht, D.W.: Predictive statistical models for user modeling. User Model. User-Adap. Inter. **11**, 5–18 (2001)
7. Davison, B.D., Hirsh, H.: Experiments in UNIX command prediction. Technical Report ML-TR-41, Dept. of Computer Science, Rutgers University (1997)
8. Davison, B.D., Hirsh, H.: Toward an adaptive command line interface. In: Proceedings of the 7th International Conference on Human-Computer Interaction, pp. 505–508. Elsevier, Amsterdam (1997)
9. MIT Reality Mining Project. See http://reality.media.mit.edu
10. Ren, K., Wang, C., Wang, Q.: Security challenges for the public cloud. IEEE Internet Comput. **16**(1), 69–73 (2012)
11. Schonlau, M., DuMouchel, W., Ju, W.-H., Karr, A.F., Theus, M., Vardi, Y.: Computer intrusion: Detecting Masquerades. Stat. Sci. **16**(1), 58–74 (2001)
12. Maxion, R.A., Townsend, T.N.: Masquerade detection using truncated command lines. In: Proceedings of the 2002 International Conference on Dependable Systems and Networks—DSN 2002), pp. 219–228 (2002)
13. Maxion, R.A.: Masquerade detection using enriched command lines. In: Proceedings of the 2003 International Conference on Dependable Systems and Networks—DSN 2003), pp. 5–14. (2003)
14. Killourhy, K.S., Maxion, R.A.: Toward realistic and artifact-free insider-threat data. In: 23rd Annual Computer Security Applications Conference—ACSAC 2007, pp. 87–96. (2007)
15. Wang, K., Stolfo, S.: One-class training for Masquerade detection. In: ICDM Workshop on Data Mining for Computer Security. (2003)
16. Tapiador, J.E., Clark, J.A.: Masquerade mimicry attack detection: A randomised approach. Comput. Secur. **30**(5), 297–310 (2011)
17. Bertacchini, M., Fierens, P.I.: Preliminary results on masquerader detection using compression-based similarity metrics. Electron. J. SADIO **7**(1), (2007)
18. Evans, S., Eiland, E., Markham, S., Impson, J., Laczo, A.: MDLcompress for intrusion detection: Signature inference and masquerade attack. In: 2007 IEEE Military Communications Conference—MILCOM 2007, pp. 1–7. (2007)
19. Posadas, R., Mex-Perera, J.C., Monroy, R., Nolazco-Flores, J.A.: Hybrid method for detecting masqueraders using session folding and hidden markov models. In: Proceedings of the 5th Mexican International Conference on Artificial Intelligence, pp. 622–631. (2006)

20. Oka, M., Oyama, Y., Abe, H., Kato, K.: Anomaly detection using layered networks based on eigen co-occurrence matrix. In: 2004 Symposium on Recent Advances in Intrusion Detection - RAID 2004, LNCS Vol. **3224**, pp. 223–237. Springer, Berlin (2004)
21. Latendresse, M.: Masquerade detection via customized grammars. In: Conference on Detection of Intrusions and Malware & Vulnerability Assessment - DIMVA 2005, LNCS Vol. **3548**, pp. 141–159. Springer, Berlin (2005)
22. Chen, L., Dong, G.: Masquerader Detection using OCLEP: One-class classification using length statistics of emerging patterns. In: WebAge Information Management Workshops—WAIMW 2006, pp. 5–5. (2006)
23. Gebski, M., Wong, R.K.: Intrusion Detection via Analysis and Modelling of User Commands. In: Proc. 7th international Conference on Data Warehousing and Knowledge Discovery - DAWAK 2005, LNCS Vol. **3589**, pp. 388–397. Springer, Berlin (2005)
24. Ourston, D., Mooney, R.J.: Changing the rules: A comprehensive approach to theory refinement. In: 8th National Conference on Artificial Intelligence—AAAI 1990, Vol. **2**, pp. 815–820. (1990)
25. Davison, B.D., Hirsh, H.: Predicting sequences of user actions. In: AAAI-98/ICML'98 Workshop on Predicting the Future: AI Approaches to Time Series Analysis, pp. 5–12. (1998)
26. Bauer, M.: Generation of alternative decompositions for plan libraries. IJCAI'99 Workshop on Learning about Users (1999)
27. Stumpf, S., Bao, X., Dragunov, A., Dietterich, T.G., Herlocker, J., Johnsrude, K., Li, L., Shen, J.Q.: Predicting user tasks: i know what you're doing!. In: 20th National Conference on Artificial Intelligence (AAAI-05), Workshop on Human Comprehensible Machine Learning (2005)
28. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann Math Stat **22**(1), 79–86 (1951)

Author Biographies

Juan E. Tapiador is Associate Professor in the Department of Computer Science at Universidad Carlos III de Madrid. Previously he was Research Associate at the University of York, UK. His main research interests are on applied cryptography and network security. He holds a M.Sc. in Computer Science from the University of Granada and a Ph.D. in Computer Science from the same university. For additional information see: http://www.seg.inf.uc3m.es/~jet.

Julio C. Hernandez-Castro is Senior Lecturer at the School of Computing of the University of Portsmouth. He has a B.Sc. in Mathematics, a M.Sc. in Coding Theory and Network Security, and a Ph.D. in Computer Science. His interests are mainly focused in cryptology, network security, steganography, information leakage and evolutionary computation. For additional information see: http://www.azlaha.com.

Pedro Peris-Lopez has an M.Sc. in Telecommunications Engineering and a Ph.D. in Computer Science. His research interests include protocol and primitive design, lightweight cryptography and cryptanalysis. He is currently doing a postdoc on RFID security at the Information and Communication Theory Group of Delft University of Technology, The Netherlands. For additional information see: http://www.lightweightcryptography.com.