Hidden Markov Model modeling of SSH brute-force attacks

Anna Sperotto, Ramin Sadre, Pieter-Tjerk de Boer, Aiko Pras

University of Twente
Centre for Telematics and Information Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
P.O. Box 217, 7500 AE Enschede, The Netherlands
{a.sperotto, r.sadre, p.t.deboer, a.pras}@utwente.nl

Abstract. Nowadays, network load is constantly increasing and high-speed infrastructures (1-10Gbps) are becoming increasingly common. In this context, flow-based intrusion detection has recently become a promising security mechanism. However, since flows do not provide any information on the content of a communication, it also became more difficult to establish a ground truth for flow-based techniques benchmarking. A possible approach to overcome this problem is the usage of synthetic traffic traces where the generation of malicious traffic is driven by models. In this paper, we propose a flow time series model of SSH brute-force attacks based on Hidden Markov Models. Our results show that the model successfully emulates an attacker behavior, generating meaningful flow time series.

1 Introduction

Since the last decade, we are facing a constant rise of both network load and speed. This has become a problem for packet-based network intrusion detection systems since a deep inspection of the packet payloads is often not feasible in high-speed networks. In this context, flow-based intrusion detection emerged as an alternative to packet-based solutions [1]. Dealing with aggregated network measures, such as flows, helps in reducing the amount of data to be analyzed. A flow is defined as "a set of IP packets passing an observation point in the network during a certain time interval and having a set of common properties" [2]. A flow carries information about the source/destination IP addresses/ports involved in the communication, but nothing is known about the content of the communication itself.

Flow-based time series are a well-known way of visualizing network information [3]. Flow-based time series allow data processing in a streaming fashion, offer a compact representation of network traffic and allow to analyze data keeping into account the temporal relations between events. The drawback of flow-based time series is that we cannot have direct evidence of when an attack happened. In most cases, a ground truth is missing. Attack-labeled flow data sets are rare and their creation is a lengthy and time consuming process [4]. To overcome this problem, approaches based on the superposition of real non-malicious traffic with synthetic attack traffic have been introduced [5, 6]. For these approaches to work, we need models of network attacks.

In this paper, we propose a time-series based model of SSH brute-force attacks built upon the concept of Hidden Markov Models. We show that our model is able to emulate important aspects of the network behavior of such attacks and generates meaningful flow-based traffic time series.

This paper is organized as follows. Section 2 summarizes the related work on modeling of malicious traffic. Section 3 describes how SSH brute-force attacks are characterized at flow level. Section 4 presents our model for SSH malicious traffic. The model is evaluated in Section 5. Finally, conclusions are drawn in Section 6.

2 Related work

Hidden Markov Models (HMM) are effective in modeling sequential data [7]. Since they have been introduced in the early 1970s [8], they have been successfully applied to different scientific fields. Examples are biological sequence analysis [9], speech recognition [10] and pattern recognition [11].

Hidden Markov Model triggered the curiosity of many researchers also in the fields of Networking and Intrusion Detection. HMM can be trained on real data and their main characteristic is the ability to capture the temporal behavior of the observed processes. The work of [12] proposes to formalize traffic exchange in terms of "HMM profiles", a stochastic structure suited for sequence alignment. The results show that the models are able to classify traffic sequences at application level. In [13, 14], the authors propose a packet-level model of traffic sources based on HMM. The model proves to be effective in application classification. Moreover, a second fruitful application of the model in [13, 14] is in traffic prediction, namely forecast of short-term future traffic behavior. Similarly to these last contributions, we will use our model for traffic generation. Nevertheless, the work in [12–14] focuses on the packet-level, while we are interested in flow-based time series.

Hidden Markov Models are particularly appealing in Intrusion Detection, since they are able to calculate how likely a certain sequence of events is. Behavioral models for host-based intrusion detection have been proposed in [15] and [16]. The authors profile the normal sequence of system calls and raise alarms whenever a sequence is unlikely to be seen. Contrary to these contributions, we model malicious activities based on network data. A different approach is the one in [17], where the hidden states model the safety status of a network. Similarly to them, we also assigned semantic meaning to the hidden states. However, in our case the hidden states model the behavior of an attacker.

SSH brute-force attacks are a well known cyber threat [20, 21]. Although the attack is very common, it is still potentially dangerous. Our studies [4] showed that newly set up vulnerable hosts can be compromised within few days and be used as platform for the same attacks. We also showed that SSH attacks are visible at flow level as peaks in the SSH flow time series [22]. However, this observation tells us only how SSH attacks affect the total network traffic when such attacks are at their peak of intensity. In this paper we want to explore how the entire time series generated by a single attacker evolves in time.

3 Flow-based characterization of SSH brute-force attacks

Brute-force SSH attacks are one of the most common threats in cyber space [21]. In this section we qualitatively characterize such attacks at flow level, namely describing what a brute-force SSH attack looks like if only flow information is available. We analyze the traffic generated by a host known to have performed a SSH brute-force attack against our university network. The attack took place in the early afternoon of July 16, 2008 and lasted approximately 40 minutes. During this interval, approximately 8300 distinct university hosts have been attacked. The attack generated a volume of traffic of approximately 32,400 flows, 279,000 packets and 30.5MB.

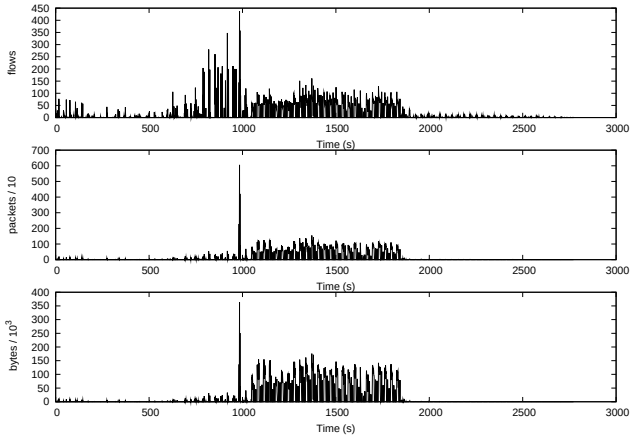
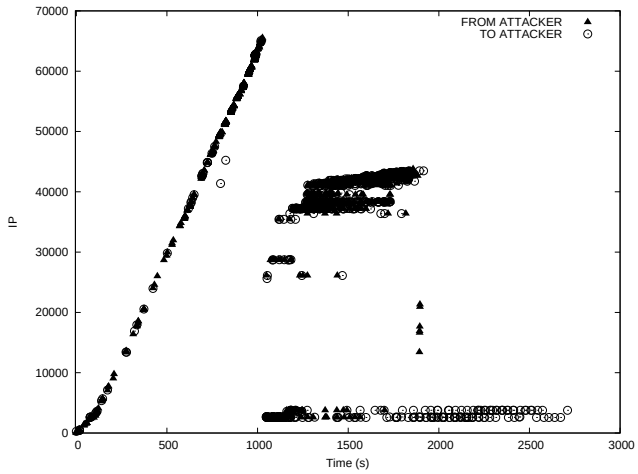
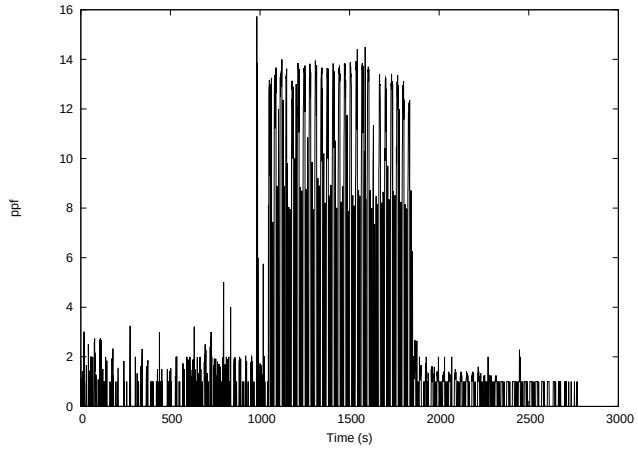


Fig. 1. Flow, packet and byte time series for a malicious SSH user

Figure 1 shows for the attacker the evolution over time of (i) the number of flows created per second, (ii) the number of packets transferred per second, and (iii) the number of bytes transferred per second. The time resolution of the time series is 1 second. Each value in the time series accounts for both the traffic generated by the attacker and the traffic that he receives from the victims. In this way, it is possible to characterize in the same time series the entire attack. During the attack, its intensity varies. In the flow time series we can see that in about the first 1000 seconds the attack intensity grows, reaching a peak of 450 flows/s. After that, the number of flows per second drops abruptly and roughly stabilizes around 100 flows/s. Finally, the attack activity slowly fades off in the last 500 seconds. Moreover, a deeper analysis of the flow time series shows that the activity pattern is not constant in time: each second of activity is often followed by one or more seconds of inactivity. The packet and byte time series closely follow the one of flows. Both show a peak around 1000 seconds since the beginning of the attack. As for flows, the trend of packets and bytes tends to stabilize for a while before the attack slowly dies. This behavior suggests that during an SSH brute-force scan, the flows, packets and bytes statistics are mutually correlated.



(a)



(b)

Fig. 2. Temporal visualization of a brute-force SSH scan (a) and variation of packets per flow during the scan (b)

A different view on the attacks is given by Figure 2(a). Each mark in the graph either represents a malicious connection from the attacker to a victim or the answering connection from the victim back to the attacker. The y-axis gives the 65,535 possible destination addresses in the university network. We identify *three attack phases*. During the *scanning phase* (first 1000 seconds), the attacker performs a sequential SSH scan spanning over the entire network address space. In this phases, the attacker gathers information on which hosts run a vulnerable SSH service. Only few victims respond to the attack. Once this phase is completed, the attacker initiates a brute-force user/password guessing attack (*brute-force phase*). In this phase, only a small subset of the hosts in

the network is involved. This phase corresponds to the second block of 1000 seconds and is characterized by a high interaction between the attacker and the victims. Finally, after about 2000 seconds since the beginning of the attack, the brute-force phase ends. Nevertheless, the time series in the previous picture have already shown that after this moment in time there is still traffic. Form Figure 2(a) it is now evident that the residual traffic is due to compromised hosts that communicate with the attacker. We refer to this final phase as the *die-off phase*.

Although the three attack phases are clearly visible in Figure 2(a), they are not so clearly identifiable from the flow, packet and byte time series shown in Figure 1. However, the fact that the three time series are correlated allows us to derive a more suitable measure. Figure 2(b) shows the evolution over time of the *packets per flow*, again with a resolution of 1 second. Using this measure, the three phases are clearly visible. The scanning phase is characterized by only few packets per flow, in average between 1 and 1.5. These values are consistent with a scenario in which several three-way handshakes are initiated but only few are completed. When the brute-force phase starts, the number of packets per flow has a sharp rise: from 1.5 to a average of about 11. During this phase, several user/password combinations are tested against the same victim. This explains why the attacker produces a higher number of packets per flow. Finally, the die-off phase sees again only few packets per flow. In the majority of the cases, we observe only one packet per flow. The variation of packets per flow over time seems therefore to be a key characteristic of the behavior of an SSH brute-force attacker. It moreover shows that the flow, packet and byte time series still carry enough information to characterize the attack.

The attack behavior that we described in this section is typical for a brute-force SSH attack. While monitoring the university network, we observe in average one attack with these characteristics per day. In the following section, we will describe how the flow-based characterization that we just proposed can be used to model the behavior of an SSH brute-force attacker.

4 Modeling with HMM

The analysis of a typical SSH brute-force attack that we presented in Section 3 pointed out three key characteristics of such attacks:

1. The flow, packet and byte time series exhibit a clear correlation.
2. Attacks consist of three phases: a scanning phase, a brute-force phase and a die-off phase.
3. The subdivision into phases may not be evident when we observe the flow, packet and byte time series directly, but it becomes manifest when we consider the packet per flow time series.

These key characteristics will play a central role in the following when modeling attacks using Hidden Markov Models (HMM).

This section is organized as follows. Section 4.1 briefly recapitulates the definition of HMM. In Section 4.2 we show how an SSH brute-force attack can be described as a Markov Chain. In Section 4.3 we describe the output probabilities associated to each

state and how they can be used to generate meaningful time series. Finally, Section 4.4 explains how the model parameters are computed from real data traces.

4.1 Hidden Markov Models

Hidden Markov Models are a class of statistical models able to describe sequences of data resulting from the interaction of several random processes.

Formally, an HMM is a discrete time Markov chain (DTMC) where each state is augmented with a probability distribution over a finite set of output symbols. Given a sequence of states $Q = q_1 q_2 \ldots$ with associated output symbols $K = k_1 k_2 \ldots$ we say Q forms the *hidden sequence* and K forms the *observation sequence*.

With $S = \{s_1, \ldots, s_n\}$ we denote the finite set of hidden states. S is called *hidden chain*. With q_t we denote the state at time t. With a_{ij} we denote the probability of jumping from state s_i to state s_j. Since we are dealing with a DTMC, this probability only depends on the current state s_i, i.e.: $a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i)$. With π_i we denote the probability of the initial state being s_i, i.e.: $\pi_i = P(q_1 = s_i)$. With $O = \{o_1, \ldots o_m\}$ we denote the finite set of output symbols. With k_t we denote the output symbol seen at time t. With $b_i(o)$ we denote the probability of seeing output symbol o when the hidden state is s_i, i.e.: $b_i(o) = P(k_t = o \mid q_t = s_i)$.

An HMM separates the state chain from the observable output. This key characteristic allows us to model malicious SSH attacks in an effective way and to generate synthetic flow, packet and byte time series.

4.2 The hidden chain

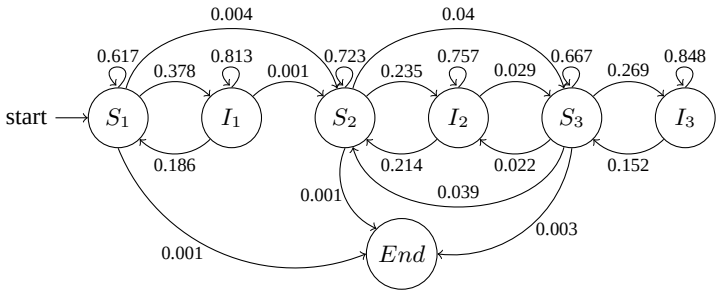


Fig. 3. HMM for the SSH brute-force attack with an example of transition probabilities learnt from real data

In Section 3, we explained that an SSH brute-force attack consists of three phases: a scanning phase, a brute-force phase and a die-off phase. We make use of these phases to define the hidden chain.

Our model consists of the following seven states:

– the states S_i, $i = 1, 2, 3$. In these states, the attacker is *active* and causes network traffic.
– the states I_i, $i = 1, 2, 3$. In these states, the attacker is *temporary inactive*, as described in Section 3.
– the end state End.

The state S_1 is the start state with $\pi_{S_1} = 1$. Figure 3 depicts the states and the possible transitions.

The states S_1 and I_1 model the scanning phase of the attack. As it can be seen in Figure 3, once the attack moves from the scanning phase to the brute-force phase, represented by the states S_2 and I_2, it cannot return to the previous phase. This ensures that the scan will not be performed more that once for each attack. On the other hand, the die-off phase (states S_3 and I_3) can partially overlap with the brute-force phase. This phenomenon is modeled by making the states $\{S_2, I_2, S_3\}$ a fully connected chain. However, the transition probabilities for this subset of states will privilege transitions in the same phase. Finally, the state End models the end of an attack. We allow the model to jump from each active state S_i to the End state, thus reflecting the fact that some attacks stop after the scan phase or the brute-force phase.

4.3 The output probabilities

The aim of our model is to generate meaningful synthetic flow, packet and byte time series for a SSH brute-force attack. Hence, at each transition, our model should output a triple (F, P, B) with the values for the three time series.

It is important to note that these three values are not independent, as shown in Section 3. Hence, to generate correctly correlated values for the three time series, a joint output probability distribution $P_{F,P,B}$ would be needed for each state of the model. In the following, we will present a different approach that approximates the triple-joint probability distribution $P_{F,P,B}$.

To each active state S_i, $i = 1, 2, 3$ in our model we assign the following two distributions:

1. an empirical probability distribution of flows P_F;
2. an empirical joint probability distribution of packets per flow (PPF) and bytes per packet (BPP), denoted as $P_{PPF,BPP}$.

At each transition, random values of F, PPF and BPP are generated according to the empirical distributions associated to the current state. Given the number of flows F, we assume the number of packets per flows and the number of bytes per packets to be the same for all the flows for this emission. We calculate

$$P = PPF \cdot F,$$
$$B = BPP \cdot PPF \cdot F.$$

The joint probability distribution $P_{PPF,BPP}$ and the indirect computation of P and B by the above expression ensure the strong correlation between F, P and B that we have observed in the data.

In the states I_i, $i = 1, 2, 3$, the attacker is by definition temporarily inactive and the triple $(0, 0, 0)$ is the only allowed output.

4.4 The parameter estimation

Once the hidden chain and the outputs of the model have been defined, we need to estimate the transition probabilities and the emission probability distributions for the states S_i, $i = 1, 2, 3$. Several methods for estimating the parameters of an HMM have been proposed in literature, for example the Baum-Welch algorithm [8], or the simulated annealing method of [23]. However, these methods are used when the training is based on sequences of observations only and the hidden state sequence is unknown.

In our training procedure, we follow a different approach. The analysis of the packets per flow time series, such as the one in Figure 2(b), offers us a way to precisely relate each observation in a trace with the hidden state that emitted it. We therefore manually labeled the traces in our training data sets. Once the hidden state sequence is known, we calculate each transition probability as

$$a_{ij} = \frac{|\{\text{transitions from } s_i \text{ to } s_j\}|}{|\{\text{transitions from } s_i\}|}.$$

Figure 3 gives an example of transition probabilities learnt from real data. The hidden state sequence is used to compute the output probabilities associated to each state. We calculate the distributions P_F and $P_{PPF,BPP}$ for a state S_i, $i = 1, 2, 3$ from the frequency histograms of the observations emitted from that state.

5 Validation

In this section we will evaluate the performance of the model proposed in Section 4. In particular, we will show that the model is able to generate synthetic traffic that has the same statistical characteristics of the SSH brute-force attack traces we used as training.

We based our validation on two data sets consisting of malicious SSH traces collected at the University of Twente network (*original data sets*). The validation proceeds as follows. First, we train an HMM for each distinct data set. Second, we use the models to generate groups of synthetic traces sufficiently large for the calculation of the confidence intervals. We refer to these traces as *synthetic data sets*. Third, we analyze the statistical properties of the synthetic data sets and we compare them with the original data sets. The aim of this analysis is to show that the model is able to encode sufficient information to correctly emulate the original traces.

Section 5.1 describes the data sets used for the training, while Section 5.2 explains how the synthetic traces are generated. Section 5.3 introduces our testing methodology. Finally Section 5.4 presents our results.

5.1 Original data sets

Our model of SSH brute-force attacks has been tested on two data sets. Each data set contains flow, packet and byte time series for a group of hosts known to have scanned our networks. The malicious hosts are all distinct.

The time series have been created considering time slots of 1 second. The volume of traffic for each time slot is comprehensive of both the traffic generated by the scanner and the traffic that it receives.

Table 1 presents the data sets. Both data sets have been collected during a monitoring window of one week on the network of the University of Twente. The offending hosts have been identified by a high interaction honeypot that is normally active in our network. *Set 1* has been collected in July 2008 and consists of 17 traces. *Set 2* has been collected in April 2009 and consists of 13 traces. Other hosts performing SSH malicious activities have not been considered part of the data sets since they appear to belong to a different class of scans. The statistical analysis of the two data sets shows that the average values of flows, packets and bytes over time have changed in time. In *Set 2*, the attackers appear to produce in average more that twice the amount of packets and bytes compared to *Set 1*. This suggests that, while the attack mechanism stays the same, the attacks' intensities are likely to vary in the course of time. As a consequence, models trained on real data would need periodical retrain.

Data Set	Collection time	Traces	Avg. Flows/sec	Avg. Packets/sec	Avg. Bytes/sec
Set 1	13-20 July 2008	17	11.06	66.91	7337.33
Set 2	19-26 April 2009	13	15.80	150.52	19016.00

Table 1. Statistical characteristic of the collected data sets

5.2 Synthetic trace generation

We define a synthetic trace as the sequence of observations that the model outputs when a random path is taken. The generation process can be summarized as follows. Let's assume the model to be in state s_i:

1. at time t, the model jumps from the current state s_i to the next state s_j according to the transition probabilities a_{ij}, $j = 1, \ldots n$.
2. if s_j is the End state, the path is concluded and the trace ends.
3. once s_j has been selected, the model randomly selects F, PPF and BPP.
4. the model outputs the triple (F, P, B), calculated on the basis of the random values generated in the previous step (as explained in Section 4.3).
5. once the observations have been emitted, the process iterates from step 1.

At each iteration, the model chooses which triple (F, P, B) will be emitted. This choice is independent from the previous outputs and is controlled only by the empirical probability distributions of F, PPF and BPP associated with the current state. Table 2 presents the range of these distributions for both *Set 1* and *Set 2*. The model controls also the duration of a trace, since a trace ends only when a transition to the End state is randomly selected.

5.3 Testing methodology

Our testing methodology aims to measure the average statistical characteristics of a set of synthetic traces and compare them to the ones of the original data sets *Set 1* and *Set 2*. Each statistical metric is calculated for flows, packets and bytes. We are interested in three types of statistical measures:

Distribution	Set 1		Set 2	
	Min	Max	Min	Max
F phase 1	1	789	1	3825
F phase 2	1	519	1	860
F phase 3	1	227	1	250
PPF phase 1	1	26.4841	1	27
PPF phase 2	1	16.5	1	17
PPF phase 3	1	5	1	5
BPP phase 1	40	156.42	40	225.71
BPP phase 2	50.88	267.27	52	319.42
BPP phase 3	40	836	46	1148

Table 2. Empirical distribution ranges for the training data sets

- the mean and standard deviation for flow (μ_F, σ_F), packets (μ_P, σ_P) and bytes (μ_B, σ_B). These measures describe the *overall behavior* of flows, packets and bytes independently of each other in a trace.
- the correlation coefficients ρ_{FP}, ρ_{FB} and ρ_{PB} between flows, packets and bytes. These measures describe the *dependence* between flows, packets and bytes in the same trace.
- autocorrelation of lag 1 of flows (R_F), packets (R_P) and bytes(R_B). The autocorrelation captures the *evolution* of a trace over time, measuring the interrelation of the trace with itself in different moments in time.

The previously introduced measures are relative to a single trace. In our experimental results, we calculate the average values of each measure for both the original data sets and the synthetic ones. For the synthetic trace, we also calculate the 95% confidence intervals. Each synthetic data set consists of 300 traces. Finally, we evaluate how well the synthetic traces approximate the original ones. In order to do so, we calculate for each measure m the relative error between the original traces and the synthetic ones:

$$Err = \frac{|m_{orig} - m_{syn}|}{m_{orig}}$$

5.4 Experimental results

This subsection presents the numerical results obtained from the analysis of the synthetic data sets. Table 3 offers an overview of the average statistical measures for both the original and the synthetic data sets. The same table also lists the relative error between original and synthetic measures. The results will be discussed in the following sections.

Average mean and standard deviation Both the model trained on *Set 1* and the one trained on *Set 2* approximate the averages of flows, packets and bytes within a 10% relative error.

	Set 1	Synthetic 1	Err	Set 2	Synthetic 2	Err
μ_F	11.06	12.27 $\pm$ 0.33	0.109	15.80	15.15 $\pm$ 0.65	0.041
μ_P	66.91	66.66 $\pm$ 3.67	0.046	150.52	138.85 $\pm$ 8.5	0.077
μ_B	7337.33	7524.73 $\pm$ 523.11	0.025	19016.00	18107.88 $\pm$ 1153.53	0.047
σ_F	36.45	38.33 $\pm$ 1.12	0.051	40.0	47.01 $\pm$ 1.87	0.174
σ_P	324.29	243.43 $\pm$ 10.91	0.249	379.38	419.16 $\pm$ 16.55	0.104
σ_B	28510.35	28345.60 $\pm$ 1616.63	0.005	47060.07	55378.58 $\pm$ 2239.91	0.176
ρ_{FP}	0.79	0.79 $\pm$ 0.012	0.001	0.83	0.86$\pm$ 0.01	0.039
ρ_{FB}	0.76	0.74 $\pm$ 0.016	0.023	0.79	0.81$\pm$ 0.01	0.024
ρ_{PB}	0.94	0.98$\pm$ 0.002	0.047	0.98	0.98$\pm$ 0.001	0.001
μ_{R_F}	0.46	0.23 $\pm$ 0.009	0.498	0.64	0.26 $\pm$0.01	0.593
μ_{R_P}	0.56	0.25 $\pm$ 0.012	0.547	0.71	0.30 $\pm$ 0.009	0.577
μ_{R_B}	0.58	0.26 $\pm$ 0.012	0.549	0.75	0.30 $\pm$ 0.009	0.592

Table 3. Numerical comparison between the original and the synthetic data sets.

The results also show that our approach approximates the standard deviation of both the original data sets within 10% relative error, with only few exceptions: the average standard deviation of packets for *Set 1* and the average standard deviations of flows and bytes for *Set 2*. Regarding *Set 1*, the synthetic measure underestimates the one in the original data set. On the contrary, in *Set 2*, the synthetic measures are higher than the original. We suspect this phenomenon is related to the autocorrelation of the traces in the original data sets.

Table 3 also presents the 95% confidence intervals for the average means and the standard deviations. For all measures, the confidence intervals are close to the average values.

Average correlation The correlation coefficients show that the proposed model is able to capture the interrelations between flows, packets and bytes, despite that the realizations of the random variables F and PPF are independently drawn. The relative error in the case of the correlation coefficients is indeed less or equal to 4.7% (ρ_{PB}) in the case of *Set 1* and less or equal to 3% (ρ_{FP}) in the case of *Set 2*.

In the same table we listed also the 95% confidence intervals for the average correlation coefficients. As in the case of the average relative error, described in the previous section, the confidence intervals are closed to the mean values.

Average autocorrelation The last measure we consider is the average autocorrelation. The autocorrelation characterizes the temporal evolution of a trace.

For both *Set 1* and *Set 2* our model fails to approximate the autocorrelation values. The autocorrelation of the synthetic traces, indeed, is roughly half of the autocorrelation in the original data sets. This means that consecutive values in a synthetic trace have a higher random component than in the original traces.

We believe that the cause of lower autocorrelation coefficients can be found in the attacker behavior during the brute-force phase. The original traces, indeed, show that

during this phase the time series presents a certain regularity, as for example a bounded number of flows per seconds. Our model, on the other hand, randomly selects at each iteration new values for flows, packets and bytes, without any memory of the previous outputs. This behavior is reflected in lower autocorrelation values. We consider to extend the model to capture regularity in the brute-force phase as a possible future work.

As for the previous measures, also in this case the confidence intervals show that the model has a low variability in the autocorrelation values.

6 Conclusions

In this paper, we have presented a compact model of SSH brute-force attacks based on Hidden Markov Models. The model has been inferred on the basis of only flow information and it encodes the network behavior of SSH attacks: scanning phase, brute-force phase and die-off phase. The model parameters have been calculated on the basis of real data traces captured at the University of Twente network.

In this paper we also demonstrate that the model, once trained on real data, is able to emulate the network behavior of a SSH brute-force attacker. Synthetic traces approximate the mean, standard deviation and correlation of flow, packet and byte time series within 10% relative error. The model fails only in approximating the autocorrelation. The synthetic traces, indeed, seem to have a higher random component than the original training trace.

As far as we are aware, this was the first time that HMM have been applied to the generation of flow-based time series for malicious users. The results are encouraging, but many aspects are open for future work. First, we aim to refine the model. For example, a more detailed model of the brute-force phase can improve the autocorrelation. In addition, the empirical emission distributions can be substituted by estimated distribution functions to make the model resilient to unforeseen observations. Second, we plan to adapt the model to be used for detection. In this context, we are also interested in investigating if the model we proposed is suitable for detection of other brute-force attacks that show a similar phase behavior. An example can be a brute-force attack against the telnet service. Third, we want to apply our HMM approach to other attack types, such as DoS attacks or worms.

References

1. Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., Stiller, B.: An Overview of IP Flow-based Intrusion Detection. IEEE Communications Surveys & Tutorials (to appear) (2009)
2. Quittek, J., Zseby, T., Claise, B., Zander, S.: Requirements for IP Flow Information Export (IPFIX). RFC 3917 (Informational)
3. NfSen - Netflow Sensor. http://nfsen.sourceforge.net (May 2009)
4. Sperotto, A., Sadre, R., van Vliet, D.F., Pras, A.: A Labeled Data Set For Flow-based Intrusion Detection. In: Proc. of the 9th IEEE International Workshop on IP Operations and Management (IPOM '09). (2009)
5. Brauckhoff, D., Wagner, A., Mays, M.: FLAME: a flow-level anomaly modeling engine. In: Proc. of the Workshop on Cyber Security Experimentation and Test (CSET'08). (2008)

6. Sommers, J., Yegneswaran, V., Barford, P.: A framework for malicious workload generation. In: Proc. of the 4th ACM SIGCOMM conference on Internet measurement (IMC '04). (2004)
7. Camastra, F., Vinciarelli, A.: Markovian models for sequential data. In: Machine Learning for Audio, Image and Video Analysis. (2008)
8. Baum, L.E., Petrie, T., Soules, G., Weiss, N.: A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. The Annals of Mathematical Statistics **41** (1970)
9. Durbin, R., Eddy, S.R., Krogh, A., Mitchison, G.: Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press (1998)
10. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. In: Proceedings of the IEEE. (1989)
11. Fink, G.A.: Markov Models for Pattern Recognition: From Theory to Applications. Springer-Verlag New York, Inc. (2007)
12. Wright, C.V., Monrose, F., Masson, G.M.: HMM Profiles for Network Traffic Classification. In: Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC '04). (2004)
13. Dainotti, A., Pescapé, A., Rossi, P.S., Palmieri, F., Ventre, G.: Internet traffic modeling by means of Hidden Markov Models. Computer Networks **52**(14) (2008)
14. Dainotti, A., de Donato, W., Pescape, A., Rossi, P.: Classification of Network Traffic via Packet-Level Hidden Markov Models. In: Proc. of IEEE Global Telecommunications Conference (GLOBECOM 2008). (2008)
15. Gao, D., Reiter, M.K., Song, D.X.: Behavioral Distance Measurement Using Hidden Markov Models. In: Proc. of 9th International Symposium Recent Advances in Intrusion Detection (RAID '06). (2006)
16. Warrender, C., Forrest, S., Pearlmutter, B.: Detecting Intrusions Using System Calls: Alternative Data Models. In: Proc. of the 1999 IEEE Symposium on Security and Privacy. (1999)
17. Khanna, R., Liu, H.: System approach to intrusion detection using hidden Markov model. In: Proceedings of the 2006 International Conference on Wireless communications and mobile computing (IWCMC '06). (2006)
18. Song, D.X., Wagner, D., Tian, X.: Timing Analysis of Keystrokes and Timing Attacks on SSH. In: Proc. of the 10th conference on USENIX Security Symposium (SSYM'01). (2001)
19. Albrecht, M.R., Paterson, K.G., Watson, G.J.: Plaintext Recovery Attacks Against SSH. In: Proc. of the 30th IEEE Symposium on Security and Privacy (SP '09). (2009)
20. Seifert, C.: Analyzing malicious ssh login attempts. http://www.securityfocus.com/infocus/1876 (September 2006)
21. SANS Institute: Top-20 2007 Security Risks (2007 Annual Update). www.sans.org (May 2009)
22. Sperotto, A., Sadre, R., Pras, A.: Anomaly Characterization in Flow-Based Traffic Time Series . In: Proc. of the 8th IEEE International Workshop on IP Operations and Management (IPOM '08). (Sept 2008)
23. Andrieu, C., Doucet, A.: Simulated Annealing for Maximum A Posteriori Parameter Estimation of Hidden Markov Models. IEEE Transactions on Information Theory **46** (2000)