Захист інформації та кібернетична безпека

UDC 621.391 DOI: 10.30748/soi.2018.155.09

Qasim Abbood Mahdi

AL Taff University College Iraq, Karbala

TRAFFIC ANALYSIS OF ANONYMITY PROTOCOL USING HIDDEN MARKOV MODEL BASED ON THE MODEL CONFIDENCE

The issue of increasing the confidentiality and stealth of users on the Internet is the most pressing issue of the day. One way to increase the secrecy of using Internet services is to install the Tor software, which protects itself from the "data flow analysis" is a type of network surveillance that threatens the privacy of users, the confidentiality of business contacts and communications implemented through routing network traffic over a distributed network of servers running volunteers from around the world that does not allow the external observer to monitor the user's Internet connection, find out which sites were visited, and also does not allow the site to know the physical location of the user. However, the software in question has vulnerabilities that result in the loss of personal user freedom. The author, through the application of general scientific methods such as analysis and synthesis, identified a list of vulnerabilities and their importance for the confidentiality of the Tor software. The author carried out the simulation of the Tor software by devices of the experimental environment and the construction of experimental procedures based on the used mathematical apparatus of the Markov chains. The results of the experiment indicate the necessity to determine the validity of the model for analysis of the anonymity protocol. In the course of this research, an algorithm for testing the anonymity of Tor software users was developed, which allows to identify possible sources of personal information of users. The effectiveness of the proposed modeling trust algorithm was demonstrated by calculating the value of a training set of data necessary for outputting a wireless access protocol, a proxy through Tor.

Keywords: *Tor anonymity network, Internet security, intrusion detection system, traffic survey.*

Introduction

Tor Browser helps you to protect yourself from "data flow analysis" that threatens personal freedom and privacy, confidentiality of business contacts and connections.

This service provides protection by routing your network traffic across a distributed network of servers launched by volunteers from around the world: this does not allow an external observer to track your Internet connection to find out which sites you visit, and also does not allow the site to know your physical location.

This program works with many existing applications, including web browsers, instant messaging systems, remote access clients, and other applications using the TCP protocol.

Hundreds of thousands of people around the world use Tor for a variety of reasons: journalists and bloggers, human rights organizations, law enforcement officers, military personnel, corporations, people in countries with repressive regimes, and just ordinary citizens.

This article describes a scenario in which a client interacts with a server through Tor. Assuming that the communication protocol used by the client can be represented by a Hidden Markov Model (HMM), we can derive a model that is an exact representation of the underlying protocol using the time information collected on the server side. The suggested trust model approach is applied to the attack experiment to determine the size of the data required to construct a statistically significant representative of the protocol.

Therefore, **the aim of this work** is the traffic analysis of anonymity protocol using Hidden Markov Model (HMM) based on model confidence.

Exposition of the main material of the research

1.1 Z-test

Among several classic statistical tests, z-test is a simple but widely used statistical test. The rationale for this test: given the random sample size n, a sequence of random variables independent and identically distributed (IID) from an unknown distribution, we are going to make a decision for each value, and this decision will be either correct or not.

Consider the distribution of the number of errors that will be made by our classification system. Since each solution is independent of the others and is binary, it is reasonable to assume that the random variable X, representing the number of errors should follow the binomial distribution $B(n,p)$, where p is the error rate. In addition, it is known that the binomial distribution $B(n,p)$ can be approximated by a normal distribution $N\left(\mu,\sigma^2\right)$:

$$\mu = np \text{ and } \sigma^2 = np(1-p) \quad (1)$$

when n is the big enough [1]. Finally, if $X \sim N(np, np(1\text{-}p))$, then the error frequency distribution is the $Y = \frac{X}{n} \sim N\left(p, \frac{p(1\text{-}p)}{n}\right)$, then $\frac{\sqrt{n}(X-\mu_0)}{\sqrt{p(1-p)}} \sim N(0,1)$.

Let's take a sample $x_1, x_2, ..., x_n$. The null hypothesis is the expected value X that is a given value μ_X. Then we can write test statistics like this:

$$z = \sigma_{\bar{X}} = \frac{\bar{X} - \mu_X}{\sigma_{\bar{X}}}, \quad (2)$$

where $\sigma_{\bar{X}} = \frac{\sigma_X}{\sqrt{n}}$ and σ_X is the dispersion X. The conversion process (2) is called standardization or normalization, and the result is called the standard estimate or z-count. z determines how many standard deviations below or above the population means that the average value of the sample is under the null hypothesis. However, in most cases σ_X is unknown and may be replaced by sample variance $x_1, x_2, ..., x_n$, if n is big enough. Using test statistics z for a given level of significance α, we compute one way or two way p-value. We reject the null hypothesis if p-value is less than α and takes it another way.

Or, since the statistics z follows the standard normal distributions, if the null hypothesis is correct, the decision to reject the null hypothesis can also be made by comparing the statistics z with a critical value without converting it to p-value.

1.2 Hidden Markov Model

The standard Hidden Markov Model (HMM) is N-Markov chain observed at discrete points in $t = 0,1,2...$. Let us assume that $S = \{1,2,...,N\}$ represents the space of the final state if we use a random variable S_t to indicate the state of the HMM at the time t, $S_t = s$ means that the HMM is in the condition of $s \in S$ in time step t. However, S_t can't be observed directly. Instead, we see one way out. $O_t = o \in O$, where $O = \{1,2,...,M\}$ stands for the final set of outputs, also known as observations. For each state $s \in S$. Two probability distributions are defined to represent the state transition and output radiation rules, respectively:

• Transition state theory

$$P_s^{s'} = \Pr\{S_{t+1} = s' | S_t = s\}, \quad \forall s, s' \in S, t = 0,1,2,... \quad (3)$$

• probability of observations

$$P_s^O = \Pr\{O_{t+1} = o | S_t = s\}, \quad \forall s \in S, o \in O, t = 0,1,2,... \quad (4)$$

As shown in fig. 1, two HMM probability distributions generate two parallel stochastic processes [1]: a process of states and state of observations.

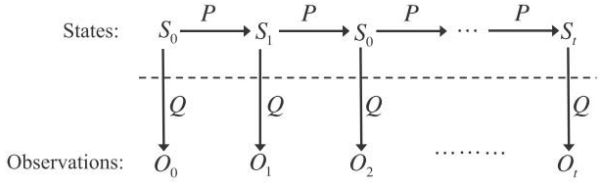

Fig. 1. The HMM process and its two stochastic processes: a probability of $\{S_t\}$ and the observation process $\{O_t\}$

In this paper, we consider the problems of HMM inference and a specific inference algorithm is the causal splitting restoration algorithm (CSRA). This approach of the HMM [2] creates state machines deterministic at the transition exit, i.e. when each observation is displayed in no more than one transition, leaving the state. In addition, the main Markov chain HMM generated using the Shalizi method when all transition states are removed [2].

1.3 Partially Observable Markov Decision Process

In the language of stochastic control, Partially Observable Markov Decision Process (POMDP) are control problems with partial observation. They usually simulate stochastic environments with hidden processes. By summarizing the Markov decision process (MDP) and providing greater uncertainty, the POMDP provides a more powerful formalism for modeling realistic problems, especially for managing systems with noisy data or limited sensitivity.

Formally, POMDP is defined as a 6-tuple $\langle S, A, O, T, Z, r \rangle$, where

$S = \{1,2,...,N\}$ is the finite set of states.

$A = \{1,2,...,M\}$ is the finite course of action by.

$O = \{1, 2, ..., K\}$ is the finite set of observations and output lends.

$T : S \times A \times S \to [0,1]$ is the state-transition function. $T(s, a, s') = \Pr(s' | s, a)$ represents the probability of transition to state s' after taking action a in the state s.

$Z : A \times S \times O \to [0,1]$ is the probability function of observation. $Z(a, s', o) = \Pr(o | a, s')$ denotes the probability of seeing o in the state s' after taking action a at the preceding step.

$r : S \times A7 \to R$ is the direct renumeration function.

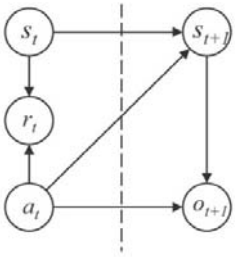

Fig. 2. An influence diagram showing the relationships between the various elements containing POMDP, which meanwhile prove the Markov property

Solid arrows represent the dependencies of existence (for example, S_{t+1} depend both on S_t, and on A_t).

At any time, the system in some state $s \in S$. The agent accepts action $a \in A$, which immediately gives a reward $r(s, a)$ and starts the transition to the new state $s' \in S$ in the next step with probability $T(s, a, s')$. Three elements S, A and T form the core-MDP and determine the dynamics of the POMDP. But, unlike conventional MDP, the agent can't observe the state of the MDP core during the decision process. Instead, he gets an observation $o' \in O$ with probability function $Z(a, s', o')$.

Unlike standalone HMMs, POMDPs are controlled by actions selected by agents or controllers. The goal of the POMDP study is to find a sequence of actions $\{At\}\, t = 1, 2, ...$ known as the policy that makes the system work as agents want. A policy is measured by a compensation function, which is a mathematical function of immediate remuneration. The goal of the agent is to optimize the compensation function.

1.4 Decentralized POMDP

When decision making becomes a collective work in which several agents need to be coordinated without effective communication and even unclear about their own local situation, the decentralized Controlled Markov Processes with partial observations (DC-POMDP) is the main tool used in decision theory to solve this problem [3]. As an extension of POMDP to the case of several agents, DC-POMDP is a more general and more powerful modeling tool. However, the DC-POMDP solution usually leads to excessive computational overhead.

DC-POMDP can be formally defined by a topple $\{S, K, A_1, ..., A_K, O_1, ..., O_K, R, P, Q\}$, where

S is the finite set of states.

K is the number of agents.

$A^{(k)}, 1 \le k \le K$ is the agent's action space k.

$O(k), 1 \le k \le K$ is the agent's observation space k.

$R : S \times A^{(1)} \times ... \times A^{(K)} \to R$ is the direct renumeration function.

$R : S \times A^{(1)} \times ... \times A^{(K)} \times S \to [0,1]$ is the state stochastic transition function. Let us assume that $\bar{a} = [a_1, a_2, ..., a_K]$, where $a_k \in A^{(K)}$, $P(s, a, s')$ represents the probability of transition from the state s to the state s'.

$Q : A^{(1)} \times ... \times A^{(K)} \times S \times O^{(1)} \times ... \times O^{(K)} \to [0,1]$ is the state stochastic transition function. Let us assume that $\bar{o} = [o_1, o_2, ..., o_K]$, where $o_k \in O^{(K)}$, $Q(\bar{a}, s', \bar{o})$ represents the probability of obtaining a sequence of observations $\bar{o}$ in the state s' after the agent k, $1 \le k \le K$ takes action a_k at the preceding step.

From the definition, we can see that each agent needs a local policy. But joint actions affect both the dynamics and the global reward.

1.5 Final State Controller (FSC)

Although any POMDP policy may be represented by a policy schedule, for some policies of an infinite horizon, infinite policy schedules may be required [4]. Therefore, most policy-based algorithms limit their search to finite political graphs, i.e. FSC, which can be defined as an extension of a probabilistic automaton $\langle N, A, y, b_0, E \rangle$ [5] along with the probability distribution $x : N \times A \to [0,1]$ and the output set O, where

N is the finite set of internal states of the controller;

A is the course of action by POMDP;

O is the set of observations POMDP.

$y : N \times O \times N \to [0,1]$ is the state-transition function, $y(n, o', n') = \Pr(n' | n, n, o')$ is the probability of transition to n' from the state n after observation o';

b_o is the default beliefs;

$E \subseteq N$ is the discrete set;

$x : N \times A \to [0,1]$ is the action choice function;

$X(n,a) = \Pr(a|n)$ is the probability of taking action $a \in A$ in the state $n \in N$.

Since the concept of FSC is not accepted, E usually is not indicated. Moreover, ε-optimal FSC for average POMDP does not depend on the initial opinion b_0. Therefore, the definition of FSC can be reduced to $\langle N, A, O, x, y \rangle$, where A and O are known with this POMDP. This leaves only two unknown variables: x and y. At each step t FSC takes $a \in A$ as entering the state $n \in N$ and generates an observation o' in the next step in response. The transition of the internal state is probabilistic and is determined by recent history, as can be seen from the definition y. The number of internal states $|N|$ represents the size of the FSC, as well as the amount of agent memory [4]. Internal states are fully checked by the agent during the decision-making process, which selects the action to be taken on each node $n \in N$ in relation to function *x*.

1.6 Sequential Quadratic Programming (SQP)

SQP is one of the most successful methods for solving problems of nonlinear limited optimization. It consists of a set of algorithms, not just one algorithm and is based on a deep theoretical foundation. SQP has demonstrated excellent performance in solving general problems of large-scale non-linear programming. In this section, we look at the following NLP (natural language processing) problem:

$$\begin{aligned} &\min\ f(x) \\ &\text{s.t.}\ \ g(x) \le 0; \\ &h(x) = 0; \\ &x \in R^n, \end{aligned} \tag{5}$$

where x is a factor from n component; $F : R \to R^n$ – objective functional; the functions $h(x) : R^n \to R^m$ and $g(x) : R^n \to R^1$ are resp. equality and inequality constraints.

SQP solves NLP to convert it into a series of problems with quadratic programming (QP). At each iteration, the original NLP is reformulated as a subtask QP, linearizing the constraints and replacing the objective function $f(x)$ with its local quadratic approximation. QP subtask is:

$$\begin{aligned} &\min\ \frac{1}{2} d^T B_k d + \nabla f(x_k)^T \\ &\text{s.t.}\ \ \nabla g(x_k)^T d + g(x_k) \le 0 \\ &\nabla h(x_k)^T d + h(x_k) = 0 \\ &d := x - x_k \in R^n \end{aligned} \tag{6}$$

$\nabla f(x_k) := \left(\frac{\delta f(x_k)}{\delta x_k^1}, \frac{\delta f(x_k)}{\delta x_k^2}, ..., \frac{\delta f(x_k)}{\delta x_k^n} \right)$ stands for the gradient function $f(x_k)$ at the point $x_k = \left[x_k^1, x_k^2, ..., x_k^n \right]$, and $B_k := Hf(x_k)$ is Hessian matrix $f(x)$ at the point x_k that is, the matrix of second partial derivatives of a function $f(x)$. Assuming that x_k is a solution to the QP routine for k iteration, which is actually an evaluation of the original NLP solution. So the sequence $\{x^k\} k \in N_0$ converges to local optimal x^* NLP. The basic idea of SQP is similar to the methods of Newton and quasi-Newton. However, the presence of constraints makes the analysis and implementation of SQP methods much more difficult [6]. There are many NLP for which individual SQP methods exist to solve them. This NLP include unconditional optimization systems, linearly limited optimizations and non-linearly limited optimizations. We speak POMDP as NLP and rely on SQP tools to find solutions.

2 Anonymity Protocol Analysis

To use the z-test, let us offer a simple algorithm for operational testing of the sequence of observations. The algorithm determines whether the built model will statistically represent the data flow in the collection process. First, we collect a sequence of observational data y of some length D and build a model from the collected data. With a built model, we define z-statistics and find if experimental statistics provides $100 \cdot (1 - \alpha)\%$ confidence that the transition with probability ε does not occur. If *y* is not long enough, we will not be able to build a model from the data; it is necessary to collect additional data. The algorithm is presented below.

2.1 Algorithm

Let us denote the transitional probability δ, when the system is in the state s:

$$\gamma_s^\delta = \frac{\varepsilon}{\pi_s}, \tag{7}$$

where π_s is the asymptotic probability of the state s is defined by the formula $\pi_s \approx \frac{n_s}{D}$, where n_s is the number of times the state s is introduced during the observation time y^D.

Standardized z-statistics for the state s is defined by the formula

$$z_s = \frac{\gamma_s^{\delta}}{\sqrt{\frac{\gamma_s^{\delta}\left(1-\gamma_s^{\delta}\right)}{n_s}}}. \quad (8)$$

Test z-statistics for the state s is defined by the formula

$$z_{exp} = \min_s z_s. \quad (9)$$

The model certainty is defined as

$$a_f = 1 - \prod_{s \in S}\left(1 - P\left(Z < z_s\right)\right), \quad (10)$$

where $P\left(Z < z_s\right)$ – the probability that a normal distribution matters less z_s.

The minimum amount of training data:

$$D^a = \frac{n_s}{\pi_s}. \quad (11)$$

Algorithm:

Input:

- repeated observation y_t of the time t;
- alphabet O;
- a threshold set by the user ε and the significance level a.

Output:

- the significance of the model a_f;
- required length $|y|$, D.

From time factor $t = 0$:

1. To build the G_t model out of sequence $y = y_0, y_1, ..., y_t$.
2. To calculate the probabilities of the asymptotic state π_s for $\forall s \in S$.
3. Under step 7 determine the values γ_s^{δ} for $\forall s \in S$.
4. To calculate experimental statistics z_s for each step 8.
5. To find z_{exp} according to equation (9).
6. If $z_{exp} > z_a$, to conclude that G_t is the underlying process with the desired level of confidence and $D = |y|$;
7. To calculate a_f according to equation (10); stop.
8. Otherwise, collect more data $|y| = D^a$, where D^a is calculated by equation (11);
9. Go to step 1.

Summing up, we make the following assumptions about the observation data and our knowledge of the underlying process. First, the process under consideration has a finite number of states and the transition probabilities are stationary. This assumption ensures that the training data set fully reflects the process. In addition, the alphabet O was completed and contains all expected observations.

Assuming this, our approach is limited to finding "known-unknowns" [7] at a given degree of statistical likelihood a. If the observation is not in the alphabet, i.e. is an "unknown-unknown" [7], the transition does not affect the confidence or the probability of an unknown transition.

Also, if $K_s = O$ and $U_s = Ø$, then the state has no possible fail-safe outgoing transitions. Transitions are not available to exit the state, and state testing does not change the confidence in the model.

2.2 POC

The experiments presented in this section are simple. The goal is to test the HMM construction algorithm [8] and provide readers with simple illustrations of the concept of model confidence. This section provides two examples. Below we consider the details of the application of the suggested algorithm for determining the model confidence in the detection of the Tor network protocol.

Example 1

The HMM used to generate the observation sequence of Example 1 is shown in fig. 3, a. Let's start with a random selection of the initial state in this model. At each step, an outgoing transition occurs with an appropriate probability and the corresponding symbol is observed. The initial process was set up to generate 10,000 data symbols, fig. 3, b. We can see that the reconstructed model has the same state structure and almost the same transition probabilities as the original model.

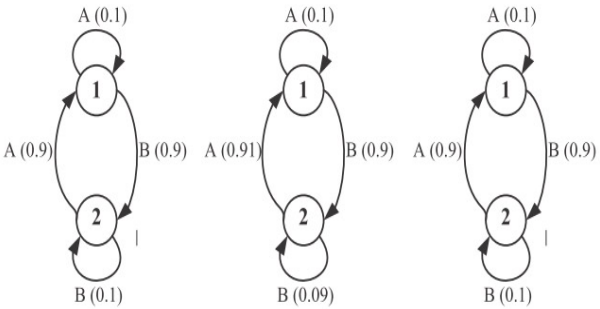

Fig. 3. Example 1:
a – original model;
b – Model built of 10 000 packages;
c – Model built of 100 000 packages

If the process is repeated for 100 000 characters, we find that the probabilities correspond to the original model.

Example 2

The same steps were applied to the Markov chain in fig. 4, a, as was done in the example.

1. A model built of the first 10 000 observations is shown in fig. 4, b. In this case, the state structure model is different from the original model, since state 4 has not been visited once since the first 10 000 observations. However, with a large amount of data collected, the state structure of the reconstructed model using 100 000 observations matches the original model. In addition, the transition probabilities are also closer to the actual values.

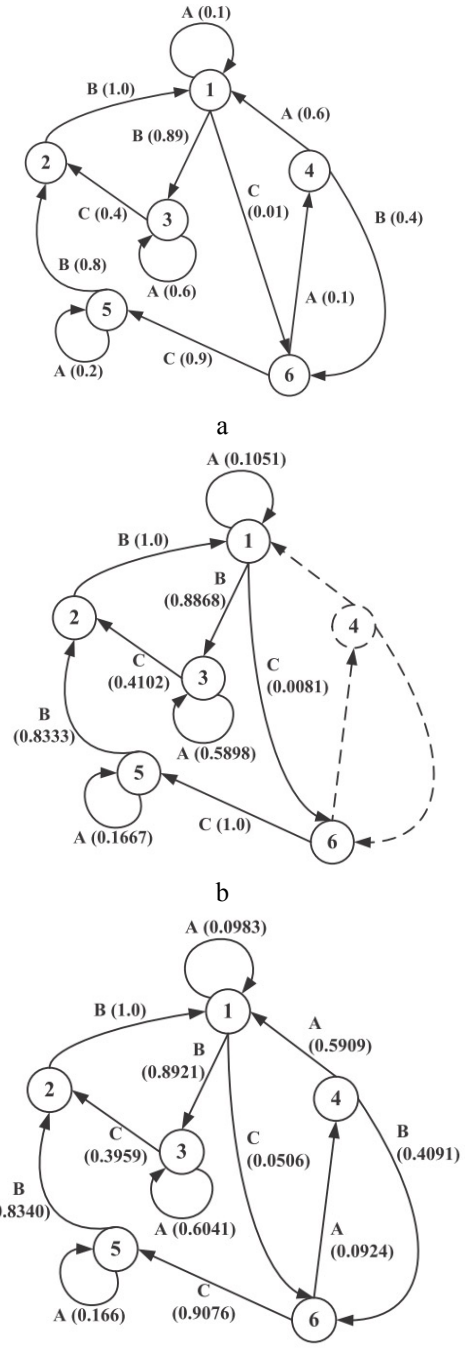

Fig. 4. Example 2:
a – Original model;
b – Model built of 10 000 characters;
c – Model built of 100 000 characters

The results of both examples illustrate a point made earlier. If an excessive number of data samples is used, the algorithm creates only a model that represents only the data used to create it and not the main process. This is what makes the difference in the state structure and transition probabilities. The probability of not seeing the transition decreases with increasing observation time. To create a model that represents the underlying process, there must be enough data in the training kit to fully describe this process.

Thus, we have shown how to determine at a given degree of statistical likelihood, if an "unknown" transition does not occur taking into account two user-defined threshold values ε. The parameter ε determines the minimum probability that transitions with probabilities of at least ε should be included in the built model.

Thus, we have shown how to determine at a given degree of statistical likelihood, if an "unknown" transition does not occur taking into account two user-defined threshold values ε. The parameter ε determines the minimum probability that transitions with probabilities of at least ε should be included in the built model. A parameter a is the confidence level that shows the accuracy of the model result. In our demonstrations of the algorithm, we specifically looked at whether the built model corresponds to a model that acts as the main process.

2.3 Protocol Detection

Now we use the model trust approach presented above to determine the protocol that the sender uses when talking to a client over the Tor network, collecting time intervals between packets on the client. The time between sending each packet depends on the symbol associated with the transition. Each character is assigned a specified time delay in milliseconds, and the server waits for this amount of time before sending the packet to the client. This method links inter-packet delays with HMM transitions. In other words, the time delays between successive packets will be our observations of the main process. This is the behavior that we expect in actual protocols that the packet time will be associated with the processing required by a particular task in this process.

Tor is a low-latency overlay network that allows applications to communicate anonymously and securely on the Internet. An overlay network is a logical network connected by virtual circuits on top of a physical network. Links that connect individual systems in the overlay network are implemented as "tunnels" through the core network. Sent packets are encrypted multiple times so that they remain logically separate from normal traffic. The stability and deployment of Tor can be explained by its practical design [9–10].

Tor basically consists of computers serving two types of services: a repeater and a directory server. There are several thousand relays, also known as onion routers, which operate on a voluntary basis by individuals and organizations around the world. The path through Tor is built of a relay. Relays and clients exchange data according to the catalog [11] for the exchange of catalog information. By default, relays listen on TCP port 9001 for incoming requests. Active relays publish their router handles to the list of predefined directory servers (organs), reporting their current status. Directory servers store router handles on the relay list and constantly check the availability of these relays. In addition, each flag is assigned a different flag in accordance with their knowledge of the network status, that is, which ones should be displayed as working, valid, stable, etc. Directory servers exchange their views with each other on the network on a regular basis, for example, every hour. After all servers match the list of available relays, which is called consensus over the network, the consensus is published on the TCP port (default 9030) and available for download.

To use Tor, the client will need an HTTP proxy to retrieve the Tor directory and an HTTPS proxy to receive the relay. The current version of Tor allows the client to use any HTTPS or SOCKS proxy server to access the Tor network. Once installed, Tor can be initiated as an onion proxy (OP) if it processes only local requests. SOCKS proxy listens on port 9050 by default for streams created by TCP-based applications, such as web browsing, SSH, instant messaging, etc. Then the traffic will be routed via Tor.

Tor starts building charts as soon as they have enough directory information. When the application flow arrives, it will be connected to a pre-built circuit, if it exists, or wait until the circuit is available. Before building the circuit, the client selects all relays (by default, by default) to use the launch with the output node. The entry node of the circuit must be one of the entry guards, which is a set of nodes used by the client as long-term entry points to Tor.

The connection between the client and the entry node is first established using TLS/SSLv3 for authentication and encryption [11]. After creating the first connection, the path extends to the second and third nodes in a similar way. Using this incremental path-building project, the client sets the session keys with each subsequent node independently [12]. The final node of the scheme, known as the output node, is selected to ensure, at best, support for connections to the destination.

Before joining a stream to a built scheme that can support a client request, Tor will send a test request. If the request is not completed, Tor will send an error to the user.

All traffic going down the scheme is packed into 512-byte cells, which is an effective measure against leakage of packet size information passing through the side channels. Then these cells are iteratively encrypted using the key of each serial relay circuit. That is, the outermost layer of the packet is encrypted using the public key of the input node.

And so on, the innermost level of encryption is performed through the key of the output node. When a cell moves down the chain and comes to each relay node, the node "expands" the cell with its private key to identify where it should send the decrypted cell, for example, clear the onion skin. Thus, each node in the chain knows only the ascending node and the node downstream and cannot evaluate the entire panorama of the circuit. Thus, the compromise of a single node does not violate anonymity.

The procedure described is illustrated in fig. 5. When the addressee, Alice, responds to Bob's request, the same process is performed in the reverse order. There are many other details of the process, such as encryption schemes, integrity checking, congestion handling, path selection, etc. A detailed specification of the Tor protocol can be found in [12].

Here is a practical example of detecting a protocol tunneled through Tor to illustrate the usefulness of the application of the suggested model trust algorithm. We use the approach [13] introduced to derive a protocol model that the server uses when talking to a client through the Tor network, by collecting time intervals between packets on the client.

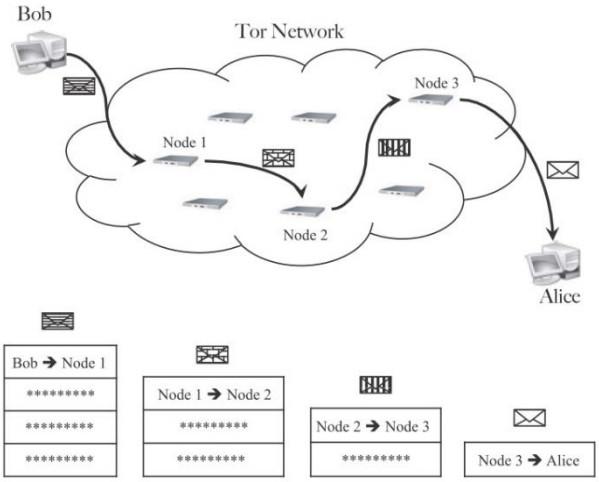

Fig. 5. The Tor Cascade, which originated from Bob, intended for Alice, is sent through a Tor circuit consisting of 3 relays

First, we have a valid HMM that represents the protocol used. The time between sending each packet depends on local processing and is represented by the symbol associated with the transition. Each character is assigned a time delay range in milliseconds, and the server waits for this amount of time before sending the packet to the client. This method links inter-packet

delays with HMM transitions. In other words, the time delays between successive packets will be our observations of the main process. In actual protocols, the packet time will be related to the processing required by a particular task in the process. After designating the data that we record, the model building algorithm is used to create the model used by the server.

2.4 Model Building

The model used by the server in this experiment is shown in fig. 6.

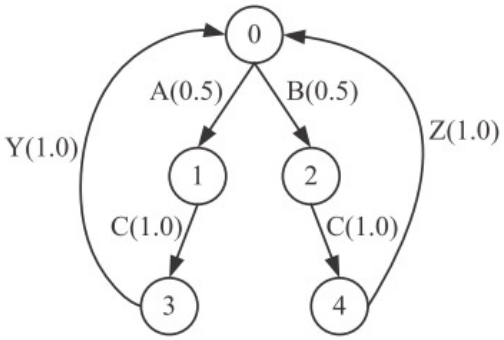

Fig. 6. Original five-state model for the pruning experiment

The server starts the process by randomly selecting a state in its model as the start state. To send each packet, the transition is taken from the current state, and the corresponding time delay waits before sending the packet to the client. If there is more than one possible transition from a state, the transition is selected randomly, weighted by the probability of each transition. All data collection was performed on processes sent via Tor. In the article [13] program was used to capture packets within the network. Calculate the difference between each successive packet time Δt. We then symbolize the data, grouping them into ranges and assigning something in that range to a unique character, such as A or B. We start with $L = 2$ and increase it as needed. We follow the process described in the flowchart in fig. 7, to create the models required by our attack.

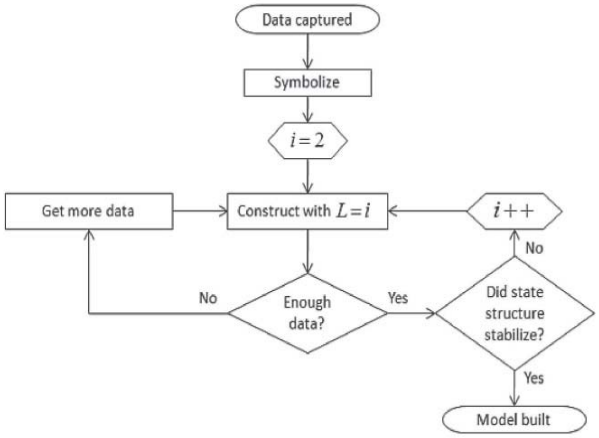

Fig. 7. Flowchart summarizing the process of building a model

When the confidence test is run on the model, we find that it requires 20 624 750 data samples. This means we need to capture more data and rebuild. Because the amount of required data is so large, it has to be generated in lengths of 200 000 packets at a time. After each set of 200 000 we rebuild the model and run the confidence test again.

Oddly enough, the required amount of data keeps increasing with each set. In a Tor connection, there are times when a circuit fails or changes, or a relay gets too busy and delays a packet. There is some extra variable latency that affects roughly one out of every 200 packets. These glitches cause the packet to arrive later than it should have and because of that, it is incorrectly symbolized. All of these new events are very low probability, which results in a lower minimum asymptotic state probability for each new set. This lower probability causes the confidence test to increase the amount of data required.

To prune these unsubstantiated states and transitions from the model we use the method of thresholding the asymptotic state probabilities.

After a model has been built with CSRA, it may have transitions that are taken very rarely and states that are visited very rarely. By setting a threshold on the asymptotic state probabilities, rare events are trimmed from the model. The pruning process is carried out mainly in three steps:

1. Any state with an asymptotic probability below the threshold is removed from the model.

2. Any transitions going to or leaving from that state are also removed.

3. Finally, any state or set of states that cannot be reached due to a removal are also deleted.

This leaves the model with only the states and transitions for which we have enough data. When we are unable to collect enough data to be confident in the full model, we leave out the parts where we would need more data to achieve confidence.

The value of the probability threshold is how often we should expect the process to deviate from the model. The smallest asymptotic state probability and corresponding result from the confidence test are plotted against the number of packets captured in fig. 8.

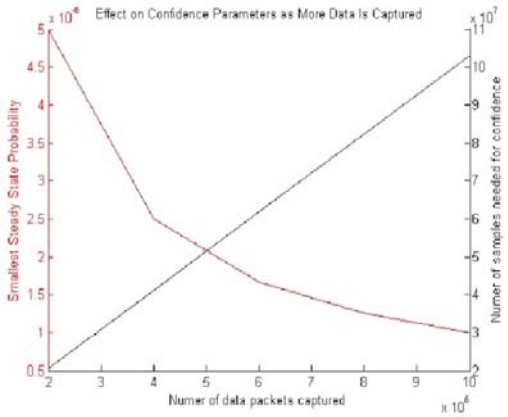

Fig. 8. The plot of model confidence results as more data is captured

The steady increase suggests we will not easily capture enough data to rebuild the model confidently. As for our experiment, analysis of the asymptotic state probabilities shows a large gap between 71 of the states and the other eight. The 71 have probabilities below 0.06%, while the other eight have probabilities above 8.2%. That is a break of over two orders of magnitude. This division makes a good level of significance for pruning. Following the pruning process, the model in fig. 9 results with a threshold of 0.01 (or 1%).

It is appropriate at this point to recall that the states of the putative HMM are characterized as having the same probability distribution over the next output symbol. In this case, it follows that nodes 3 and 4 in fig. 9 can be considered as one and the same state and should be combined with each other. Similarly, nodes 5 and 7 are combined, as well as states 2 and 6.

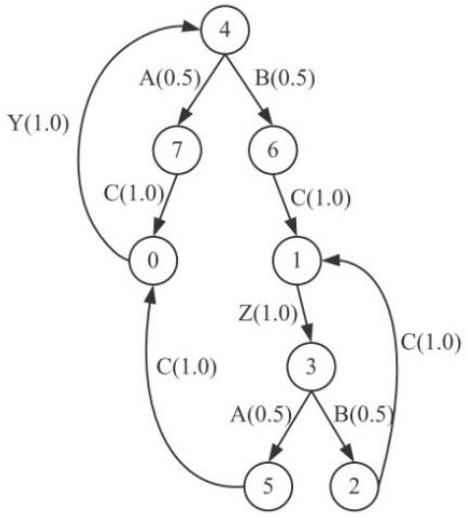

Fig. 9. The result after pruning low-probability states and transitions

Nodes 6 and 7, although both have the same output, must remain two separate states. Otherwise, the transition leading to the combined state will be displayed on more than one symbol, that is, on A and B. In fig. 10 shows the resulting model, which essentially coincides with the original model in fig. 6.

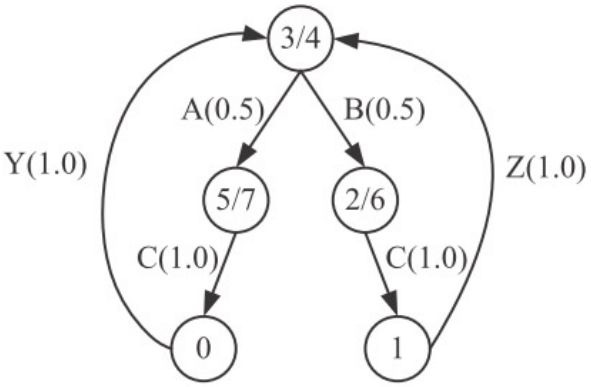

Fig. 10. The resulting model after the merging of states with the same probability distribution of the next output

Conclusion

This article analyzes the traffic of the anonymity protocol using a hidden model of the model based on the Markov model, reveals its main features.

Thus, the work describes the temporal side of the synchronization channel attack to detect a communication protocol tunneled through Tor. Model trust algorithm is applied to the implementation of the attack. A proof-of-concept experiment on our private Tor network showed that a model could successfully be reconstructed from inter-packet timings, and also proved the practical application of the model trust algorithm.

The direction of further research should be considered the development of methods for increasing the confidentiality of traffic in public networks.

Список літератури

1. The path less travelled: Overcoming Tor's bottlenecks with traffic splitting / Mashael AlSabah, Kevin Bauer, Tariq Elahi, Ian Goldberg // Privacy Enhancing Technologies Symposium (PETS). – Springer, 2013. – P. 143-163.

2. Chaabane A. Digging into anonymous traffic: A deep analysis of the Tor anonymizing network / A. Chaabane, P. Manils, M.A. Kaafar // IEEE Network and System Security (NSS). – 2010.

3. A Survey of Electric Power Synchrophasor Network Cyber Security / C. Beasley, X. Zhong, J. Deng, R. Brooks, G. Kumar Venayagamoorthy // Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), 2014 IEEE PES. – P. 1-5. https://doi.org/10.1109/ISGTEurope.2014.7028738.

4. Aras R. An Investigation into Mathematical Programming for Finite Horizon Decentralized POMDPs / R. Aras, A. Dutech // Journal of Artificial Intelligence Research. – 2010. – Vol. 37. – P. 329-396. https://dx.doi.org/10.1613/jair.2915.

5. Zero knowledge hidden Markov model inference / J.M. Schwier, R.R. Brooks, C. Griffin, S. Bukkapatnam // Pattern Recognition Letters. – 2009. – Vol. 30(14). – pp. 1273-1280. https://dx.doi.org/10.1016/j.patrec.2009.06.008.

6. Archer G.E.B. Parameter estimation for hidden Markov chains / G.E.B. Archer, D.M. Titterington // Journal of Statistical Planning and Inference. – 2002. – Vol. 108 (1-2). – P. 365-390.

7. Ephraim Y. Hidden Markov processes / Y. Ephraim, N. Merhav // Special issue on Shannon theory: perspective, trends, and applications. Institute of Electrical and Electronics Engineers. Transactions on Information Theory. – 2002. – Vol. 48(6). – P. 518-1569.

8. Poupart P. Exploiting structure to efficiently solve large scale partially observable Markov decision processes: PhD thesis / P. Poupart. – University of Toronto, 2005.

9. Spaan M.T.J. Perseus: randomized point-based value iteration for POMDPs / M.T.J. Spaan, N. Vlassis // JAIR. – 2005. – Vol. 24. – P. 195-220.

10. Side Channel Analysis of Multiple PMU Data in Electric Power Systems / X. Zhong, P. Arunagirinathan, A. Ahmadi, R. Brooks, G.K. Venayagamoorthy, L. Yu, Y. Fu // Power System Conference (PSC). – 2015. – Clemson University. – P. 1-6.

11. Fu Y. Using botnet technologies to counteract netowrk traffic analysis: Ph.D. thesis / Y. Fu. – Clemson University, 2017.

12. Rabiner L.R. An introduction to hidden markov models / L.R. Rabiner, B.H. Juang // IEEE ASSP Magazine. – 1986. – P. 4-16.

13. Capp´e O. Inference in Hidden Markov Models / O.Capp´e, E. Moulines, T. Ryden. – Springer Series in Statistics, Springer-Verlag New York, Inc., Secaucus, 2005. – NJ, USA.

14. Loesing K. Case Study on Measuring Statistical Data in the Tor Anonymity Network / K. Loesing, S.J. Murdoch, R.A. Dingledine // Proc. of the Workshop on Ethics in Computer Security Research. – 2010.

References

1. AlSabah, M., Bauer, K., Elahi, T. and Goldberg, I. (2013), The path less travelled: Overcoming Tor's bottlenecks with traffic splitting, *Privacy Enhancing Technologies Symposium (PETS)*, pp. 143-163, Springer.

2. Chaabane, A., Manils, P. and Kaafar, M.A. (2010), Digging into anonymous traffic: A deep analysis of the Tor anonymizing network, *IEEE Network and System Security (NSS)*.

3. Beasley, C., Zhong, X., Deng, J., Brooks, R. and Kumar Venayagamoorthy, G. (2014), A Survey of Electric Power Synchrophasor Network Cyber Security, *Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), 2014 IEEE PES*, pp. 1-5. https://doi.org/10.1109/ISGTEurope.2014.7028738.

4. Aras, R. and Dutech, A. (2010), An Investigation into Mathematical Programming for Finite Horizon Decentralized POMDPs, *Journal of Artificial Intelligence Research*, Vol. 37, pp. 329-396. https://dx.doi.org/10.1613/jair.2915.

5. Schwier, J.M., Brooks, R.R., Griffin, C. and Bukkapatnam, S. (2009), Zero knowledge hidden Markov model inference, *Pattern Recognition Letters*, Vol. 30(14), pp. 1273-1280. https://dx.doi.org/10.1016/j.patrec.2009.06.008.

6. Archer, G.E.B. and Titterington, D.M. (2002), Parameter estimation for hidden Markov chains, *Journal of Statistical Planning and Inference*, Vol. 108 (1-2), pp. 365-390.

7. Ephraim, Y. and Merhav, N. (2002), Hidden Markov processes, *Institute of Electrical and Electronics Engineers. Transactions on Information Theory, Special issue on Shannon theory: perspective, trends, and applications*, Vol. 48(6), pp. 1518-1569.

8. Poupart, P. (2005), *Exploiting structure to efficiently solve large scale partially observable Markov decision processes: PhD thesis*, University of Toronto.

9. Spaan, M.T.J. and Vlassis, N. (2005), Perseus: randomized point-based value iteration for POMDPs, *JAIR*, Vol. 24, pp. 195-220.

10. Zhong, X., Arunagirinathan, P., Ahmadi, A., Brooks, R., Venayagamoorthy, G.K., Yu, L. and Fu, Y. (2015), Side Channel Analysis of Multiple PMU Data in Electric Power Systems, *Power System Conference (PSC)*, Clemson University, pp. 1-6.

11. Fu, Y. (2017), *Using botnet technologies to counteract netowrk traffic analysis: Ph.D. thesis*, Clemson University.

12. Rabiner, L.R. and Juang, B.H. (1986), An introduction to hidden markov models, *IEEE ASSP Magazine*, pp. 4-16.

13. Capp´e, O., Moulines, E., and Ryden, T. (2005), *Inference in Hidden Markov Models*, Springer Series in Statistics, Springer-Verlag New York, Inc., Secaucus, NJ, USA.

14. Loesing, K., Murdoch, S.J., and Dingledine, R.A. (2010), Case Study on Measuring Statistical Data in the Tor Anonymity Network, *Proc. of the Workshop on Ethics in Computer Security Research*.

Received by Editorial Board 10.10.2018
Signed for Printing 11.12.2018

Відомості про автора:

Касім Аббуд Махді
кандидат технічних наук
доцент коледжу університету Ал Тафф,
Кербела, Ірак

Information about the author:

Qasim Abbood Mahdi
Candidate of Technical Sciences
Senior Lecturer of AL Taff University College,
Karbala, Iraq

АНАЛІЗ ТРАФІКУ АНОНІМНОСТІ ПРОТОКОЛУ НА ОСНОВІ ВИКОРИСТАННЯ СКРИТОЇ МАРКІВСЬКОЇ МОДЕЛІ ДОВІРИ

Касім Аббуд Махді

Питання підвищення конфіденційності та скритності роботи користувачів в мережі Інтернет є найбільш актуальним питанням сьогодення. Одним з способів підвищення скритності користування послугами мережі Інтернет є встановлення програмного забезпечення Tor, що дозволяє захиститися від "аналізу потоку даних" – різновиду мережевого нагляду, який загрожує персональній свободі і приватності користувачів, конфіденційності бізнес контактів і зв'язків, що реалізується за рахунок маршрутизації мережевого трафіку по розподіленої мережі серверів, запущених добровольцями з усього світу, що не дає можливості зовнішньому спостерігачеві відстежувати інтернет-з'єднання користувача, дізнатися які сайти були відвідані, а також не дає можливості сайту дізнатися фізичне місце знаходження користувача. Проте зазначене програмне забезпечення має вразливості, що призводять до втрати персональної свободи користувачів. Автором, шляхом застосування загальнонаукових методів, таких як аналіз та синтез, визначено перелік вразливостей та їх важливість для конфіденційності роботи програмного забезпечення Tor. Автором проведено моделювання роботи програмного забезпечення Tor за допомогою експериментального середовища та побудови експериментальних процедур, що засновані на використані математичного апарату Марківських ланцюгів. Результати експерименту свідчать про необхідність визначення вірності моделі для аналізу протоколу анонімності. Також в ході зазначеного дослідження розроблено алгоритм перевірки анонімності роботи користувачів програмного забезпечення Tor, що дозволяє визначити можливі місця витоку персональних відомостей користувачів. Ефективність запропонованого модельного алгоритму довіри демонструється шляхом обчислення величини набору даних навчання, необхідних для виведення протоколу бездротового доступу, проксі через Tor.

Ключові слова: *мережа анонімності Tor, безпека Інтернету, система виявлення вторгнень, дослідження трафіку.*

АНАЛИЗ ТРАФИКА АНОНИМНОСТИ ПРОТОКОЛА НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ СКРЫТОЙ МАРКОВСКОЙ МОДЕЛИ ДОВЕРИЯ

Касим Аббуд Махди

Вопросы повышения конфиденциальности и скрытности работы пользователей в сети Интернет являются наиболее актуальными вопросами современности. Одним из способов повышения скрытности пользования услугами сети Интернет является установление программного обеспечения Tor, что позволяет защититься от "анализа потока данных" – разновидности сетевого надзора, который угрожает персональной свободе и приватности пользователей, конфиденциальности бизнес контактов и связей, реализуется за счет маршрутизации сетевого трафика по распределенной сети серверов, запущенных добровольцами со всего мира, что не дает возможности внешнему наблюдателю отслеживать интернет-соединения пользователя, узнать какие сайты были посещены, а также не дает возможности сайта узнать физическое местонахождение пользователя. Однако указанное программное обеспечение имеет уязвимости, приводящие к потере персональной свободы пользователей. Автором путем применения общенаучных методов ,таких как анализ и синтез, определен перечень уязвимостей и их важность для конфиденциальности работы программного обеспечения Tor. Автором проведено моделирование работы программного обеспечения Tor с помощью экспериментального среды и построения экспериментальных процедур, основанных на использованные математического аппарата марковской цепи. Результаты эксперимента свидетельствуют о необходимости определения верности модели для анализа протокола анонимности. Также в ходе указанного исследования разработан алгоритм проверки анонимности работы пользователей программного обеспечения Tor, что позволяет определить возможные места утечки персональных сведений пользователей. Эффективность предложенного модельного алгоритма доверия демонстрируется путем вычисления величины набора данных обучения, необходимых для вывода протокола беспроводного доступа, прокси через Tor.

Ключевые слова: *сеть анонимности Tor, безопасность Интернета, система обнаружения вторжений, исследования трафика.*