doi.org/10.51891/rease.v12i11.32078

MARKOV CHAINS FROM EVENT FREQUENCIES: FORMALIZATION FOR SYSTEM CALL SEQUENCES AND ANALYSIS ON ADFA-LD

CADEIAS DE MARKOV A PARTIR DA FREQUÊNCIA DE EVENTOS: FORMALIZAÇÃO PARA SEQUÊNCIAS DE CHAMADAS DE SISTEMA E ANÁLISE NO ADFA-LD

CADENAS DE MARKOV A PARTIR DE LA FRECUENCIA DE EVENTOS: FORMALIZACIÓN PARA SECUENCIAS DE LLAMADAS AL SISTEMA Y ANÁLISIS EN ADFA-LD

Regis Nunes Vargas[1]

ABSTRACT: This article proposes defining the states of a Markov chain by the frequency of events, applied to system call sequences, whose large alphabets make chains hard to estimate and to interpret. Each call receives the frequency of its type in a reference set of normal behavior, and the quartiles of that frequency classify it into four states, from the rarest to the most common; each trace becomes a sequence of four symbols, over which a 4 × 4 transition matrix is estimated. The definition was analyzed on the public ADFA-LD dataset, with 833 normal training traces, 4,372 validation traces and 746 attack traces, about 2.7 million events. The matrix of normal behavior is dominated by its diagonal, with a mean stay of 2.3 to 2.6 events per state. It is stable on normal validation data (mean distance between rows of 0.07) and four times farther from the attack matrix (0.28), as attacks have more rare events (41.1% against 27.6%). Four classes proved a compromise between stability and resolution, with an AUC of 0.66 in trace scoring, against 0.77 with the full alphabet. The definition yields interpretable states at a measurable cost in information.

Keywords: Markov chains. System calls. Intrusion detection.

RESUMO: Este artigo propõe definir os estados de uma cadeia de Markov pela frequência dos eventos, aplicada a sequências de chamadas de sistema, cujos alfabetos grandes tornam as cadeias difíceis de estimar e de interpretar. Cada chamada recebe a frequência do seu tipo em um conjunto de referência de comportamento normal, e os quartis dessa frequência a classificam em quatro estados, do mais raro ao mais comum; cada traço vira uma sequência de quatro símbolos, sobre a qual se estima uma matriz de transição 4 × 4. A definição foi analisada na base pública ADFA-LD, com 833 traços normais de treino, 4.372 de validação e 746 de ataque, cerca de 2,7 milhões de eventos. A matriz do comportamento normal é dominada pela diagonal, com permanência média de 2,3 a 2,6 eventos por estado. Ela é estável nos dados normais de validação (distância média entre linhas de 0,07) e quatro vezes mais distante da matriz dos ataques (0,28), que têm mais eventos raros (41,1% contra 27,6%). Quatro classes mostraram-se um compromisso entre estabilidade e resolução, com AUC de 0,66 na pontuação de traços, contra 0,77 com o alfabeto completo. A definição produz estados interpretáveis, a um custo mensurável em informação.

Palavras-chave: Cadeias de Markov. Chamadas de sistema. Detecção de intrusão.

[1] Doutor em Engenharia Elétrica pela Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia e professor na Secretaria Municipal de Educação de Uberlândia.

RESUMEN: Este artículo propone definir los estados de una cadena de Markov por la frecuencia de los eventos, aplicada a secuencias de llamadas al sistema, cuyos alfabetos grandes dificultan estimar e interpretar las cadenas. Cada llamada recibe la frecuencia de su tipo en un conjunto de referencia de comportamiento normal, y los cuartiles de esa frecuencia la clasifican en cuatro estados, del más raro al más común; cada traza se vuelve una secuencia de cuatro símbolos, sobre la cual se estima una matriz de transición 4 × 4. La definición se analizó en la base pública ADFA-LD: 833 trazas normales de entrenamiento, 4.372 de validación y 746 de ataque, cerca de 2,7 millones de eventos. La matriz normal está dominada por su diagonal, con permanencia media de 2,3 a 2,6 eventos por estado. Es estable en la validación normal (distancia media entre filas de 0,07) y cuatro veces más distante de la matriz de los ataques (0,28), que tienen más eventos raros (41,1% frente a 27,6%). Cuatro clases resultaron un compromiso entre estabilidad y resolución, con AUC de 0,66 en la puntuación de trazas, frente a 0,77 con el alfabeto completo. La definición produce estados interpretables, a un costo medible.

Palabras clave: Cadenas de Markov. Llamadas al sistema. Detección de intrusiones.

1. INTRODUCTION

Anomaly-based host intrusion detection systems (HIDS) learn the normal behavior of a program and flag what departs from it. One of the most widely used representations of that behavior is the sequence of system calls that the process makes to the kernel, and Markov chains are among the simplest and most interpretable models for these sequences (WARRENDER C, et al., 1999; JU WH, et al., 2001; TIAN X, et al., 2008; MAGGI F, et al., 2010; SHA W, et al., 2018).

The alphabet of these sequences, however, is large. In ADFA-LD there are 150 call types in the training data, and another 25 appear only later. A chain of order k over V call types has V^k contexts: for second order, more than 22 thousand, estimated from about 308 thousand training events. The estimates become sparse and the states hard to interpret. The choice of order is also usually empirical: An ZQ (2005) reported better performance with first order than with second order, whereas Sha W, et al. (2018) combine several orders.

We propose defining the states of the chain by the frequency of the events. Each call receives the frequency of its type in a reference set of normal behavior, and the quartiles of that frequency classify it into four states, from the rarest to the most common. The transition matrix then has 4×4 entries, the states have a direct meaning (rare, common), and the same quartiles apply to any trace. The cost is losing the identity of each call.

This definition is based on two earlier works. Vargas RN and Linhares RR (2026) study long-range dependence in lexical renewal series: they rewrite a text as a sequence of four symbols, based on the quartiles of word frequency, in analogy with the bases of a DNA

sequence. Linhares RR (2023) deals with the fractional Poisson process and long-range dependence in DNA sequences. We adopt from them the way of rewriting a sequence into four frequency classes, but we apply it to system call sequences and use it to estimate a Markov chain, instead of studying long-range dependence.

This paper formalizes that definition and analyzes it in a simple way on ADFA-LD (CREECH G, et al., 2014), a public set of Linux system call traces, with normal traffic and six types of attack, already analyzed, for example, by Xie M, et al. (2013). There are three contributions: (i) the definition of the states $Y(t)$ from event frequency and the corresponding transition matrix; (ii) a descriptive analysis of the definition on about 2.7 million real events (class composition, normal-behavior matrix, stability and difference from attacks); (iii) the effect of the number of classes.

The paper is organized as follows: Section 2 summarizes related work; Section 3 presents the formalization; Section 4 brings the analysis on ADFA-LD; Section 5 discusses results and limitations; and Section 6 concludes.

2. RELATED WORK

Markov chains on system calls. Warrender C, et al. (1999) compared, on system call data, sequence enumeration, relative frequencies, rule induction and hidden Markov models, and concluded that, for the problem studied, methods simpler than hidden Markov models are probably sufficient. Ju WH, et al. (2001) used high-order Markov chains, combined with an independence model, to profile the command sequences of a user. Tian X, et al. (2008) characterized the normal behavior of privileged programs with a homogeneous Markov chain whose states correspond to the unique call types of the training data. Maggi F, et al. (2010) added a Markov behavioral model to a detector based on call arguments, in order to capture temporal correlations, and Sha W, et al. (2018) proposed a framework of multi-order Markov chains to detect anomalies in cloud servers. Our proposal differs in grouping the call types into a few states by frequency, instead of using one state per type.

ADFA-LD. The ADFA Linux Dataset was created by Creech G, et al. (2014) with a modern operating system and contemporary attack methods, and is publicly available. It gathers system call traces of normal use and of six types of attack. Xie M, et al. (2013) carried out a preliminary analysis of the dataset, examining the length, common patterns and

frequency of the traces, and evaluated a simple nearest-neighbor detector, with acceptable performance for some types of attack.

3. FORMALIZATION

3.1 Concepts: system call, event and trace

System call. A program does not directly touch files, network or memory: it asks the operating system kernel to do so on its behalf, and each such request is a system call (opening a file, reading, writing, creating a process, opening a connection). Each type of call has a number that identifies it. ADFA-LD provides only these numbers, without the names.

Event. It is the generic word we use for each item of a sequence. In this study, an event is a system call, that is, a number such as 6, 63 or 195.

Trace. It is the ordered record of all the calls that a process made during one execution. In ADFA-LD, each file is a trace, a line of numbers. The file UTD-0001.txt, of normal use, is in the Training_Data_Master folder of ADFA-LD (CREECH G, et al., 2014; Section 4 indicates where to obtain the dataset), has 819 events and begins like this:

6 6 63 6 42 120 6 195 120 6 6 114 114 1 1 252 252 252 ...

3.2 Definition of the states

Let $e_1, e_2, \ldots, e_N$ be the sequence of events of a trace, in the order in which they occur (here, the number of each system call). Given a reference set of normal behavior (the training data), we count how many times each event type e appears in it:

$$\text{freq}(e) = |\{\, t : e_t = e \,\}| \tag{1}$$

Each position t of any trace receives the value $X(t) = \text{freq}(e_t)$, the frequency of the type of that event in the training data; types never seen in training have frequency zero. We compute the quartiles Q_1, Q_2, Q_3 (25th, 50th and 75th percentiles) of the distribution of $X(t)$ over all training positions, so that every occurrence counts, and define the state series:

$$Y(t) = \begin{cases} C_1, & X(t) \le Q_1 \\ C_2, & Q_1 < X(t) \le Q_2 \\ C_3, & Q_2 < X(t) \le Q_3 \\ C_4, & X(t) > Q_3 \end{cases} \tag{2}$$

Three properties of the definition are worth noting. First, the values are sorted only to find the quartiles; the series Y is never reordered and follows the original order of the sequence,

which is the information the chain needs to capture. Second, the quartiles and frequencies come only from the normal training data, and the same values are applied to any other trace (validation or attack), which avoids information leakage; unseen types fall into C_1. Third, the state has a direct meaning: C_1 gathers the rarest call types and C_4 the most common ones. Since many occurrences belong to the same type, the states do not have exactly the same number of events (see Section 4.2).

3.3 Transition matrix

We treat $Y(t)$ as a first-order Markov chain over the states $\{C_1, C_2, C_3, C_4\}$ and estimate the transition matrix P by counting how many times each state is followed by each other state, within each trace (there is no transition between different traces) and summing over the training traces:

$$\hat{p}_{ij} = \frac{n_{ij}}{n_i} \tag{3}$$

where n_{ij} is the number of times C_i is followed by C_j and n_i is the total number of times the chain left C_i. The resulting matrix is an average over the training processes.

3.4 What the chain allows us to compute

With P one obtains simple and interpretable quantities. The stationary distribution π, to which the chain converges, solves $\pi P = \pi$ with $\sum_j \pi_j = 1$ and is computed as the left eigenvector of P. The probability of being in C_j after n steps, starting from C_i, comes from the Chapman-Kolmogorov equation, $P_{ij}^{(n)} = (P^n)_{ij}$, where P^n is the matrix P raised to the power n; the speed at which the rows of P^n become equal shows after how many steps the chain "forgets" the initial state. The mean stay time in C_i is $1/(1 - p_{ii})$ events. And the score of a trace of L events, given by the mean negative log-likelihood per transition under the matrix estimated on the training data, measures how much the trace departs from normal behavior:

$$s = -\frac{1}{L-1}\sum_{t=1}^{L-1} \ln \hat{p}_{Y(t)\,Y(t+1)} \tag{4}$$

The larger s is, the more anomalous the trace.

3.5 Measures used in the analysis

To compare two matrices P and Q we use the mean total variation distance between the rows:

$$d(P,Q) = \frac{1}{S}\sum_{i=1}^{S}\frac{1}{2}\sum_{j=1}^{S}|P_{ij} - Q_{ij}| \quad (5)$$

which equals 0 for identical matrices and 1 for totally different rows (S is the number of states). To evaluate the score s as a separator of attacks and normal traffic we use the AUC (Area Under the ROC Curve), with attack traces as positives and normal validation traces as negatives, and 95% confidence intervals by *bootstrap* over the traces (500 resamples, fixed seed). These values describe the definition of $Y(t)$; they do not propose a detector.

4. ANALYSIS ON ADFA-LD

4.1 Data

We use ADFA-LD (CREECH G, et al., 2014), whose official page is indicated in the references (UNSW CANBERRA, [n.d.]). We obtained the data from a labelled copy published on GitHub (VERAZUO, [n.d.]), which reproduces the official structure; inside that copy, the file ADFA-LD.zip contains the three data folders (Training_Data_Master, Validation_Data_Master and Attack_Data_Master). The example trace of Section 3, UTD-0001.txt, is located at ADFA-LD/Training_Data_Master/UTD-0001.txt. The script baixar_adfa.py of the package downloads and unpacks everything into dados/ADFA-LD/, and the dataset is not redistributed with the paper. Each file is a trace, in the sense defined in Section 3. The normal traces come from ordinary use of the system, and the attack traces were recorded while an attack ran in a laboratory: attempts to guess passwords on FTP and SSH (*Hydra-FTP* and *Hydra-SSH*), the creation of a user (*Adduser*), two remote-control payloads (*Meterpreter* and *Java-Meterpreter*) and a *Web Shell*. Table 1 summarizes the three groups: there are 833 normal training traces, 4,372 validation traces and 746 attack traces, with a median of about 200 to 340 events per trace. The attack traces split into *Adduser* (91 traces), *Hydra-FTP* (162), *Hydra-SSH* (176), *Java-Meterpreter* (124), *Meterpreter* (75) and *Web Shell* (118). The training data contain 150 distinct call types; another 25 types appear only in the validation and attack data.

Table 1 - ADFA-LD data used in this work (lengths in number of events per trace).

Group	Traces	Events	Median	Mean	Range
Normal training	833	308,077	201	370	79 to 2,948
Normal validation	4,372	2,122,085	343	485	77 to 4,494
Attack	746	317,388	290	425	75 to 2,712

Source: Data extracted from ADFA-LD (CREECH G, et al., 2014).

4.2 The classes

Applying the definition to the normal training data, the cutoff frequencies are $Q_1 = 8{,}223$, $Q_2 = 13{,}709$ and $Q_3 = 31{,}342$ occurrences. The states gather 138 call types in C_1, 7 in C_2, 4 in C_3 and only 1 in C_4 (call number 3, which alone accounts for 18.1% of the events). The shares of events are 26.7%, 24.7%, 30.5% and 18.1%: they are not equal to 25% because the occurrences of a given type always go to the same state, and a few types concentrate many occurrences.

From event to state. The protocol counts how many times each call type appears in the training data and classifies it by frequency. Call 6 appears 22,782 times, so it goes to state C_3. Call 63 appears only 942 times, so it goes to C_1 (rare). Thus, the beginning of trace UTD-0001.txt becomes the sequence of states below:

event:	6	6	63	6	42	120	6	195	120	6	6	114	114	1	1	252	252	252	1	1
state:	C_3	C_3	C_1	C_3	C_1	C_1	C_3	C_3	C_1	C_3	C_3	C_1	C_1	C_1	C_1	C_1	C_1	C_1	C_1	C_1

The Markov chain works on this sequence of states: it counts how often C_3 is followed by C_3, by C_1, and so on.

4.3 Matrix of normal behavior

The matrix estimated on the normal training data, over 307,244 transitions, is

$$P = \begin{pmatrix} 0.562 & 0.154 & 0.210 & 0.075 \\ 0.174 & 0.619 & 0.116 & 0.091 \\ 0.179 & 0.108 & 0.595 & 0.118 \\ 0.107 & 0.110 & 0.216 & 0.567 \end{pmatrix} \quad (6)$$

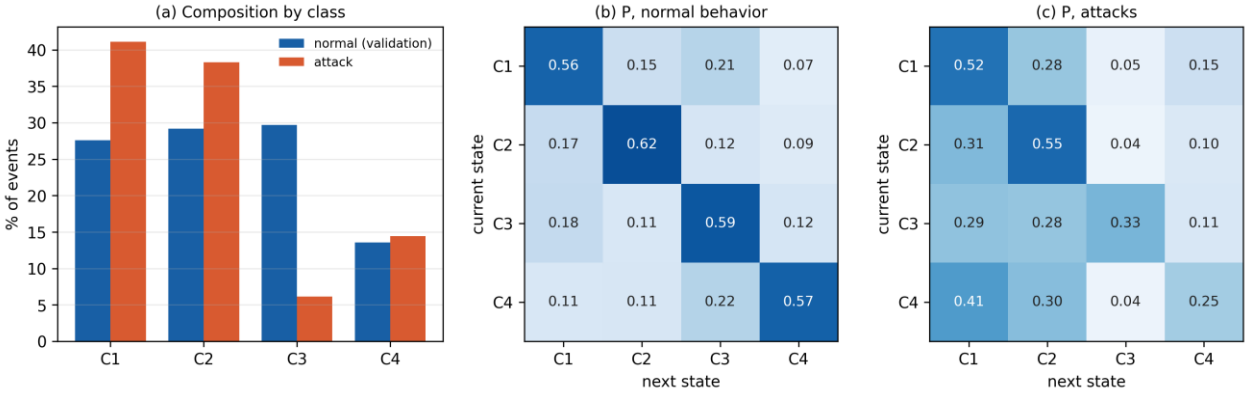
The diagonal dominates (from 0.56 to 0.62): calls tend to repeat in the same frequency class, which reflects loops and repetitive stretches of the programs. The mean stay time is 2.28

events in C_1, 2.63 in C_2, 2.47 in C_3 and 2.31 in C_4. The stationary distribution is $\pi = (0.267;\ 0.247;\ 0.305;\ 0.181)$, practically equal to the observed composition, as expected for a matrix estimated from the sequence itself. The chain converges quickly: the second-largest eigenvalue of P is 0.52, and the rows of P^n differ by less than 0.01 with $n = 8$ steps and by less than 0.001 with $n = 11$.

4.4 Stability and difference from attacks

Figure 1 compares normal behavior and attack behavior.

Figure 1 - (a) Composition of the events by class, in the normal validation traces and in the attack traces. (b) Transition matrix estimated on the normal training data. (c) Transition matrix estimated on the attack traces, with the same quartiles as the normal training data.



Source: Data extracted from ADFA-LD (CREECH G, et al., 2014).

Stability. The training matrix is similar to the one estimated on the normal validation traces: the mean distance between the rows is 0.070. The composition by class, however, is not identical (for example, 13.6% of the events in C_4 in validation, against 18.1% in training), which shows that the definition depends on the reference set.

Difference from attacks. The matrix estimated on the attack traces, with the same quartiles as the training data, is much farther from the normal matrix: the distance is 0.279, four times the distance between training and validation. The difference has two simple marks. The composition changes: 41.1% of the attack events fall in C_1 (rare calls), against 27.6% in the normal validation traces, and only 6.1% fall in C_3, against 29.7%. All six attack types have more events in C_1 than normal traffic (from 35.5% in *Web Shell* to 48.4% in *Adduser*). And persistence drops in the most common states: the diagonal of the attack matrix is 0.52; 0.55; 0.33 and 0.25, against 0.56; 0.62; 0.59 and 0.57 in normal behavior, which corresponds to mean stays of 1.49 events in C_3 and 1.33 in C_4, against 2.47 and 2.31.

4.5 Effect of the number of classes

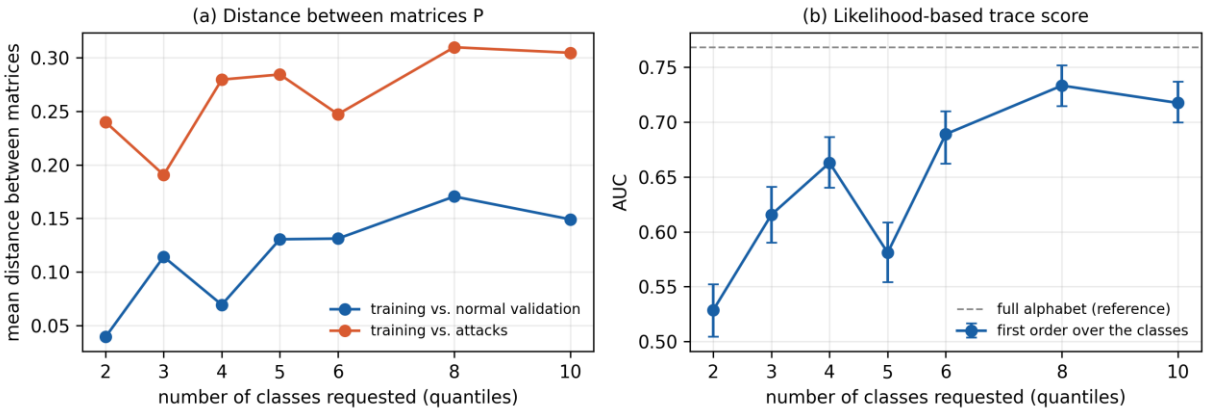
Quartiles give four classes, but the definition holds for any number of quantiles. Table 2 and Figure 2 show what changes with 2 to 10 quantiles. When the cut points fall inside the mass of a very frequent call (call 3 has 18.1% of the events), some quantiles coincide and the effective number of classes is smaller than the number requested: 6 quantiles give 5 classes, 8 give 7 and 10 give 9.

Table 2 - Effect of the number of classes. d is the distance between matrices (Section 3); the ratio is d(training, attack) divided by d(training, validation); the AUC is that of the first-order likelihood-based trace score, with a 95% CI by *bootstrap* (300 resamples, except in the 4-class row, which uses the 500 of the main analysis).

Quantiles requested	Effective classes	Largest class (% of events)	d (training, validation)	d (training, attack)	Ratio	AUC [CI]
2	2	51.4	0.040	0.240	6.1	0.529 [0.504; 0.552]
3	3	36.0	0.114	0.191	1.7	0.616 [0.590; 0.641]
4	4	30.5	0.070	0.279	4.0	0.663 [0.640; 0.687]
5	5	21.7	0.131	0.284	2.2	0.581 [0.554; 0.609]
6	5	28.2	0.131	0.247	1.9	0.689 [0.662; 0.710]
8	7	18.1	0.170	0.310	1.8	0.733 [0.714; 0.752]
10	9	18.1	0.149	0.304	2.0	0.717 [0.699; 0.737]

Source: Data extracted from ADFA-LD (CREECH G, et al., 2014).

Figure 2 - Effect of the number of classes requested. (a) Distance between the training matrix and those of the normal validation data and of the attacks. (b) AUC of the first-order likelihood-based score; the dashed line is the reference with the full alphabet (Section 4.6).



Source: Data extracted from ADFA-LD (CREECH G, et al., 2014).

Three observations. First, stability (distance between training and validation) is higher with few classes (0.040 with 2, 0.070 with 4) and drops with many (0.13 to 0.17 with 5 to 9 effective classes), because each cell of the matrix receives fewer transitions. Second, the separation between normal and attack, measured by the ratio between the two distances, is

larger with 2 classes (6.1) and with 4 (4.0), and stays close to 2 in the others. Third, the **AUC** rises from 0.53 (2 classes) to 0.73 (7 effective classes), but with no regular order: 5 classes (0.58) is worse than 4 (0.66), possibly because of cut points that fall on the most frequent calls, which we did not verify. Four classes is therefore a compromise: stable matrix, clearly distant attack, small number of parameters. It is not the choice that maximizes the AUC (7 effective classes give 0.73), and there is no ideal number of classes in general.

4.6 Simple trace scoring

To see what the definition preserves, we score each validation (normal) and attack trace by the equation for s, with the four-class matrix estimated on the training data. As a reference, we repeat the scoring with a first-order chain over the full alphabet of call types (smoothing of 0.1 on each count, and unseen types gathered into one extra state), which does not lose the identity of the calls.

Table 3 - AUC of the likelihood-based score, with attacks as positives and normal validation data as negatives (4,372 normal traces and 746 attack traces). CI: 95% confidence interval by *bootstrap*. The remaining columns are the AUC per attack type (H: *Hydra*; J-Met.: *Java-Meterpreter*; Meterp.: *Meterpreter*; Web Sh.: *Web Shell*).

Model	Overall	CI	*Adduser*	*H-FTP*	*H-SSH*	*J-Met.*	*Meterp.*	*Web Sh.*
4 classes, 1st order	0.663	0.640 to 0.687	0.56	0.77	0.75	0.51	0.74	0.58
Full alphabet, 1st order	0.768	0.745 to 0.790	0.73	0.81	0.81	0.71	0.69	0.78

Source: Data extracted from ADFA-LD (CREECH G, et al., 2014).

The four-class chain separates the *Hydra* attacks reasonably well (0.77 and 0.75) and *Meterpreter* (0.74), but barely separates *Java-Meterpreter* (0.51) and *Adduser* (0.56). The full alphabet, with the same first order, is better overall by 0.105 (95% CI: 0.079 to 0.133). This is the cost of grouping the call types into four classes, and it is the price of interpretability and of a small number of parameters.

5. DISCUSSION

5.1 What the definition offers

Defining the states by the frequency of the events reduces the alphabet of 150 call types to four symbols with a direct meaning, and produces a 4×4 matrix estimated from hundreds of thousands of transitions. On ADFA-LD, this matrix is stable across normal-traffic sets (distance of 0.07) and distinct from that of the attacks (0.28), with two marks that are easy to

explain: attacks have more rare events and persist less in the common states. The analysis of the number of classes shows that there is a balance between stability and resolution, and that four classes is a reasonable point of that balance, without being the best for detection. The cost of the definition, measured by comparison with the full alphabet, is a loss of 0.105 in AUC.

5.2 Limitations

First, we use a single dataset, generated in the laboratory with a Linux system of a specific era and with attacks run in a controlled manner; the results may not extend to other systems, workloads and attacks. Second, the matrix P is an average over different processes, and the dataset does not report which program generated each trace, so we did not run per-program analyses. Finally, the definition depends on the reference set: the training frequencies do not coincide with those of the validation data (for example, 18.1% and 13.6% of the events in C_4), and the quantile cut points fall on a few very frequent calls, which makes the classes uneven and makes the effective number of classes vary without regularity.

6. CONCLUSION

We formalized the definition of the states of a Markov chain from the frequency of the events, applied to system call sequences: each call is classified by the quartiles of the frequency of its type in a normal reference set, and each trace becomes a sequence of four symbols with a 4×4 transition matrix. On ADFA-LD, the definition produces a matrix that is stable on normal traffic and clearly different from that of the attacks, which have more rare events and persist less in the common states; varying the number of classes shows a balance between stability and resolution, in which four classes is a compromise and not an optimum; and grouping the calls costs 0.105 in AUC relative to the full alphabet.

REFERENCES

1. AN ZQ. Markov chain model of syscall based intrusion detection. Application Research of Computers, 2005.

2. CREECH G, et al. A semantic approach to host-based intrusion detection systems using contiguous and discontiguous system call patterns. IEEE Transactions on Computers, 2014.

3. JU WH, et al. A hybrid high-order Markov chain model for computer intrusion detection. Journal of Computational and Graphical Statistics, 2001.

4. LINHARES RR. Fractional Poisson process: long-range dependence in DNA sequences. Model Assisted Statistics and Applications, 2023; 18: 33-43.

5. MAGGI F, et al. Detecting intrusions through system call sequence and argument analysis. IEEE Transactions on Dependable and Secure Computing, 2010.

6. SHA W, et al. Statistical learning for anomaly detection in cloud server systems: a multi-order Markov chain framework. IEEE Transactions on Cloud Computing, 2018.

7. TIAN X, et al. Intrusion detection based on system calls and homogeneous Markov chains. Journal of Systems Engineering and Electronics, 2008.

8. UNSW CANBERRA. ADFA IDS Datasets (includes the ADFA Linux Dataset, ADFA-LD). UNSW Canberra, [n.d.]. https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-IDS-Datasets/

9. VARGAS RN, LINHARES RR. Longa dependência em séries de renovação lexical [Long-range dependence in lexical renewal series]. RECIMA21 - Revista Científica Multidisciplinar, 2026; 7(10): e7109165. DOI: 10.47820/recima21.v7i10.9165.

10. VERAZUO. A labelled version of the ADFA-LD dataset for HIDS evaluation. GitHub, [n.d.]. https://github.com/verazuo/a-labelled-version-of-the-ADFA-LD-dataset

11. WARRENDER C, et al. Detecting intrusions using system calls: alternative data models. In: IEEE Symposium on Security and Privacy. Proceedings [...]. IEEE, 1999.

12. XIE M, et al. Evaluating host-based anomaly detection systems: a preliminary analysis of ADFA-LD. In: 6th International Congress on Image and Signal Processing. Proceedings [...]. IEEE, 2013.