Bayesian Models for Targeted Cyber Deception Strategies (Student Abstract)

Nazia Sharmin

University of Texas, El Paso, 500 W University Ave, El Paso, TX 79968
nsharmin@miners.utep.edu

Abstract

We propose a model-driven decision support system (DSS) based on a Bayesian belief network (BBN) to support cyber deception based on a detailed model of attacker beliefs. We discuss this approach using a case study based on passively observed operating system (OS) fingerprinting data. In passive reconnaissance attackers can remain undetected while collecting information to identify systems and plan attacks. Our DSS is intended to support preventative measures to protect the network from successful reconnaissance, such as by modifying features using deception. We validate the prediction accuracy of the model in comparison with a sequential artificial neural network (ANN). We then introduce a deceptive algorithm to select a minimal set of features for OS obfuscation. We show the effectiveness of feature-modification strategies based on our methods using passively collected data to decide what features from a real operating system (OS) to modify to appear as a fake [different] OS.

Background

Passive reconnaissance is one of the core techniques used by cyber attackers, and recently attackers have placed more importance on stealth to remain undetected in the target system and prevent detection by the defender or intrusion detection system (IDS) (Pham et al. 2020). Adversaries employ a variety of methods to collect information, which is subsequently leveraged to exploit specific vulnerabilities. Existing methods such as firewalls and cannot totally secure a network (Hagos et al. 2020), so one of the most promising directions for actively defeating reconnaissance is the use of cyber deception strategies. However, most existing methods for cyber deception have very limited models of how attackers actually perform reconnaissance and form beliefs about the network. We propose decision support tools for cyber deception based on more detailed modeling of attacker beliefs, which can support targeted decisions of which deceptions are most likely to affect critical adversary knowledge and beliefs.

We first present a case study illustrating the decision-support system (DSS) framework. Then, we show the procedure to build a model using passively collected data and include it in the DSS framework. Finally, we provide a case study for the DSS framework using data associated with the operating system's features. We intend to use it to analyze features that aid in identifying OS and then use it for implementing deception. State-of-the-art research proposes various OS identification techniques. Wei-Hua et al. (Weihua, Wei-hua, and Jun 2003) utilize characteristics of ICMP packets to detect OS. Robert (Beverly 2004) presents a passive OS detection method based on packet headers and develops a Naive Bayesian classifier to passively infer a host's operating system from packet headers.

Several studies have used machine learning algorithms to identify operating systems (Song, Cho, and Won 2019; Khatouni et al. 2019; Al-Shehari and Shahzad 2014). In our case study, we analyze traffic generated from operating systems and build a knowledge base/model that can help to understand which operating system features cause the OS to be more vulnerable and easily identifiable. For example, TCP features are almost identical between Mac and iOS; as a result, using solely TCP features tends to misclassify iOS as Mac or vice versa (Lastovicka et al. 2018). Android is also easily distinguishable from other common OSs like iOS, Mac, and Windows as it exhibits some distinctive TCP features. As a result, learning from the data to create a model can be useful to extract meaningful information to protect the network and feature deception. Our models can be learned using a wide variety of passively collected network data.

Current Results

Our framework for the DSS has three components i) **Database module:** a database containing passively collected data. ii) **Model management module:** where the user can build one or more models based on the problem and flexibility of implementation. We generate a Bayes net from the data (Figure 1) and learn the conditional probability table (CPT). By analyzing the CPT table the defender can infer the corresponding importance of feature value for each operating system. We compare our model with a baseline sequential ANN. ANN has been used in OS fingerprinting (Song, Cho, and Won 2019), and it performs well in terms of OS prediction. However, the ANN is generally considered a black box model; it is difficult to show the relationship between cause and effect with ANN, whereas our model can be used for prediction and different types of analysis and in-

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

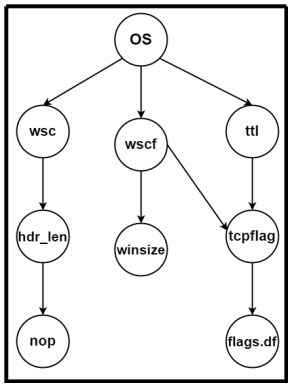

Figure 1: Bayes net for model .

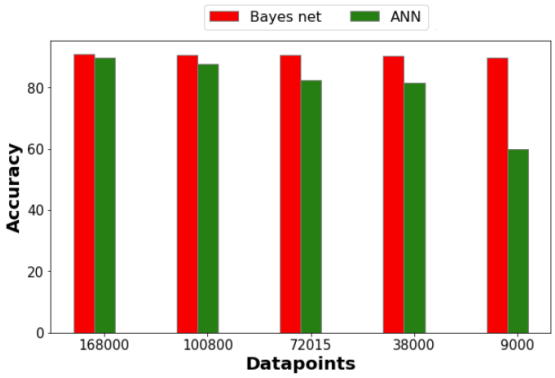

Figure 2: Accuracy of BBN vs ANN.

terpretation. Figure 2 shows the accuracy of the Bayes net in comparison with the ANN for our data set. We observe that the Bayes net performs well in prediction.

iii) **A dialog module:** The user can perform what-if-analysis (inference), change inputs, access the data, and modify models. We use the dialog module for causal inference and predictive and diagnostic reasoning. We build a deception algorithm using the CPT table used in DSS.

Deception Algorithm: We utilize the CPT table to build a deception algorithm. The purpose of the deception algorithm is to select a minimal set of features that is still effective for deception. The deception algorithm is implemented on a real OS to behave like another (fake) OS (figure 3). We apply the ML algorithm to deceptive and non-deceptive data to evaluate the deception. Figure 3 shows that recall and f1 scores of Linux dropped after applying deception on Linux (Ubuntu). Here, we modified the feature utilizing our deception algorithm in order to make Ubuntu appear as Mac. Learning from the data and building a knowledge base can be used for what-if analysis and infer whether a particular asset can be protected. Furthermore, we can analyze the plausibility of feature deception. Our case study is based on Os data, but the general procedure can be applied to many different types of data and attacker reconnaissance decisions, providing the ability to make deception decisions that are targeted to specific attacker goals.

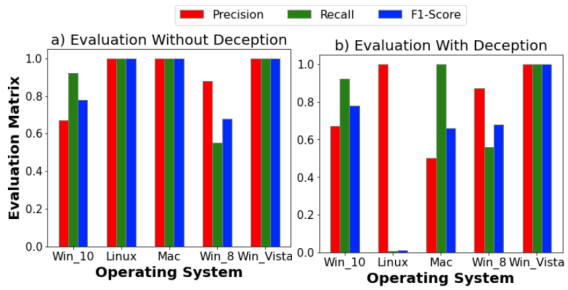

Figure 3: Deception implemented in linux (Ubuntu) (b)

Acknowledgments

This work was supported by the Army Research Office under award W911NF-17-1-0370.

References

Al-Shehari, T.; and Shahzad, F. 2014. Improving operating system fingerprinting using machine learning techniques. *International Journal of Computer Theory and Engineering*, 6(1): 57.

Beverly, R. 2004. A robust classifier for passive TCP/IP fingerprinting. In *International Workshop on Passive and Active Network Measurement*, 158–167. Springer.

Hagos, D. H.; Yazidi, A.; Kure, Ø.; and Engelstad, P. E. 2020. A Machine-Learning-Based Tool for Passive OS Fingerprinting With TCP Variant as a Novel Feature. *IEEE Internet of Things Journal*, 8(5): 3534–3553.

Khatouni, A. S.; Zhang, L.; Aziz, K.; Zincir, I.; and Zincir-Heywood, N. 2019. Exploring nat detection and host identification using machine learning. In *2019 15th International Conference on Network and Service Management (CNSM)*, 1–8. IEEE.

Lastovicka, M.; Jirsik, T.; Celeda, P.; Spacek, S.; and Filakovsky, D. 2018. Passive os fingerprinting methods in the jungle of wireless networks. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, 1–9. IEEE.

Pham, L. H.; Albanese, M.; Chadha, R.; Chiang, C.-Y. J.; Venkatesan, S.; Kamhoua, C.; and Leslie, N. 2020. A quantitative framework to model reconnaissance by stealthy attackers and support deception-based defenses. In *2020 IEEE Conference on Communications and Network Security (CNS)*, 1–9. IEEE.

Song, J.; Cho, C.; and Won, Y. 2019. Analysis of operating system identification via fingerprinting and machine learning. *Computers & Electrical Engineering*, 78: 1–10.

Wei-hua, J.; Wei-hua, L.; and Jun, D. 2003. The application of ICMP protocol in network scanning. In *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*, 904–906. IEEE.