

Journal of the American Statistical Association

ISSN: 0162-1459 (Print) 1537-274X (Online) Journal homepage: http://www.tandfonline.com/loi/uasa20

Malware Family Discovery Using Reversible Jump MCMC Sampling of Regimes

Alexander D. Bolton & Nicholas A. Heard

To cite this article: Alexander D. Bolton & Nicholas A. Heard (2018): Malware Family Discovery Using Reversible Jump MCMC Sampling of Regimes, Journal of the American Statistical Association, DOI: 10.1080/01621459.2018.1423984

To link to this article: https://doi.org/10.1080/01621459.2018.1423984

Accepted author version posted online: 19 Jan 2018.

Submit your article to this journal

Article views: 13

View related articles

View Crossmark data

Full Terms & Conditions of access and use can be found at
http://www.tandfonline.com/action/journalInformation?journalCode=uasa20

ACCEPTED MANUSCRIPT

Check for updates

Malware Family Discovery Using Reversible Jump MCMC Sampling of Regimes

Alexander D. Bolton*
Department of Mathematics, Imperial College London

Nicholas A. Heard
Department of Mathematics, Imperial College London
& Heilbronn Institute for Mathematical Research, University of Bristol

Abstract

Malware is computer software which has either been designed or modified with malicious intent. Hundreds of thousands of new malware threats appear on the internet each day. This is made possible through reuse of known exploits in computer systems which have not been fully eradicated; existing pieces of malware can be trivially modified and combined to create new malware which is unknown to anti-virus programs. Finding new software with similarities to known malware is therefore an important goal in cyber-security.

A dynamic instruction trace of a piece of software is the sequence of machine language instructions it generates when executed. Statistical analysis of a dynamic instruction trace can help reverse engineers infer the purpose and origin of the software that generated it. Instruction traces have been successfully modeled as simple Markov chains, but empirically there are change points in the structure of the traces, with recurring regimes of transition patterns.

Here, reversible jump MCMC for change point detection is extended to incorporate regime-switching, allowing regimes to be inferred from malware instruction traces. A similarity measure for malware programs based on regime matching is then used to infer the originating families, leading to compelling performance results.

Keywords: RJMCMC, change point analysis, regime-switching, dynamic instruction trace

*The authors gratefully acknowledge funding from EPSRC and Heilbronn Institute

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

1 Introduction

Malicious software, or *malware*, is a growing problem in cyber-security, with McAfee (2017) reporting the detection of approximately 240 new samples every minute. However, most new malware is created by modifying an existing program (Storlie et al., 2014), meaning that any new malware is likely to be similar to some existing known malware. Malware programs all derived from the same original program are said to form a *malware family*.

One approach to determining whether a new software executable is malicious is to compare the program against a repository of known malware to search for similar entities. However, to obscure the function of a piece of malware, developers often compress, or *pack*, the program into several layers so that no information about source code can be directly obtained (Royal et al., 2006). In this case, inference about the software can only be made using the instructions that are called when the program actually runs. The sequence of instructions generated by a program at runtime is called its *dynamic instruction trace*.

Reverse engineers analyze dynamic instruction traces to try to manually unravel the origin and purpose of malware samples they receive. However, this process can take many hours, if not days or weeks (Anderson et al., 2014). This article presents a statistical approach for automatically detecting the malware family that gave rise to a new piece of malware, which shows good promise for reducing the burden on the reverse engineer.

More specifically, the proposed methodology uses a regime-based change point model to define a similarity measure between pairs of software samples using their dynamic instruction traces: Each trace is modeled as a piecewise homogeneous Markov chain with change points indicating changes to the underlying transition probability matrix. At each change point, the transition probability matrix entries can change to one of a finite (but unknown) number of states, which are termed *regimes*. Malware similarity is then quan-

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

tified by the proportion of instructions for which two samples are governed by transition probability regimes which are common to both malware traces. Single-link clustering can then be applied to the matrix of estimated similarities for a collection of malware samples to identify possible families. The approach is demonstrated on a database of malware instruction traces provided by reverse engineers at Los Alamos National Laboratory. A hybrid classifier combining the proposed method with the previous state-of-the art method is shown to be highly competitive.

1.1 Background to Inferring Change Points

Online change detection methods have previously been proposed for detecting changes in the state transition probabilities of Markov chains. Yakir (1994) derives the Bayes rule for detecting a change in the transition probability matrix, when the values of the matrix before and after the change point are assumed to be known. Han and Tsung (2009) shows the theoretical optimality of the CUSUM chart (Page, 1954) for detecting the change from one known transition probability matrix to another. Xian et al. (2016) uses the CUSUM chart to detect a change in distribution of a Markov chain when the post-change transition probabilities are unknown. Online change point methods are also used in Tartakovsky et al. (2006) to detect changes in the behavior of a computer network, applying a multivariate version of the CUSUM chart.

Fitzpatrick and Marchev (2013) considers multiple Markov chains whose transition probabilities change according to a hidden Markov model. Assuming the number of hidden states is known (which is not the case in this article), the posterior distribution of transition probabilities and change points is estimated using an extension of the Gibbs sampler of Chib (1996). Xing et al. (2012) models consumer credit ratings using a finite-state continuous-time Markov process, assuming the transition matrices alter at change points.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

There is a broad range of methods of varying complexity in the statistical literature for making inference about change points. If a single estimate of the best positioning of change points is needed, then the Pruned Exact Linear Time (PELT) method of Killick et al. (2012) can be used to efficiently find the set of change points that minimizes a suitable penalty function such as the AIC or BIC. Polansky (2007) applies information criteria to identify the change points that are present in Markov chains. In relation to malware, Bolton and Heard (2014) uses the PELT method to identify change points and regimes in malware dynamic traces, assuming that some regimes are known beforehand. Zhao and Marriott (2014) uses variational Bayes to estimate the number of regimes that affect a regime-switching log-normal model and describe a method to efficiently identify a single estimate of the number of regimes, the regime-switching probabilities and the parameters characterizing each regime; however, this approach does not yield estimates of the marked change point locations, which are integral to the proposed malware similarity measure (4).

Hidden Markov models (HMMs) can be used to perform inference when there is an unobserved Markovian sequence of states that governs the behavior of an observed process. For example, Goldfeld and Quandt (1973) describes a linear regression model where the regression coefficients can switch between two unknown vector values which could be regarded as regimes. An unobserved two-state Markov chain determines the parameters at each time t; maximum likelihood estimation is used to estimate the two vectors of regression parameters and the regime transition probabilities. When the number of regimes is unknown, inference can be made using the infinite HMM of Beal et al. (2002), which assumes a countably infinitely many hidden states; an extended Dirichlet process is used to model the transition structure. Beal et al. (2002) makes inference about the hidden states using a Gibbs sampler that repeatedly iterates through every observation, sampling the sequence of hidden states. Van Gael et al. (2008) uses beam sampling to improve mixing, but

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

this still requires repeatedly iterating through all observations. Making inference about the change points that exist in the trace is preferable to using HMMs since there are some very long segments between change points, containing millions of instructions. It is convenient to make inference about the number of changes and their positions, rather than working with an HMM which is more suited to applications where the regimes change frequently.

Vasas et al. (2007) simulates change points and regimes, using a mixture of Gibbs sampling and RJMCMC to estimate regime changes in a time series of hydrological data. It is assumed that there are two regimes in total and that the parameterizations of the regimes differ. Unlike previous work, in this article RJMCMC is used to make inference about both the change points and regimes that affect a sequence of observations. Before the RJMCMC sampler is run, none of the following are assumed to be known: the number or position of change points, the number of regimes that affect each process, and the probability distributions of the parameters that govern the regimes.

The remainder of the article is organized as follows: Section 2 gives some background to dynamic instruction traces and Section 3 proposes a piecewise homogeneous Markov chain change point model with recurring regimes. Section 4 explains how the change point regimes can be used to define similarities between malware instruction traces. Section 5 presents a new reversible jump Markov Chain Monte Carlo algorithm for sampling the posterior distribution of change points and regimes. Section 6 investigates the reliability of the proposed method, initially through a simulation study and then with an analysis of a repository of real malware dynamic instruction traces. Section 7 summarizes the main contributions of the work.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

2 Dynamic Instruction Traces as Markov Chains

A typical computer architecture has several thousands of different possible instructions that can be called as a program runs, and these can be grouped into functional categories (Storlie et al., 2014). The sequence of categories called by a program form a *dynamic instruction trace*. To give a specific example of a small part of a dynamic trace, the following sequence of instructions was generated by a malicious piece of software:

```
push, push, push, mov, call, xor, mov, mov, mov, xor, mov
```

The `push` instruction writes a value to memory, `mov` moves memory contents, `call` implements a subroutine in the code, and `xor` is a bitwise logical operation.

To avoid risking infection when obtaining a dynamic instruction trace, a malware reverse engineer seeking to make inference about a new piece of software will run the program safely within a virtual environment on their computer, recording the sequence of instructions as the program executes. This methodology for generating instruction traces has two potential setbacks: firstly, it is unclear for how long the reverse engineer should track the program and record the instructions. Secondly, if the malware carries the capability to detect that it is being run in a virtual environment, then it may call instructions to confound a reverse engineer. Nevertheless, generating dynamic instruction traces provides a reliable means of obtaining data of some kind from each program.

Following Storlie et al. (2014) and others, dynamic instruction traces will be modeled here as Markov chains. With thousands of possible instructions, it would be too cumbersome to model transitions between individual instruction types. Different categorizations of varying granularity have been proposed, and in particular Storlie et al. (2014) describes categorizations of instructions into 8, 56 or 86 groups. The categorization into 8 groups includes categories such as math, logic, memory and stack, and this will be used for the

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

remainder of the article. Denoting the chosen number of instruction categories by c, a dynamic instruction trace is then a sequence $\boldsymbol{x} = (x_1, \ldots, x_n)$ with each $x_t \in \{1, \ldots, c\}$.

Figure 1 shows 11.4 million transitions $x_{t-1} \to x_t$ observed in the dynamic instruction trace from an example piece of malware. Some clear Markovian structure is revealed by the plot: instructions in category 1, for example, are always followed by a category 5 instruction. However, the plot also reveals something else; the Markov chain is not homogeneous. There are at least two obvious change points in the transition patterns, one after 3.3 million instructions were observed, another after 4.8 million instructions. Empirical estimates of the transition probability matrices for these three segments are displayed in Figure 2, showing some clear differences; for example, in segments 2 and 3, a category 2 or 3 instruction is always followed by a category 5 instruction

To compare two dynamic instruction traces, Storlie et al. (2014) and Dai et al. (2009) make use of global summaries of whole traces, assuming that the Markov structure does not change over time. However, from Figure 1 it is clear that changes can occur in the patterns of instructions in a trace. Therefore this article proposes modeling each instruction trace $\boldsymbol{x}$ as a Markov chain with a time-varying, $c \times c$ transition probability matrix $\boldsymbol{P}(t)$, where

$$\boldsymbol{P}_{i,i'}(t) = \Pr(x_t = i' \mid x_{t-1} = i).$$

Three Bayesian change point models for $\boldsymbol{P}(t)$, of increasing complexity, will be presented in the next section. The two simpler models will be sequentially deployed during the burn-in phase of a reversible jump MCMC sampler, described in Section 5, to aid convergence.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

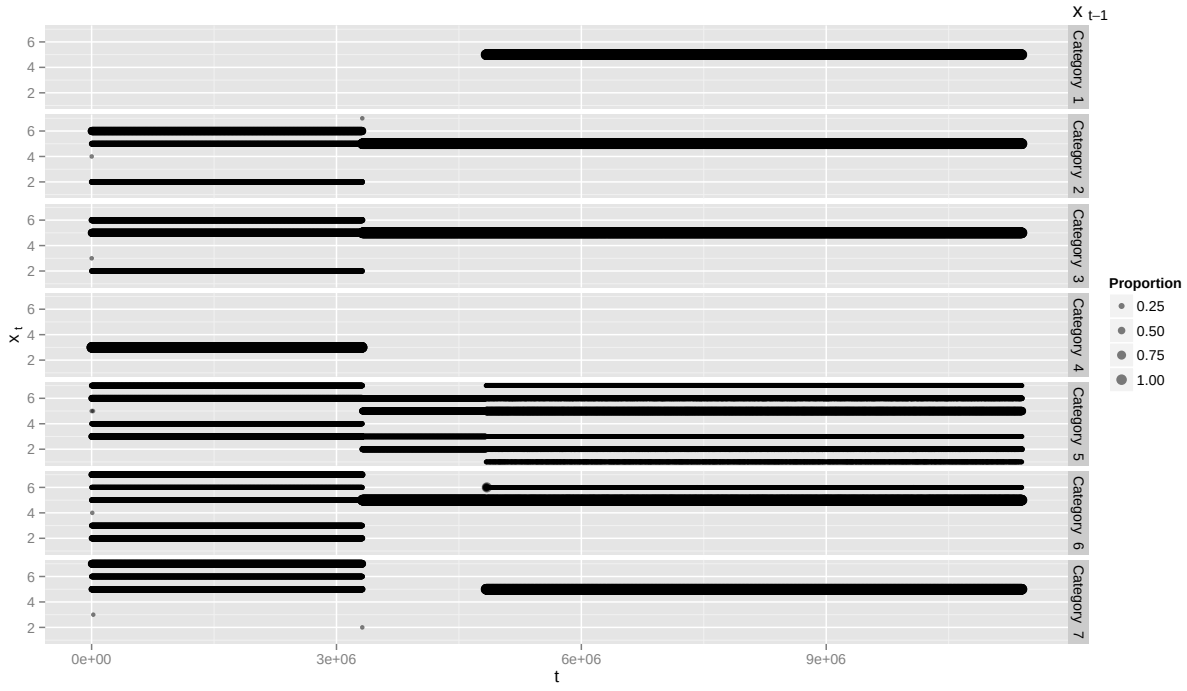


Figure 1: The transition structure of a malware trace. Each block corresponds to an observed instruction category x_{t-1} (labeled on the right hand side), and each point corresponds to an observed transition to x_t. The point size is proportional to the local frequency of each category, using bins of 1000 observations. Category 8 is not observed in this example.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

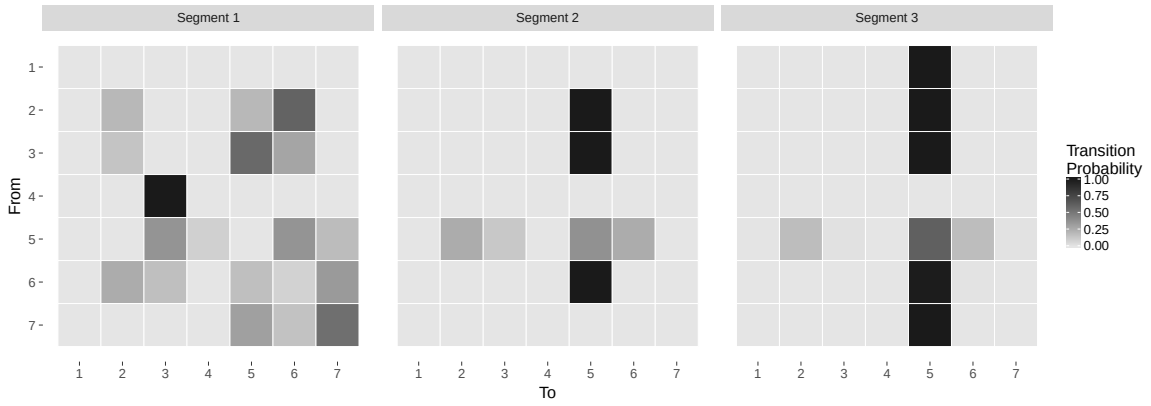


Figure 2: Empirical transition probabilities for the three segments visible in Figure 1.

3 Change Point Models for Dynamic Instruction Traces

3.1 Unmarked change points

For an observed instruction trace $\boldsymbol{x}$ of length n, let $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_k)$ be a vector of $k \geq 0$ change points, ordered such that $1 \equiv \tau_0 < \tau_1 < \ldots < \tau_k < \tau_{k+1} \equiv n + 1$. Change points are assumed to follow a Bernoulli process, implying a joint prior probability mass function $\pi(\boldsymbol{\tau}, k) = p^k(1-p)^{n-1-k}$ for some $0 < p < 1$.

A k-vector of change points $\boldsymbol{\tau}$ partitions the instruction sequence indices $\{1, \ldots, n\}$ into $k+1$ segments, with jth segment $\{\tau_{j-1}, \tau_{j-1}+1, \ldots, \tau_j - 1\}$. Within each segment $\boldsymbol{x}$ is assumed to be a homogeneous Markov chain with corresponding transition probability matrices $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_{k+1}$ governing the instruction sequences observed in each segment, so

$$\boldsymbol{P}(t) = \boldsymbol{P}_j, \quad \tau_{j-1} \leq t < \tau_j.$$

In the simplest Bayesian model, independent, identical conjugate Dirichlet($\boldsymbol{\gamma}$) priors are assumed for each row of each matrix, where $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_c)$ is a c-vector of positive

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

values which could be set informatively to reflect the relative frequencies of the different instruction categories if such prior information were available.

The conjugate Dirichlet priors for the transition matrices allow simple evaluation of the marginal likelihood of the dynamic instruction trace $\boldsymbol{x}$ given a k-vector of change points $\boldsymbol{\tau}$,

$$L(\boldsymbol{x} \mid \boldsymbol{\tau}, k) = \left\{\frac{\gamma_*}{\gamma^*}\right\}^{c(k+1)} \prod_{j=1}^{k+1} \left[\left\{ \frac{1}{\Gamma(\gamma_\cdot + \tau_j - \tau_{j-1} - \mathbb{I}(j=1))} \right\}^c \prod_{i=1}^{c} \prod_{i'=1}^{c} \Gamma\{\gamma_{i'} + n_{j,i,i'}(\boldsymbol{\tau})\} \right],$$

where $\gamma_\cdot = \sum_{i=1}^{c} \gamma_i$, $\gamma_* = \Gamma(\gamma_\cdot)$, $\gamma^* = \prod_{i=1}^{c} \Gamma(\gamma_i)$ and $n_{j,i,i'}(\boldsymbol{\tau}) = \sum_{t=\tau_{j-1}}^{\tau_j - 1} \mathbb{I}(x_{t-1} = i, x_t = i')$ is the number of transitions from instruction i to instruction i' in the jth segment.

3.2 Binary marked change points

In the simplest change point model above, the whole transition probability matrix is assumed to change at each change point. However, with dynamic instruction traces it is often apparent that not all rows of the transition probability matrix $\boldsymbol{P}(t)$ change simultaneously. For an example of this, refer back to Fig. 1; the second 'clear' change point does not have any discernible effect on the transitions from category 2, for example.

For a more realistic model of instruction traces, a generalization of the change point model allows change points to indicate a change to just a subset of the rows of the transition probability matrix. Define a $c \times k$ matrix $\boldsymbol{I}$ for which $I_{i,j} \in \{0, 1\}$ for $1 \leq i \leq c, 1 \leq j \leq k$, such that $I_{i,j} = 1$ if and only if the ith row of $\boldsymbol{P}(t)$ changes at the jth change point τ_j. The jth column of $\boldsymbol{I}$ can therefore be considered as a mark vector associated with the jth change point, indicating which rows of $\boldsymbol{P}(t)$ the change point affects.

Conditional on k, $\boldsymbol{I}$ is assumed to be statistically independent of the change point locations $\boldsymbol{\tau}$, implying the joint prior mass function $\pi(\boldsymbol{I}, \boldsymbol{\tau}, k)$ can be factorized as

$$\pi(\boldsymbol{I}, \boldsymbol{\tau}, k) = \pi(\boldsymbol{\tau}, k)\pi(\boldsymbol{I} \mid k),$$

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

with $(\boldsymbol{\tau}, k)$ still generated by a Bernoulli process.

It is assumed that independently $I_{i,j} \sim \text{Bernoulli}(w_i)$; individual Bernoulli prior parameters are assumed for each row i of $\boldsymbol{I}$, admitting the possibility that some rows of the transition probability matrix will be more changeable than others. Independent, identical conjugate prior distributions, $w_i \sim \text{Beta}(\alpha, \beta)$, are assumed for each $i = 1, \ldots, c$. The unknown Bernoulli parameters can then be integrated out, yielding the prior

$$\pi(\boldsymbol{I} \mid k) = \left\{ \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)\Gamma(\alpha+\beta+k)} \right\}^c \prod_{i=1}^{c} \Gamma(\alpha + k_i)\Gamma(\beta + k - k_i),$$

where $k_i = \sum_{j=1}^{k} I_{i,j}$ is the number of change points in $\boldsymbol{\tau}$ that affect the ith row of $\boldsymbol{P}(t)$.

Define the k_i-vector $\boldsymbol{\tau}(i)$ to be the subsequence of change points $\boldsymbol{\tau}$ which affect the ith row of $\boldsymbol{P}(t)$, such that τ_j is a component of $\boldsymbol{\tau}(i)$ if and only if $I_{i,j} = 1$. Within the segments of $\{1, \ldots, n\}$ created by $\boldsymbol{\tau}(i)$, the ith row of $\boldsymbol{P}(t)$ is constant and assumed to be an independent draw from Dirichlet$(\boldsymbol{\gamma})$, yielding a modified marginal likelihood

$$L(\boldsymbol{x} \mid \boldsymbol{I}, \boldsymbol{\tau}, k) = \left\{ \frac{\gamma_*}{\gamma^*} \right\}^{k_\cdot + c} \prod_{i=1}^{c} \prod_{j=1}^{k_i+1} \left[\frac{1}{\Gamma(\gamma_\cdot + n_{j,i,\cdot}(\boldsymbol{\tau}(i)))} \prod_{i'=1}^{c} \Gamma\{\gamma_{i'} + n_{j,i,i'}(\boldsymbol{\tau}(i))\} \right],$$

where $k_\cdot = \sum_{i=1}^{c} k_i = \sum_{i=1}^{c} \sum_{j=1}^{k} I_{i,j}$ and $n_{j,i,\cdot}(\boldsymbol{\tau}(i)) = \sum_{i'=1}^{c} n_{j,i,i'}(\boldsymbol{\tau}(i))$.

3.3 Regime-switching model

A final elaboration of the model is to incorporate regimes for each row of $\boldsymbol{P}(t)$, where a regime characterizes a recurring pattern of behavior. Under this model, when a change point occurs each row of the transition probability matrix may change to a new vector of Dirichlet probabilities, but may otherwise revert to a vector of probabilities which governed the Markov chain in some earlier segments.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Suppose the ith row of $\boldsymbol{P}(t)$ has r_i underlying regimes, meaning there are only r_i unique, unknown c-vectors of values that the ith row can take, each draws from Dirichlet($\boldsymbol{\gamma}$). Then parametrically, the regime-switching model can be expressed as an extension to the previous marked change point model, where the marks $I_{i,j}$ now take values in $\{1, \ldots, r_i\}$ rather than $\{0, 1\}$; rather than indicate whether a change occurs, the marks $I_{i,j}$ now indicate the regime of the ith row of $\boldsymbol{P}(t)$ in the jth segment. Note that for identifiability, without loss of generality each row of the transition matrix $\boldsymbol{P}(t)$ is assumed to start in regime 1.

Letting $\boldsymbol{r} = (r_1, \ldots, r_c)$, the prior for $(\boldsymbol{I}, \boldsymbol{\tau}, k, \boldsymbol{r})$ is assumed to factorize as

$$\pi(\boldsymbol{I}, \boldsymbol{\tau}, k, \boldsymbol{r}) = \pi(\boldsymbol{\tau}, k)\pi(\boldsymbol{r})\pi(\boldsymbol{I} \mid k, \boldsymbol{r}).$$

The numbers of regimes for each row of $\boldsymbol{P}(t)$ are assumed to be independent, identically distributed Geometric(ρ) random variables, $0 < \rho < 1$, leading to

$$\pi(\boldsymbol{r}) = (1 - \rho)^{r_\cdot - c}\rho^c$$

where $r_\cdot = \sum_{i=1}^{c} r_i$. Conditional on k and r_i, the ith row of $\boldsymbol{I}$, which specifies the sequence of regimes for the ith row of $\boldsymbol{P}(t)$ at each of the change points, is assumed to follow its own Markov chain, with an independent Dirichlet($\delta \cdot \mathbb{1}_{r_i}$) prior for each row of the corresponding $r_i \times r_i$ transition probability matrix, where $\delta > 0$ and $\mathbb{1}_{r_i}$ is a r_i-vector of ones. The implied prior mass function for the regime matrix $\boldsymbol{I}$ is

$$\pi(\boldsymbol{I} \mid k, r) = \prod_{i=1}^{c} \left[\left\{ \frac{\Gamma(r_i\delta)}{\Gamma(\delta)^{r_i}} \right\}^{r_i} \prod_{r=1}^{r_i} \frac{1}{\Gamma(r_i\delta + \iota_{i,r,\cdot})} \prod_{r'=1}^{r_i} \Gamma(\delta + \iota_{i,r,r'}) \right],$$

where $\iota_{i,r,r'} = \sum_{j=1}^{k} \mathbb{I}(I_{i,j-1} = r, I_{i,j} = r')$ is the number of transitions from regime r to regime r', $\iota_{i,r,\cdot} = \sum_{r'=1}^{r_i} \iota_{i,r,r'}$ and $I_{i,0} = 1$ for $i = 1, \ldots, c$.

For each category $i \in \{1, \ldots, c\}$ and each regime $r \in \{1, \ldots, r_i\}$, define the set

$$A_{r,i} = \bigcup_{j:0 \le j \le k; I_{i,j} = r} \{\tau_j, \tau_j + 1, \ldots, \tau_{j+1} - 1\}$$

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

to be the indices of the dynamic instruction trace for which the ith row of $\boldsymbol{P}(t)$ is in regime r. Then the marginal likelihood for the entire trace $\boldsymbol{x}$ under the regime-switching model is

$$L(\boldsymbol{x} \mid \boldsymbol{I}, \boldsymbol{\tau}, k, \boldsymbol{r}) = \left\{\frac{\gamma_*}{\gamma^*}\right\}^{r_\cdot} \prod_{i=1}^{c} \prod_{r=1}^{r_i} \left[\frac{1}{\Gamma(\gamma_\cdot + \tilde{n}_{r,i,\cdot})} \prod_{i'=1}^{c} \Gamma\{\gamma_{i'} + \tilde{n}_{r,i,i'}\}\right], \tag{1}$$

$$\tilde{n}_{r,i,i'} = \sum_{t \in A_{r,i}} \mathbb{I}(x_{t-1} = i, x_t = i'), \tag{2}$$

$$\tilde{n}_{r,i,\cdot} = \sum_{i'=1}^{c} \tilde{n}_{r,i,i'}. \tag{3}$$

4 Comparing Dynamic Instruction Traces

Section 3.3 gave a representation of a single dynamic instruction trace as a piecewise homogeneous Markov chain, where each row of the transition probability matrix is an embedded Markov chain at the sequence of change points, making transitions between an unknown number of regimes. To compare two or more dynamic instruction traces, a joint model of this character is required.

Let $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}$ be the dynamic instruction traces from N pieces of software. Dynamic instruction traces from running different pieces of software will typically have highly variable lengths, and so there is negligible scope in comparing the numbers or positions of change points for the different traces. Instead, to allow joint learning about shared regimes and the proportion of instructions generated by these common regimes, a joint model will assume the same Markov chains of regimes for all instruction traces. This simple extension implies an equivalent likelihood function to (1), with the exception that the count $\tilde{n}_{r,i,i'}$ is now taken to be a sum of the previous quantity (2) across the change points from all of the

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

traces,

$$\tilde{n}_{r,i,i'} = \sum_{s=1}^{N} \tilde{n}^{(s)}_{r,i,i'},$$
$$\tilde{n}^{(s)}_{r,i,i'} = \sum_{t \in A^{(s)}_{r,i} \setminus \{1\}} \mathbb{I}(x^{(s)}_{t-1} = i, x^{(s)}_{t} = i'),$$
$$A^{(s)}_{r,i} = \bigcup_{j: 0 \le j \le k; I^{(s)}_{i,j} = r} \{\tau^{(s)}_j, \tau^{(s)}_j + 1, \ldots, \tau^{(s)}_{j+1} - 1\}$$

where $\boldsymbol{\tau}^{(s)}$ and $\boldsymbol{I}^{(s)}$ are the change points and marks for the instruction trace $\boldsymbol{x}^{(s)}$. Note that when jointly modeling multiple traces, identifiability only requires that the rows of the transition matrix for the first trace, $\boldsymbol{P}^{(1)}(t)$ must start in regime 1; all other traces should be free to start in any regime, and here the ith row of $\boldsymbol{P}^{(s)}$ starts in a regime drawn uniformly from $\{1, \ldots, r_i\}$, for each $s \in \{2, 3, \ldots, N\}$.

For each trace $s \in \{1, \ldots, N\}$, category $i \in \{1, \ldots, c\}$ and regime $r \in \{1, \ldots, r_i\}$, let $\tilde{n}^{(s)}_{r,i,\cdot}$ be the corresponding quantity (3) for the dynamic trace $\boldsymbol{x}^{(s)}$, counting the number of instructions observed from the ith row of $\boldsymbol{P}^{(s)}(t)$ under regime r. Now suppose $(\boldsymbol{\tau}^{(s_1)}, \boldsymbol{I}^{(s_1)})$ and $(\boldsymbol{\tau}^{(s_2)}, \boldsymbol{I}^{(s_2)})$ are change points and regime sequences inferred for two instruction traces of lengths n_{s_1} and n_{s_2}. The similarity of the two vectors of change points and their regimes for the two software sample instruction traces will be defined as

$$S_{s_1,s_2}(\boldsymbol{\tau}^{(s_1)}, \boldsymbol{I}^{(s_1)}, \boldsymbol{\tau}^{(s_2)}, \boldsymbol{I}^{(s_2)}) =$$
$$\min \left\{ \frac{1}{n_{s_1}} \sum_{i=1}^{c} \sum_{r=1}^{r_i} \mathbb{I}(\tilde{n}^{(s_1)}_{r,i,\cdot} > 0, \tilde{n}^{(s_2)}_{r,i,\cdot} > 0) \tilde{n}^{(s_1)}_{r,i,\cdot}, \frac{1}{n_{s_2}} \sum_{i=1}^{c} \sum_{r=1}^{r_i} \mathbb{I}(\tilde{n}^{(s_1)}_{r,i,\cdot} > 0, \tilde{n}^{(s_2)}_{r,i,\cdot} > 0) \tilde{n}^{(s_2)}_{r,i,\cdot} \right\}. \quad (4)$$

This similarity measure calculates the minimum for the two samples of the proportions of instructions which occur within regimes shared by both traces, as determined by the indicator function. For high similarity, a large proportion of observations in both traces need

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

to be drawn from regimes that are common to both traces, ensuring that this is a robust estimator of similarity. Figure 3 shows three cartoon examples of similarity calculations for simplified pairs of traces. (Note that in practice, elements of the calculations would be summed across c transition categories according to (4).)

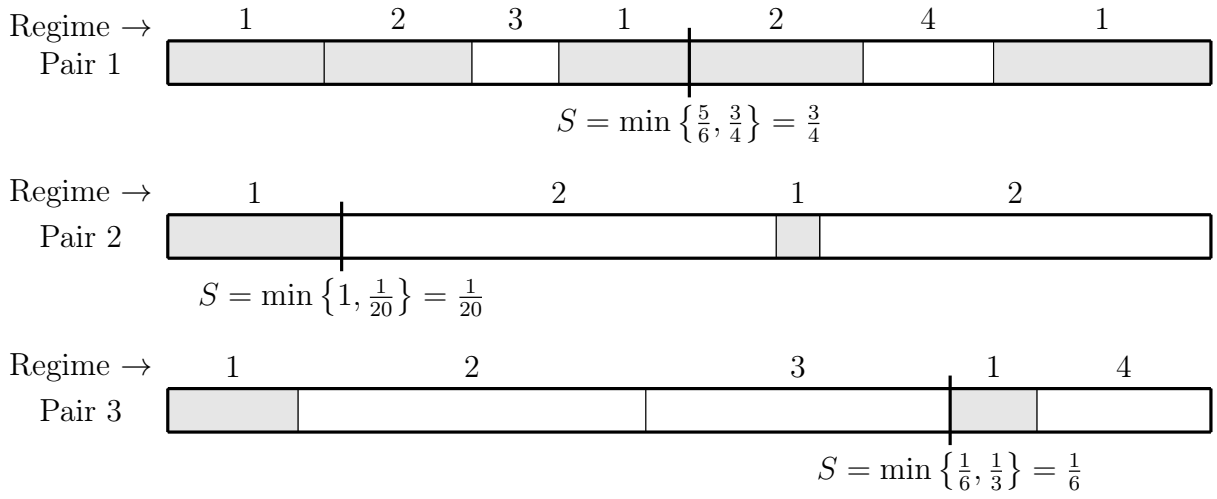


Figure 3: Three examples of the calculation of the similarity measure S_{s_1,s_2} for two traces given a set of regimes and change points. The thick vertical lines separate the two sample traces being compared. Shaded segments represent regimes common to both traces.

To accommodate the uncertainty surrounding the change points and marks, the overall similarity of two traces s_1, s_2 will be the posterior expectation of S_{s_1,s_2} from (4); this will be calculated by model averaging over samples obtained from a reversible jump MCMC (RJMCMC) sampler, extending the change point analysis of Green (1995).

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

5 Sampling Change Points via Reversible Jump MCMC

5.1 Unmarked change points

For the simplest case described in Section 3.1 where all transition probability rows change at each change point, following Denison et al. (2002) one of three move types are proposed at each RJMCMC iteration. A new change point is proposed uniformly from $\{2, \ldots, n\} \backslash \boldsymbol{\tau}$, or else an existing change point is uniformly selected from $\boldsymbol{\tau}$ and either deleted or shifted to a new position.

5.2 Binary marked change points

To extend the sampler to include binary marked vectors (*cf.* Section 3.2) each of the three previous moves is extended to sampling associated binary mark vector with each change point. If a new change point τ'_h is proposed, for example, then the new column of $\boldsymbol{I}, \boldsymbol{I}'_{\cdot,h}$, is sampled from the full conditional,

$$I'_{i,h} \sim \text{Bernoulli}\left(\frac{(\alpha + k_i)B_{i,h}}{(\alpha + k_i)B_{i,h} + (\beta + k - k_i)}\right) \text{ for } i \in \{1, \ldots, c\},$$

where $B_{i,h}$ is the Bayes factor for adding a change point to process i at position τ'_h with $I'_{i,h} = 1$. Defining $\boldsymbol{\tau}'(i)$ to be $\boldsymbol{\tau}(i)$ with τ'_h inserted,

$$B_{i,h} = \frac{\gamma_*}{\gamma^*} \frac{\prod_{j=h}^{h+1} [\{\prod_{i'=1}^{c} \Gamma(\gamma_{i'} + n_{j,i,i'}(\boldsymbol{\tau}'(i)))\}/\Gamma(\gamma_\cdot + n_{j,i,\cdot}(\boldsymbol{\tau}'(i)))]}{\{\prod_{i'=1}^{c} \Gamma(\gamma_{i'} + n_{h,i,i'}(\boldsymbol{\tau}(i)))\}/\Gamma(\gamma_\cdot + n_{h,i,\cdot}(\boldsymbol{\tau}(i)))}.$$

Additionally, a similar proposal move is introduced for resampling an existing mark vector.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

5.3 Regime-switching model

When adopting regimes (*cf.* Section 3.3) the mark vector sampling must be suitably adapted. When a regime vector is proposed for a new or existing change point, for each element there must be opportunities to either revert to an existing regime or switch to a new regime. If, as a consequence of deleting a change point or resampling a regime vector, a regime no longer features in $\boldsymbol{I}_{i,\cdot}$, then this becomes an *unobserved regime*. For computational simplicity, additional proposals for the RJMCMC sampler are required to propose adding/deleting unobserved regimes. These proposals are made infrequently because the number of unobserved regimes will have most probability at zero.

When a new change point τ'_h is proposed, an updated regime vector $\boldsymbol{I}'_{i,\cdot}$ must be sampled for each row i. For regime r, define $\boldsymbol{I}'_{i,\cdot}(r)$ to be $\boldsymbol{I}_{i,\cdot}$ with $I'_{i,h} = r$ inserted at τ'_h. Let $B_{i,h}(r)$ be the Bayes factor associated with setting $I'_{i,h} = r$, and $r'_i(r) = r_i + \mathbb{I}(r = r_i + 1)$ be the number of observed regimes. Then the full conditional distribution is

$$\Pr(I'_{i,h} = r) \propto \pi(\boldsymbol{I}'_{i,\cdot}(r) \mid k+1, r'_i(r)) B_{i,h}(r)(1-\rho)^{\mathbb{I}(r=r_i+1)}, \tag{5}$$

for $i \in \{1, \ldots, c\}, r \in \{1, \ldots, r_i + 1\}$. If a change point is deleted and that deletion causes regime r_i for row i becomes unobserved, then regime r_i is removed from the available regimes. This is necessary in order to ensure that a move to add a change point and a new regime is reversible. A consequence of this is that in the edge case of (5), where $r = r_i$ and regime r_i is unobserved for row i, then $\Pr(I'_{i,h} = r) = 0$.

For each row i, (5) must be computed $r_i + 1$ times. Hence, when any marked vector is added, removed or resampled, $\sum_{i=1}^{c}(r_i + 1)$ computations are required. The total computation time for the regime marked change point sampling is then $\mathcal{O}(Mc\bar{r})$, where M is the number of iterations and $\bar{r}$ is the average number of regimes in a process.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

5.4 Burn-in strategy

Change point and regime inference will be performed jointly on N dynamic instruction traces. If the burn-in for the sampler begins by immediately sampling change points with regimes then the mixing of the sampler can be poor. This problem can be alleviated by introducing a three-stage burn-in: For the first third of the burn-in the change points are assumed to affect all transition probability matrix rows; binary marked vectors are introduced in the next third, and in the final third of the burn-in the full regime-switching model is sampled. In the early stages of burn-in the sampler is able to establish sensible positioning of change points without the overhead of also detecting the corresponding regimes. This helps the sampler avoid local maxima which can be entered by regime mislabeling at an early stage.

6 Application to Malware Traces

In this section the RJMCMC sampler is first applied to a single piece of malware in order to show how it can be segmented with regimes. Second, a simulation study is carried out to investigate whether the sampler is able to reliably estimate the similarity measure (4) for traces. Third, a prior sensitivity analysis is conducted. Fourth and finally, a method for classifying malware traces into families without using change points is examined, to contrast its results with those the full regime-switching change point model; this leads to consideration of a hybrid strategy, which delivers the best performance.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

6.1 Regime Change Points for a Single Dynamic Instruction Trace

The regime detection methodology was first applied to the dynamic instruction trace of a single piece of malware provided by Los Alamos National Laboratory. The trace contains $n = 495{,}210$ instruction transitions and so for simplicity the prior used for the change points was a Bernoulli$(1/n)$ process, suggesting a prior expectation of just a single change point. The regime Markov chain Dirichlet parameter δ was set to 1 and the geometric parameter ρ for the number of regimes for each state was 0.9, together ensuring a parsimonious collection of regimes and change points. The Dirichlet parameters $\boldsymbol{\gamma}$ for $\boldsymbol{P}(t)$ were all set to 1 since there was no information about which transitions were likely to be more common. The RJMCMC sampler was run with a burn-in of 3×10^6 iterations, followed by 5×10^6 retained iterations. The burn-in iterations were split evenly between the unmarked, binary and regime change point models as described in Section 5.4. The sampler took 289 seconds to run on a 2.6GHz Intel Core i7 processor.

The maximum a posteriori (MAP) change points and regimes from RJMCMC are displayed in Figure 4, whilst Figure 5 shows an enlargement of a region that contains several short segments. Note that one of the eight instruction categories is not observed.

Figure 6 shows the posterior probabilities that instructions were generated from the same regime, for 1,000 equally spaced points in the trace sequence. It can be seen that some category transitions, such as those following an instruction from category 5, have clear regimes.

6.2 Trace Similarity Simulation Study

A simulation study was conducted to investigate the reliability of the RJMCMC sampler for estimating the similarity measure (4) between a pair of traces. 1000 pairs of traces

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

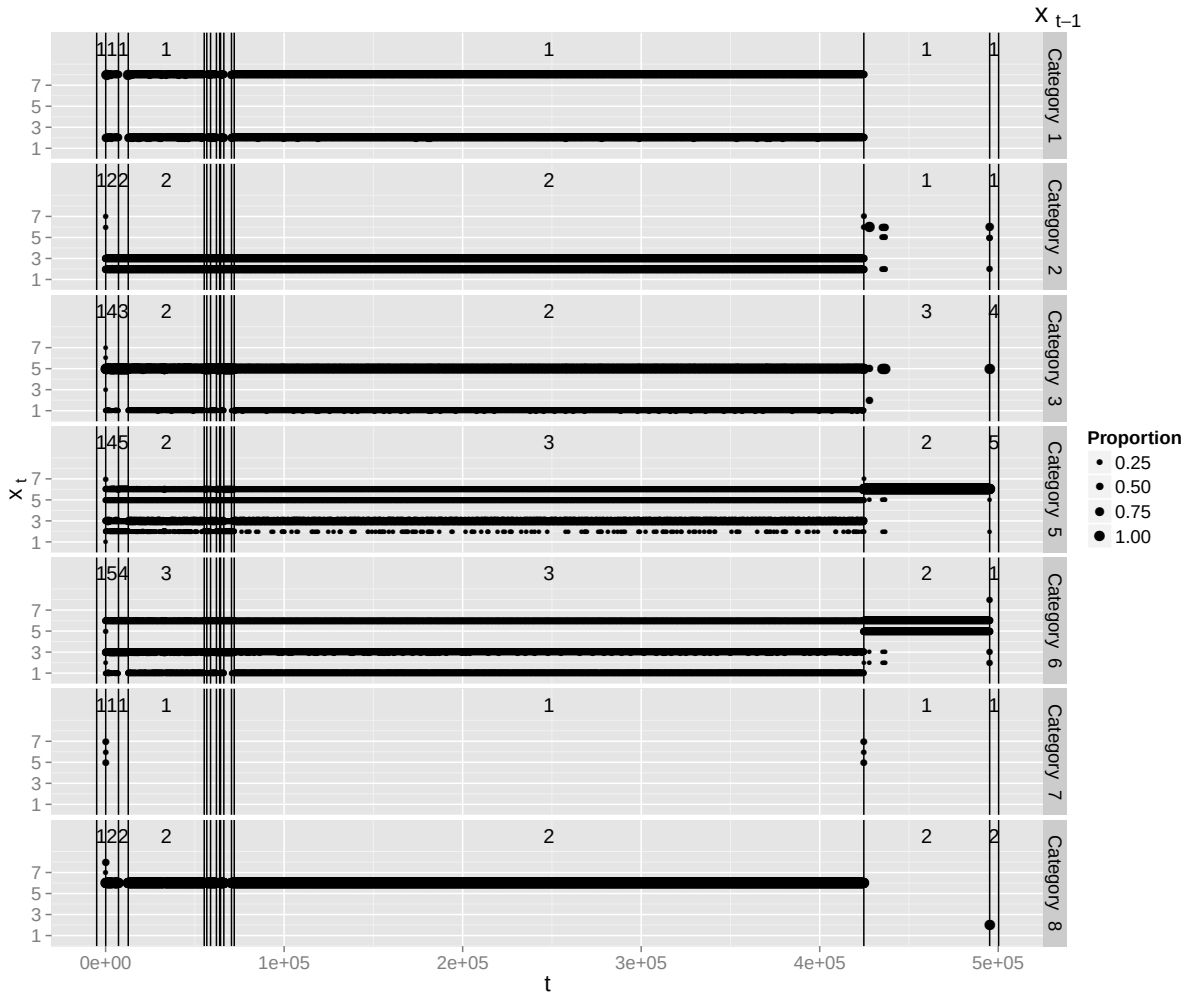


Figure 4: Plot of instruction transitions. The size of the points represents the proportions of transitions observed in each instruction category. Regimes numbers are labeled in each segment. Instruction category 4 is never observed.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

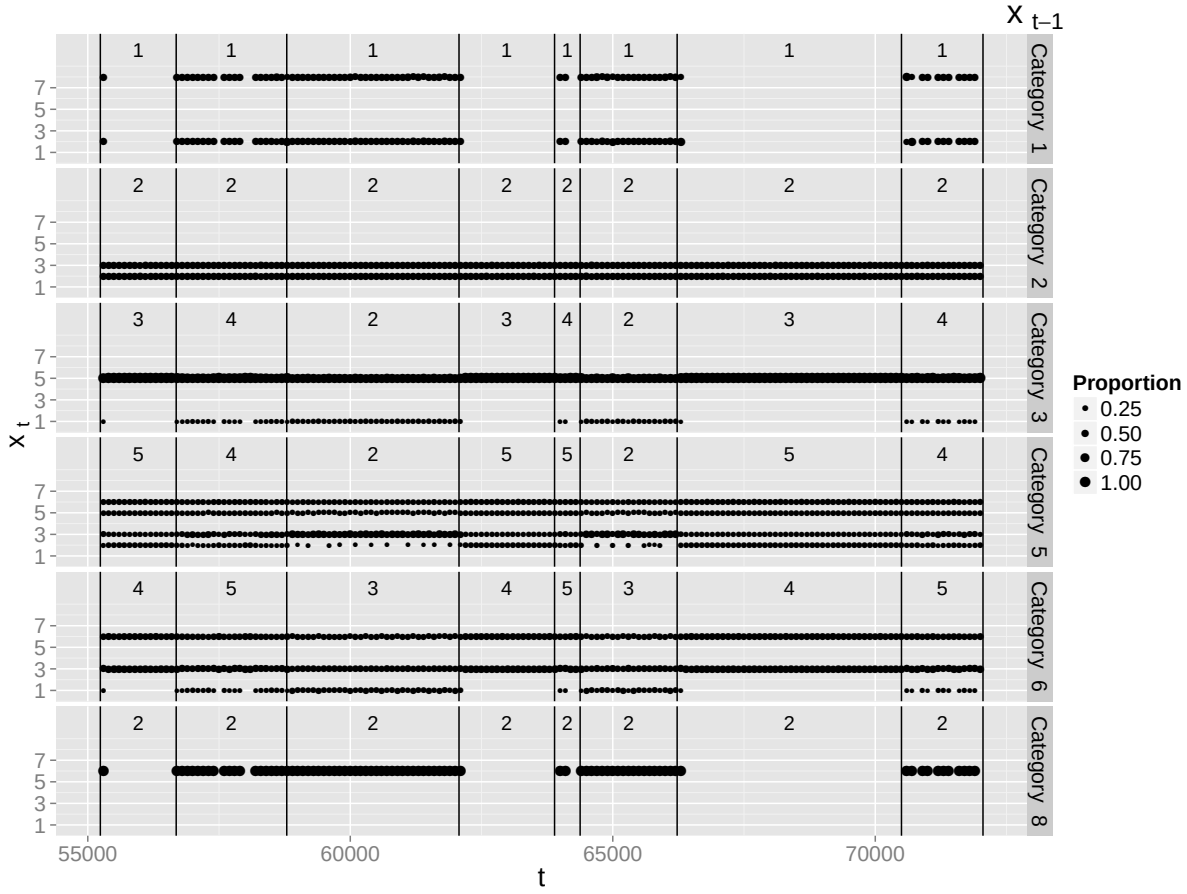


Figure 5: Enlargement of the region that undergoes many change points in Figure 4. Regimes are labeled at the top of each segment.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

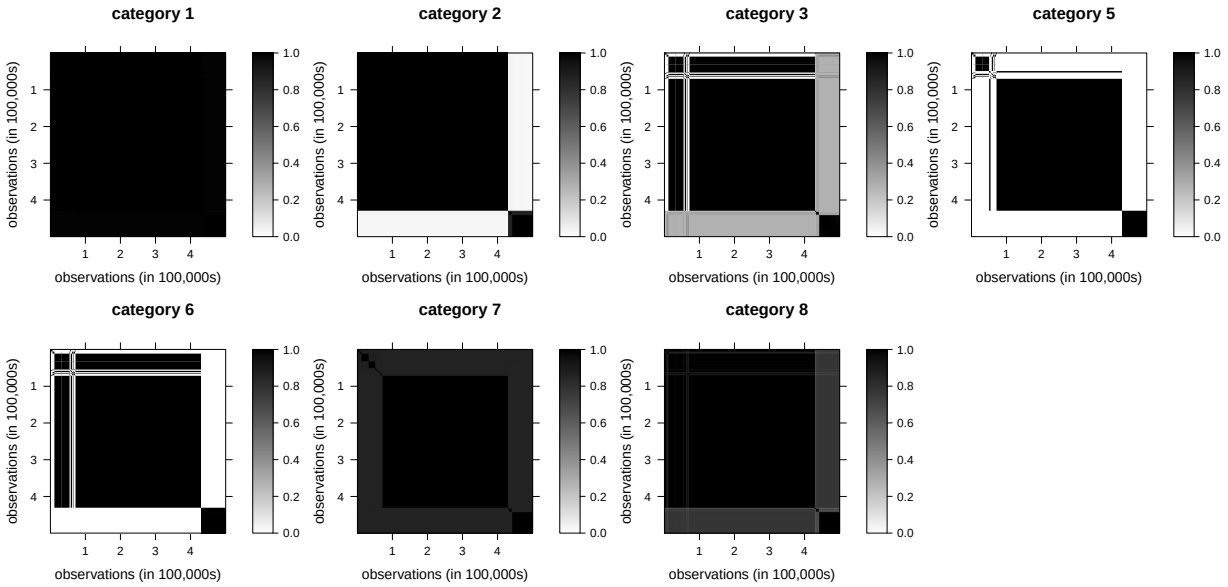


Figure 6: Associations between the times for different instructions. The scale indicates the posterior probability that a pair of points are generated from the same regime.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

were simulated from the model in Section 3.3 using the regime model parameters: $n = 10{,}000, p = 5/10{,}000, \gamma_i = 1, c = 4, \delta = 1, \rho = 0.2$. For performing inference, the prior parameters for the sampler were the true parameters, with the exception of p, which was set to be 1/20,000 to ensure a parsimonious set of change points.

For each pair of traces, change points minimizing the BIC were found using PELT to provide an initial change point configuration. Then the RJMCMC sampler was run with a burn-in of 1.5×10^7 iterations, split evenly between the three change point models, as described in Section 5.4. After this extended burn-in, the length of the final RJMCMC sampling was 5×10^6 iterations. Since (4) is relatively expensive to calculate, the samples were thinned at a rate of one per 1000; this is still sufficient for convergence of posterior estimates, illustrated by diagnostic plots in Figure 7. The full inference procedure took an average of 591 seconds per simulated pair on a 2.6GHz Intel Core i7 processor.

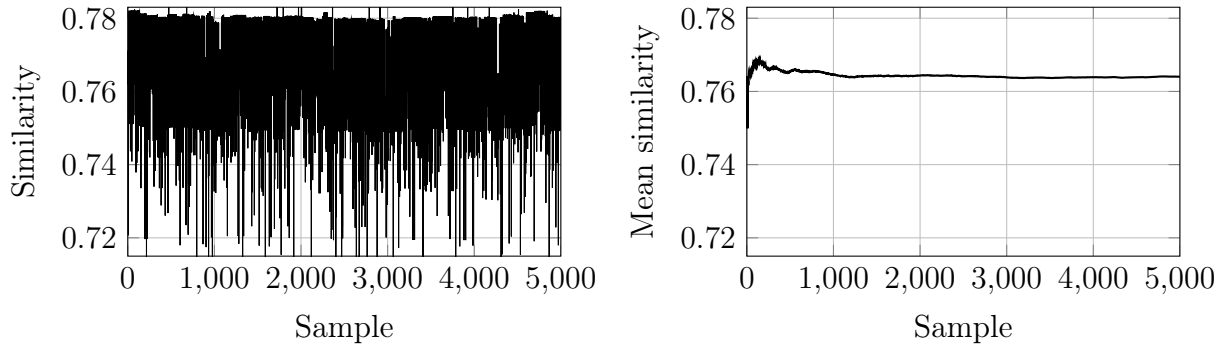


Figure 7: Sample value (left) and convergence of posterior mean (right) for the estimated similarity of an example pair of simulated instruction traces.

The true similarity between pairs of traces was calculated by applying (4) to the true change points and regimes that generated each trace. Figure 8 plots the posterior mean esti-

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

mated similarities against the true similarities. It is clear that the estimation is performing well, with 91% of the estimates lying within 5% of the true value.

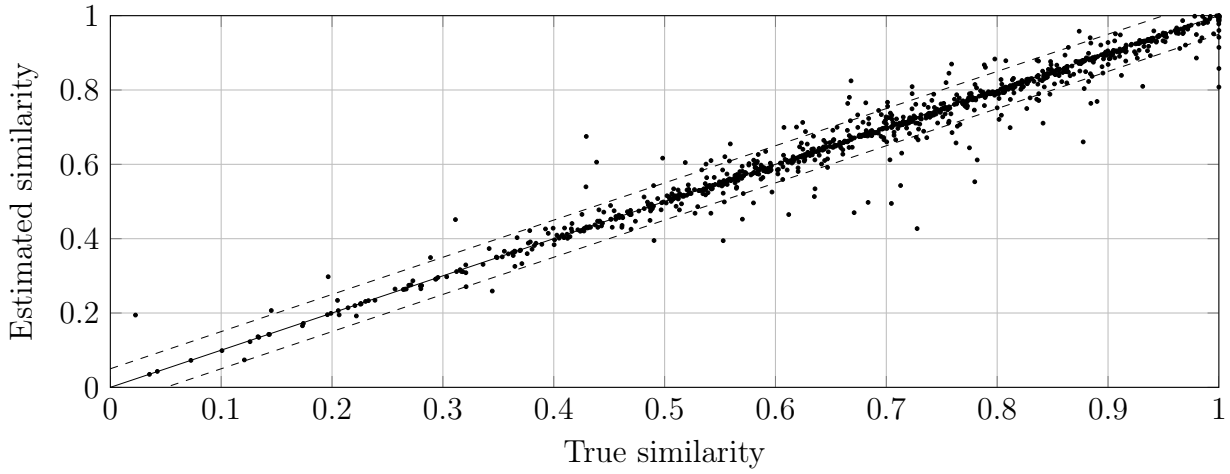


Figure 8: The relationship between the actual similarity of a pair of traces and the mean estimated similarity. The solid line represents exact estimation, and the dashed lines represent a 5% error level.

6.3 Prior Sensitivity Analysis

To examine the sensitivity of analysis to the choice of the prior distribution parameters, we take a more detailed look at three particular malware samples: two were versions of the so-called "Bagle" family and one was from a different family called "Newsreels". The Bagle worm was a computer worm that created a large botnet, whilst malware programs from the Newsreels family are capable of uploading and downloading files, as well as executing programs on exploited computers (Contagio, 2013). The two Bagle worm traces, bagle.g and bagle.k were compared using the RJMCMC sampler with the same numbers of iterations

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

and thinning as used in the simulation study in Section 6.2. The similarity of each pair was estimated by calculating (4) at each retained iteration. Then two samples from different families, bagle.g and the Newsreels sample, were also compared. Different, plausible values for the parameters p and ρ were tried: letting m be the length of the combined trace, the tested values for p were $0.5/m, 1/m$ and $5/m$, and the values for ρ were $0.5, 0.7$ and 0.9. As the value of p increases, the implicit penalty for adding a change point is lowered, and as ρ increases, the penalty for adding new regimes is increased.

For all parameter choices, the two samples from the Bagle family were judged to be extremely similar according to definition (4), with estimated similarity always inside $[0.999, 1]$. For each of the nine parameter combinations, the effective sample size (ESS) (Kass et al., 1998) of the similarity measures from the 5000 retained RJMCMC samples was calculated; the average ESS was 2657. The range of the mean similarities for the nine parameter combinations when comparing the bagle.g and the Newsreels traces was $[0.016, 0.040]$, and the average ESS was 3012. Figure 9 demonstrates the convergence of the estimated similarity of bagle.g and the Newsreels for an example pair of parameter settings.

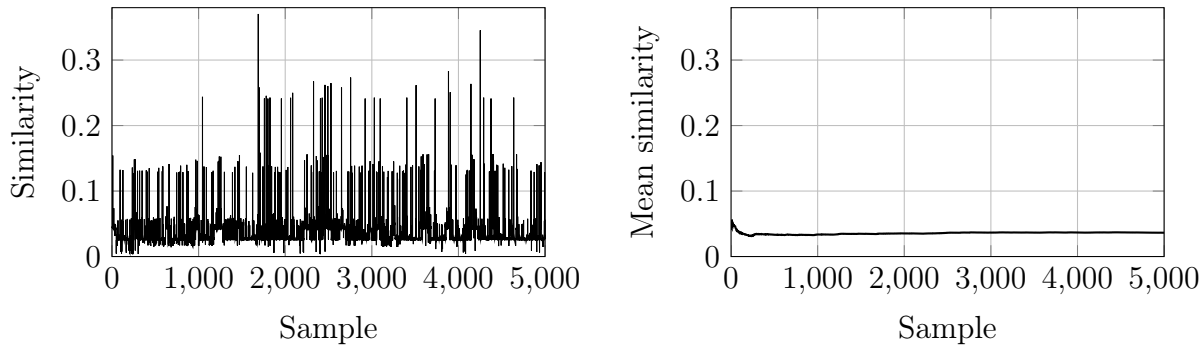


Figure 9: Sample value (left) and convergence of posterior mean (right) for the estimated similarity of the "bagle.g" and "Newsreels" malware instruction traces, $p = 0.05/m, \rho = 0.5$.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Figure 10 shows the posterior distributions of the similarity of these two traces under different prior assumptions, and the differences are relatively small. The average runtime for a pair of malware traces was 787 seconds on a 2.6GHz Intel Core i7 processor.

6.4 Malware Family Classification Without Change Points

To compare the proposed regime change point methodology with the existing state-of-the-art in malware comparison (Anderson et al., 2014), consider the following kernel method for estimating malware families when the traces are assumed to have been generated by a homogeneous Markov chain with a single transition probability matrix.

Viewing the instruction traces as c-state Markov chains, the empirical probability distribution of transitions between each pair of categories can be calculated for each trace: For a trace ℓ of length $(n+1)$, if $n_{i,i'}$ transitions are observed from category i to category i', $i, i' \in \{1, \ldots, c\}$, the empirical probability $p_\ell(i, i') = n_{i,i'}/n$. The distance between any pair of malware samples is then calculated as the squared-exponential kernel distance of their corresponding empirical probability distributions: For two probability distributions p_1, p_2 on a countable set $\mathcal{X}$, their squared-exponential kernel distance is

$$KD(p_1, p_2) = \exp\left\{\sum_{x \in \mathcal{X}} |p_1(x) - p_2(x)|^2\right\}.$$

Removing just one sample label at a time, nearest neighbor classification was performed to estimate the malware family of each trace, where each piece of malware was assigned to the class of the sample that was the most similar in KD.

This article analyses 141 malware instruction traces categorized into malware families by reverse engineers from Los Alamos National Laboratory. The malware families were also further divided into sub-families; for example, the "Tarsip" malware family contains two sub-families, "Tarsip-Moon" and "Tarsip-Eclipse". For $c = 8$ instruction categories, the

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

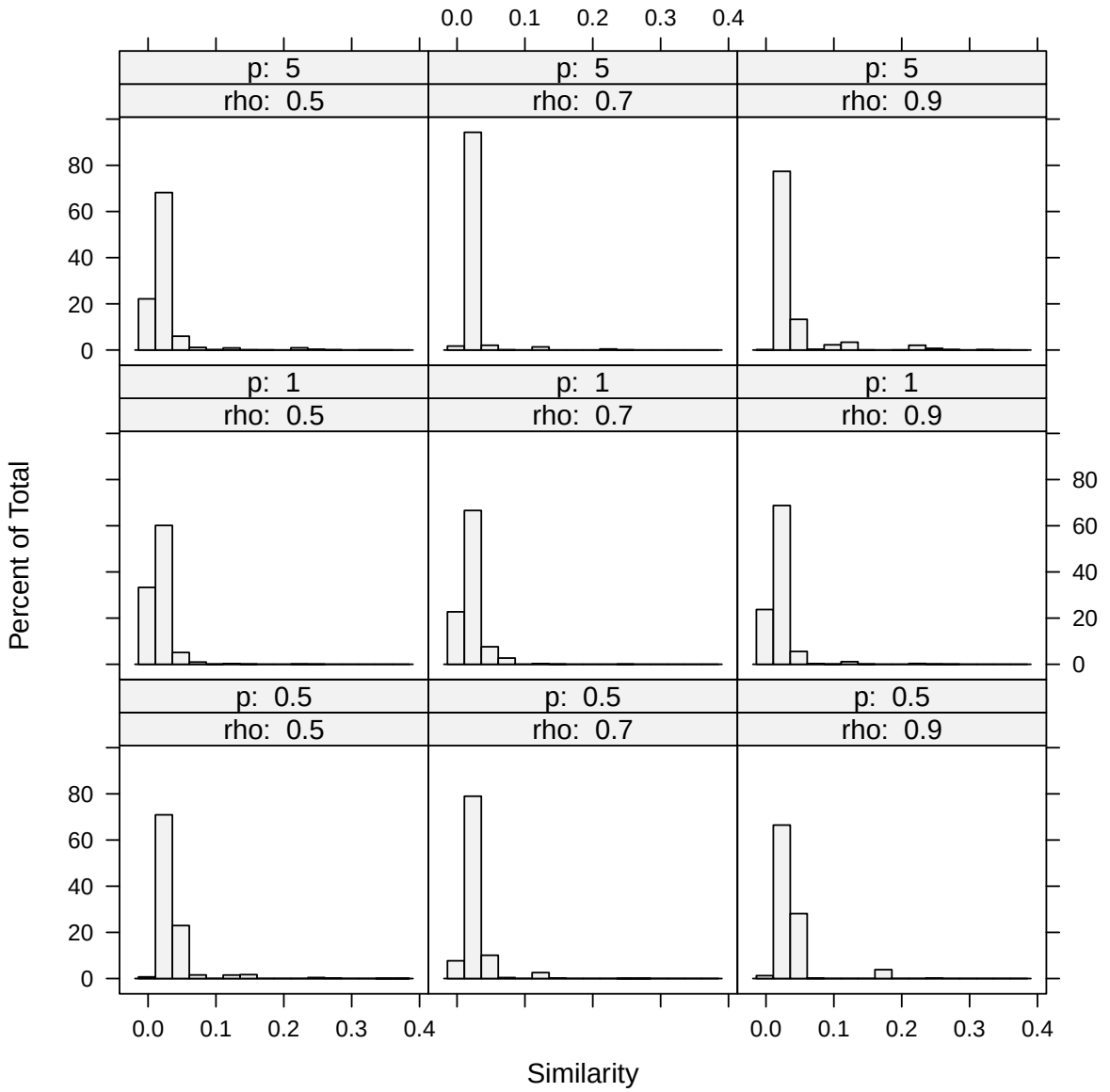


Figure 10: Histograms of the distribution of the similarity (4) between two samples from the Bagle and Newsreels families, under different prior assumptions.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

accuracy rate from this approach was 91% for malware families and 85% for sub-families. The average running time to calculate the kernel distance between a pair of samples was less than a second on a 2.6GHz Intel Core i7 processor.

6.5 Malware Family Classification With Regime Change Points

The methodology from Section 5 was applied to the same collection of malware samples analyzed in Section 6.4. For each pair of samples the similarity was computed by estimating the expected value of the regime similarity measure (4). The RJMCMC sampler was run on the joint model for all of the malware samples with a Bernoulli$(1/m)$ process as a prior for $(\boldsymbol{\tau}, k)$, where m is the number of instructions in total.

132 independent RJMCMC samplers with different random seeds were run. Each run was initialized with the change points minimizing the BIC, found using PELT (Killick et al., 2012); then followed a burn-in of 3×10^7 iterations, split evenly between the unmarked, binary and regime change point models as described in Section 5.4; finally, 5×10^7 further iterations from the regime change point model were obtained, and a thinned sample of 5000 were retained from each run for performing inference. From these independent runs, the standard errors of the expected similarity (4) was calculated; across malware pairs, 89% of the standard errors were below 0.05 and 99.5% of standard errors were below 0.1, suggesting the expected similarities are being estimated reasonably accurately.

The diagnostic of Brooks and Gelman (1998) was also applied to check whether the samplers had converged. For each estimated pairwise similarity S_{s_1,s_2}, the within-sequence and between-sequence variances were estimated from the runs. A weighted average of these two estimated variances is used to estimate the posterior marginal variance of each S_{s_1,s_2}, denoted $\hat{V}_{s_1,s_2}$. Denoting the within-sequence variance W_{s_1,s_2}, the potential scale reduction factor (PSRF) is given by $\sqrt{\hat{V}_{s_1,s_2}/W_{s_1,s_2}}$. Brooks and Gelman (1998) states that if the

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

PSRF is large then either the estimated posterior marginal variance can be reduced or that further simulations will increase the within-sequence variance, as the chain has yet to converge, whereas if the PSRF is close to 1 then each of the runs is close to the target distribution. For the parameter estimates in each of the chains, the median PSRF was 1.326, and 90% of PSRFs were below 1.6, showing that the PSRFs were generally very close to 1, providing evidence that the RJMCMC chains reached a good level of convergence. The effective sample size was also calculated for the 5000 retained samples for each of the pairwise similarity parameters; the mean and median effective sample sizes were 2029 and 1476.

Single link clustering was performed using the posterior mean estimates of the similarity measure (4) to generate a dendrogram, shown in Figure 11. Malware families can be seen to be clustered together well. Single-link clustering is natural here because each sample is likely to be the direct adaptation of a single other member of a malware family, rather than being generated by the family as a whole, as discussed in Anderson et al. (2014).

Nearest-neighbor classification was also performed using the similarity measure (4) and the accuracy rate was found to be 91% for malware families and 84% for sub-families. These figures are similar to those from the simple homogeneous Markov chain approach from Section 6.4, but around 7% of the samples which are correctly classified by family by one of the methods are classified incorrectly by the other. This suggests there is potential to improve the classification results by combining the two approaches.

6.6 Combined Classifier

A hybrid of the two classification methods from Sections 6.4 and 6.5 is now proposed, which leads to improved classification results. In this combined classifier, the kernel distance method, which is trivial to compute, is first used as a filter on the database, and then the

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

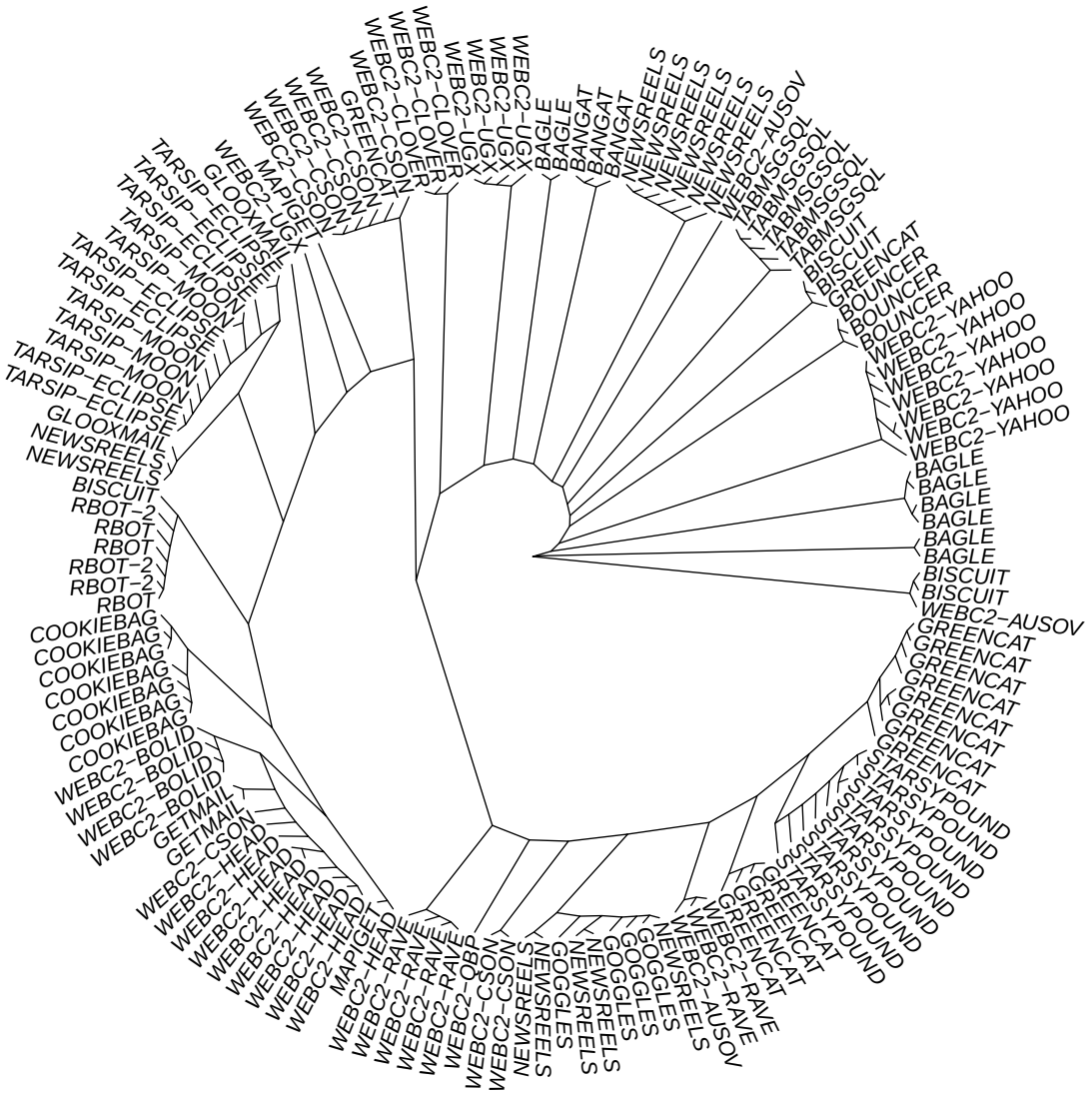


Figure 11: Dendrogram from single-link clustering of estimated malware similarities. Each malware sample is labeled by its malware sub-family.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

more computationally demanding regime comparison method is run on the filtered samples.

To compare an unlabeled trace to the database, a filtered shortlist is created comprising all database samples for which the kernel distance from the new trace is less than 1.05. Empirically, from the analysis from Section 6.4, samples shortlisted in this way are six times more likely to belong to the same family as the new trace than samples that are not shortlisted. The unlabeled trace is then compared to the traces in this shortlist using the regime comparison method of Section 6.5. For each pairwise comparison, the kernel distance and regime similarity are combined to form a single score of dissimilarity using Fisher's method for combining p-values (Fisher, 1929); letting p_1 be the proportion of pairs of traces in the database with kernel distance less than the divergence of the proposed pair, and p_2 be the proportion of trace pairs with change point regime similarity greater than that for the proposed pair, these are combined into a single, approximate p-value given by $\Pr\{\chi^2_4 > -2(\ln p_1 + \ln p_2)\}$.

The new sample is assigned to the family of the trace which attains the lowest value from Fisher's method. This combined method performed better than either individual classifier: the correct classification rate for the combined classifier was 94% for families and 88% for sub-families.

In a two-class classification setting, the Area Under the Curve (AUC) gives the probability that a randomly selected positive case will be ranked above a randomly selected negative case. Identifying malware families is a multiclass classification problem, and so an averaged AUC value can be calculated using the framework of Domingos and Provost (2000), by first calculating each family AUC curve in turn (for example, classifying whether malware stems from the "Bagle" family, or not) and then taking a weighted sum, where the weights are proportional to the number of members of each family. Table 1 compares all the results from the three classification methods. The weighted AUC for the combined method

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Method	Sub-Family Accur.	Family Accur.	Sub-Family AUC	Family AUC
Kernel Dist.	85%	91%	0.92	0.95
Regime Simil.	85%	89%	0.92	0.93
Kernel-Regime	88%	94%	0.93	0.96

Table 1: Three most similar samples to AURIGA-6B31344B40 in the database.

is 0.96 for determining malware families and 0.93 for determining sub-families, meaning the hybrid classifier is performing best by all measurements. In all measures, the combined classifier outperforms the regime similarity classifier - in particular the respective family classification accuracies are 94% and 89% respectively.

Figure 12 shows the standardized confusion matrix, with columns showing the predicted malware sub-family classes and rows showing the true classes. The classification is mostly accurate, with many dark cells on the diagonal. Some of the misclassifications are between similar families, for example all "Webc2-QBP" samples are misclassified as "Webc2-CSON". These two sub-families have similarities; Contagio (2013) states that samples from both sub-families are designed to receive instructions from the malware designer in similar ways, and both are *backdoors*, designed to circumvent normal authentication procedures.

A significant advantage of this hybrid method is that only a subset of plausible candidate malware samples are used for comparison. This provides considerable computational savings, as well as improving the overall accuracy of the classifier.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

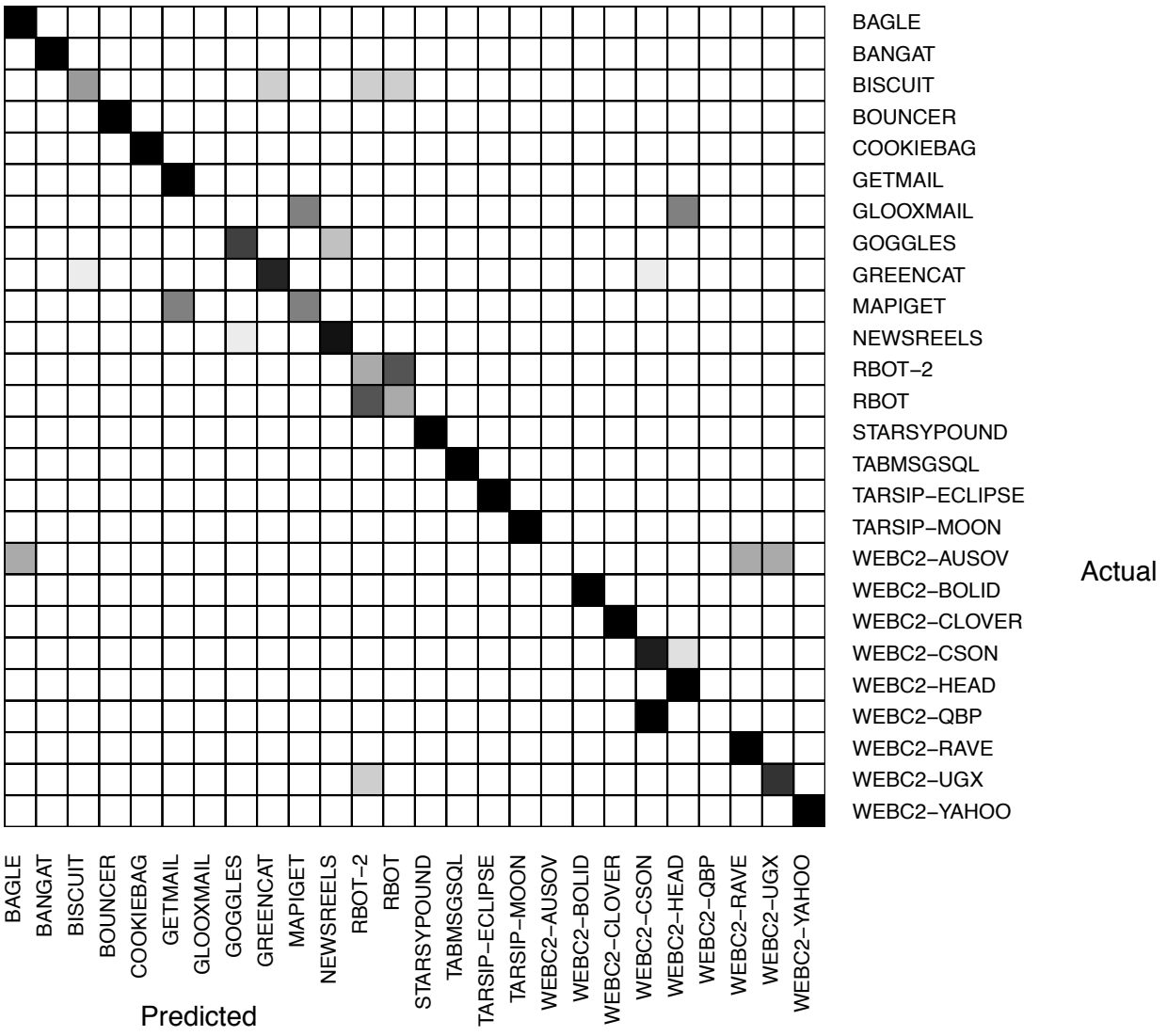


Figure 12: Heat map of the confusion matrix for the combined method. Shading in a cell is proportional to the number of row observations contained within the cell.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

7 Conclusion

This article provides a method for comparing the similarity of dynamic instruction traces to help reverse engineers identify new malicious software. The method uses RJMCMC sampling to detect change points and regimes within dynamic instruction traces. The method can be widely applied in the context of malware detection, as the dynamic trace can almost always be obtained. The RJMCMC sampling scheme itself is novel, as regime detection adds an extra layer of difficulty and there are no established methods in the literature for detecting regimes in multivariate processes that share change points.

For the task of attributing new pieces of malware to malware families, a regime change point similarity measure was shown to improve the classification performance when combined with the current state-of-the-art. Perhaps more importantly, regime-based inference offers the reverse engineer an additional insight into the runtime functions of malware.

8 Supplementary Material

The code for implementing the RJMCMC sampler described in Section 5 is available at `https://github.com/AlexanderDBolton/Regimes_RJMCMC`, with user instructions found in the README file.

References

Anderson, B., T. Lane, and C. Hash (2014). Malware phylogenetics based on the multiview graphical lasso. In H. Blockeel, M. van Leeuwen, and V. Vinciotti (Eds.), *Advances in Intelligent Data Analysis XIII*, Volume 8819 of *Lecture Notes in Computer Science*, pp. 1–12. Springer International Publishing.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Beal, M. J., Z. Ghahramani, and C. E. Rasmussen (2002). The infinite hidden Markov model. In *Machine Learning*, pp. 29–245. MIT Press.

Bolton, A. and N. Heard (2014). Application of a linear time method for change point detection to the classification of software. In *Intelligence and Security Informatics Conference (JISIC), 2014 IEEE Joint*, pp. 292–295.

Brooks, S. P. and A. Gelman (1998). General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics 7*(4), 434–455.

Chib, S. (1996). Calculating posterior distributions and modal estimates in Markov mixture models. *Journal of Econometrics 75*(1), 79 – 97.

Contagio (2013). Mandiant APT1 samples categorized by malware families. http://contagiodump.blogspot.co.uk/2013/03/mandiant-apt1-samples-categorized-by.html.

Dai, J., R. Guha, and J. Lee (2009). Efficient virus detection using dynamic instruction sequences. *Journal of Computers 4*(5), 405–414.

Denison, D. G. T., C. C. Holmes, B. K. Mallick, and A. F. M. Smith (2002). *Bayesian Methods for Nonlinear Classification and Regression*. Wiley Series in Probability and Statistics. Wiley.

Domingos, P. and F. Provost (2000). Well-trained PETs: Improving probability estimation trees. Technical report, CDER Working Paper# 00-04-IS Stern School of Business.

Fisher, R. A. (1929). *Statistical methods for research workers*. Edinburgh.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Fitzpatrick, M. and D. Marchev (2013). Efficient Bayesian estimation of the multivariate double chain Markov model. *Statistics and Computing 23*(4), 467–480.

Goldfeld, S. M. and R. E. Quandt (1973). A Markov model for switching regressions. *Journal of Econometrics 1*(1), 3 – 15.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika 82*(4), 711–732.

Han, D. and F. Tsung (2009). The optimal stopping time for detecting changes in discrete time Markov processes. *Sequential Analysis 28*(1), 115–135.

Kass, R. E., B. P. Carlin, A. Gelman, and R. M. Neal (1998). Markov chain Monte Carlo in practice: A roundtable discussion. *The American Statistician 52*(2), 93–100.

Killick, R., P. Fearnhead, and I. A. Eckley (2012). Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association 107*(500), 1590–1598.

McAfee (2017). McAfee Labs Threats Report June 2017. Accessed June 2017. https://www.mcafee.com/us/resources/reports/rp-quarterly-threats-jun-2017.pdf.

Page, E. S. (1954). Continuous inspection schemes. *Biometrika 41*(1/2), 100–115.

Polansky, A. M. (2007). Detecting change-points in Markov chains. *Computational Statistics and Data Analysis 51*(12), 6013–6026.

Royal, P., M. Halpin, D. Dagon, R. Edmonds, and W. Lee (2006). Polyunpack: Automating the hidden-code extraction of unpack-executing malware. In *Computer Security Applications Conference, 2006. ACSAC '06. 22nd Annual*, pp. 289–300.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Storlie, C., B. Anderson, S. Vander Wiel, D. Quist, C. Hash, and N. Brown (2014). Stochastic identification of malware with dynamic traces. *The Annals of Applied Statistics 8*(1), 1–18.

Tartakovsky, A. G., B. L. Rozovskii, R. B. Blažek, and H. Kim (2006). Detection of intrusions in information systems by sequential change-point methods. *Statistical Methodology 3*(3), 252–293.

Van Gael, J., Y. Saatci, Y. W. Teh, and Z. Ghahramani (2008). Beam sampling for the infinite hidden Markov model. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, New York, NY, USA, pp. 1088–1095. ACM.

Vasas, K., P. Elek, and L. Márkus (2007). A two-state regime switching autoregressive model with an application to river flow analysis. *Journal of Statistical Planning and Inference 137*(10), 3113–3126. Special Issue: Bayesian Inference for Stochastic Processes.

Xian, J.-G., D. Han, and J.-Q. Yu (2016). Online change detection of Markov chains with unknown post-change transition probabilities. *Communications in Statistics - Theory and Methods 45*(3), 597–611.

Xing, H., N. Sun, and Y. Chen (2012). Credit rating dynamics in the presence of unknown structural breaks. *Journal of Banking and Finance 36*(1), 78 – 89.

Yakir, B. (1994). Optimal detection of a change in distribution when the observations form a Markov chain with a finite state space. *Lecture Notes-Monograph Series 23*, 346–358.

Zhao, H. and P. Marriott (2014). Variational Bayes for regime-switching log-normal models. *Entropy 16*(7), 3832.

ACCEPTED MANUSCRIPT