Poster: Learning Models of a Network Protocol using Neural Network Language Models

Bernhard Aichernig[†], Roderick Bloem[*], Franz Pernkopf[‡], Franz Röck[*], Tobias Schrank[‡] and Martin Tappler[†]
[*]Institute of Applied Information Processing and Communications [†]Institute of Software Technology
[‡]Signal Processing and Speech Communication Laboratory
Graz University of Technology, Graz, Austria
{aichernig@ist., roderick.bloem@iaik., pernkopf@, franz.roeck@iaik., tobias.schrank@, martin.tappler@ist.}tugraz.at

***Abstract*—We present an automatic method to learn models of network protocol implementations which uses only network traces for learning. We employ language modelling techniques to infer a model for legitimate network communication which can then be used to identify network traces which deviate and thus have to be investigated.**

While previous work on learning models of network protocols often requires manual interference – especially fine-tuned adapters and abstractions – our experiment is performed in a highly automated manner. Furthermore, our model can process large vocabularies successfully which allows us to consider more complex protocols and to make use of more detailed abstractions than are currently in use. Most importantly, by using neural network language models with embeddings we are able to estimate the conformance of a test trace even if it contains messages that the model has not seen in training. To evaluate our work, we train a model on traces generated by nqsb-TLS, a modern implementation of the TLS protocol. In our experiment, we use this model to predict the next message given an arbitrary prefix with 92.9% accuracy and 58.0% F_1 score.

I. Introduction

The correct implementation of network protocols is a complex endeavour. Yet, for cryptographic network protocols correctness is of utmost importance to ensure security and integrity of network communication. Due to the ever existing need for stronger cryptography and for additional capabilities, cryptographic network protocols become a complex patchwork of extensions. Correctness of implementations can be established with model-based testing which aims at showing that the behaviour of an implementation does not differ from its intended behaviour as expressed by an explicit behavioural model [1]. However, large specifications pose serious issues to conventional model-based testing as generating models manually becomes increasingly difficult. To overcome this issue, the model can be learned from a reference implementation instead and then be used to apply model-based testing methods on arbitrary implementations of the same protocol. This model can also be used to identify suspicious traces in a running system.

In this paper, we apply methods from language modelling to automatically learn a behavioural model of the Transport Layer Security (TLS) protocol from a collection of network traces. TLS is today's most widely used transport layer cryptographic protocol (e.g., in *HTTPS*, *SIP*, *OpenVPN*). Most TLS implementations provide a wide array of TLS versions, protocol extensions, authentication modes and key exchange modes in order to be maximally compatible with clients. To deal with this complexity, we present an automatic method that indicates if an implementation behaves as it did in the training period. This method learns models of network protocols using a neural network with *message embeddings*. Similar to word embeddings in human language technology (HLT), message embeddings are projections of network messages into a real-valued space in which similar messages cluster. These embeddings enable us to provide meaningful estimates when encountering unseen messages during evaluation, a crucial feature for many real-world applications (e.g., intrusion detection systems [2]).

II. Related work

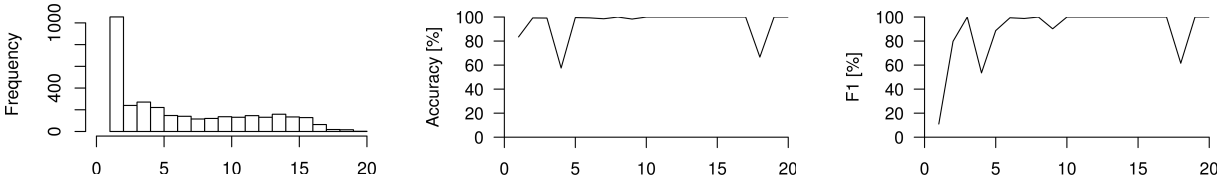
Models of network protocols have been learned in the field of reverse engineering but these often makes use of additional information aside from network traces. For instance, [3] perform dynamic taint analysis of the running executable in order to infer protocol message formats. However, a lot of security-critical and privacy-critical software runs on resource-constrained devices which are only accessible via network interfaces.

Finite state machines (FSM) of protocol implementations can be learned with the learning algorithm L^* [4]. L^* infers the implementation's actual FSM and makes it accessible to manual inspection [5]. At the same time L^* is restricted to learning simple protocols or highly abstracted versions of protocols due to high computational cost (cf. [6]). Sequence modelling with large vocabularies has first been studied in the context of HLT. Statistical language models in their simplest form are Markov chains assigning probability to a sequence of words [7]. In recent years, language models based on neural networks (NNLM) have been introduced [8] and extended to allow for variable-length dependencies [9]. They also embed words into a real-valued space in which relationships between words are reflected by a relation-specific vector offset [10]. As a result, closeness in this space is indicative of similarity and is thus a form of clustering.

III. TLS experiment

A. Data

In our experiment we used a collection of roughly 52k TLS network traces as output by the tracing component of [11]. We



(a) Distribution of prefix lengths of network traces in the test set

(b) Accuracy

(c) F_1 measure

Fig. 1: Accuracy and F_1 of test set in dependence of prefix length.

only use network traces and exclude other information present in this data, i.e., machine state information.

The data is formatted as a sequence of trees of message fields. In order to limit the vocabulary size we ignored any information below the third level (e.g., `Handshake-In ClientHello Random 01234...` → `Handshake-In ClientHello Random`). However, this happens in a completely automatic manner and thus many fields remain in the data that are irrelevant to our experiment.

B. Method

In our experiment, we employ a long short-term memory (LSTM) NNLM that we implemented using Torch [12]. The neural network has 2 hidden layers with 128 neurons each and embeds messages into a 64-dimensional continuous space. This embedding space allows the model to estimate the similarity of messages it has not seen at training time (e.g., `TLS 1.3` in the future).

To evaluate our model we split the data into a training set (80%), a development set (10%) and an evaluation set (10%). For the final score we take prefixes of random length from the evaluation traces and let the model predict the next message. We compare this predicted message with the true message found in the evaluation set and then compute accuracy and F-measure (F_1). Due to the high number of parameters in TLS headers, the frequency distribution of message types is heavy-tailed which skews accuracy (see Figure 1a). F_1 takes frequency of message type into account and thus is a more balanced performance metric.

C. Results

For the evaluation traces, the model predicts the next message correctly in 92.9% of the cases. This is a good result considering the vocabulary size of 24k. However, an considerably lower F_1 of 58.0% reveals that the model's performance is severely impeded by the large number of infrequent messages. Analysis of performance measures in dependence on prefix length (see Figure 1b, c) indicates that the model fails to predict the `ServerKeyExchange` message with high accuracy when Diffie-Hellman cipher suites are used (prefix length 4). The drop in accuracy for prefixes of length 18 is caused by either receiving empty or non-empty `ApplicationData` messages after a successful handshake.

IV. Conclusion

We presented a method to automatically learn behavioural models of network protocols to identify suspicious traces. To learn these models we adapted methods from language modelling that are particularly well-suited for dealing with large vocabularies and unseen data. We evaluated our model on 52k traces generated by nqsb-TLS. This led to our model correctly predicting the next message given a trace prefix in 92.9% of the cases and an F_1 score of 58.0%.

While we were able to estimate similarity between messages by means of their context reasonably well, we would prefer to use also the information inside a message (i.e., information extracted from its fields). Our model can be extended to make use of this kind information similar to sub-word modelling in HLT [13].

Acknowledgment

This work was supported by Graz University of Technology through the LEAD Project "Dependable Internet of Things in Adverse Environments" and the European Commission through project IMMORTAL (317753).

References

[1] M. Utting, A. Pretschner, and B. Legeard, "A taxonomy of model-based testing approaches," *Software Testing, Verification and Reliability*, vol. 22, no. 5, pp. 297–312, 2012.

[2] K. Rieck and P. Laskov, "Language models for detection of unknown attacks in network traffic," *Journal in Computer Virology*, vol. 2, pp. 243–256, 2007.

[3] P. Milani Comparetti, G. Wondracek, C. Kruegel, and E. Kirda, "Prospex: Protocol specification extraction," in *IEEE Symposium on Security & Privacy*, 2009.

[4] D. Angluin, "Learning regular sets from queries and counterexamples," *Information and Computation*, vol. 75, pp. 87–106, 1987.

[5] J. de Ruiter and E. Poll, "Protocol state fuzzing of TLS implementations," in *24th USENIX Security Symposium*, Aug. 2015, pp. 193–206.

[6] F. Aarts, H. Kuppens, J. Tretmans, F. Vaandrager, and S. Verwer, "Learning and testing the bounded retransmission protocol," in *JMLR*, vol. 21, 2012, pp. 4–18.

[7] J. T. Goodman, "A bit of progress in language modeling," *Computer Speech & Language*, pp. 403–434, 2001.

[8] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *JMLR*, vol. 3, pp. 1137–1155, 2003.

[9] T. Mikolov, M. Karafiát, J. H. Černocký, and S. Khudanpur, "Recurrent nureal network based language model," in *INTERSPEECH*, 2010, pp. 1045–1048.

[10] T. Mikolov, W. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proceedings of NAACL HLT*, 2013.

[11] D. Kaloper-Meršinjak, H. Mehnert, A. Madhavapeddy, and P. Sewell, "Not-quite-so-broken TLS: Lessons in re-engineering a security protocol specification and implementation," in *24th USENIX Security Symposium*, 2015, pp. 223–238.

[12] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS Workshop*, 2011.

[13] T. Mikolov, I. Sutskever, A. Deoras, H.-S. Le, S. Kombrink, and J. Černocký. Subword language modeling with neural networks. [Online]. Available: http://www.fit.vutbr.cz/~imikolov/rnnlm/char.pdf