# Using machine learning to create a host-based intrusion detection system

Noah Zbozny
Mentored by John Burghardt

## Introduction

Current cybersecurity solutions struggle to identify new attacks quickly enough to be effective, driving the need for new methods capable of identifying new attacks as they occur. Cao, Badger, Kalbarczyk, Iyer, and Slagell (2015) found that behavioral Intrusion Detection Systems (IDS's) were able to find attacks that professional security analysts failed to discover.

Network devices can be accessed and their security bypassed by remote users with tools such as a secure shell (SSH) connection. An SSH honeypot is a machine designed to look vulnerable, but actually capture connection attempts to provide the system administrator with data about network connections without the remote user's knowledge. An SSH honeypot set up for this project saw over 3,000 malicious login attempts in 10 days of data collection. A common way to examine such large amounts of data and identify meaningful patterns within them is through machine learning algorithms. Machine learning is a type of artificial intelligence designed to be capable of pattern recognition with multi-dimensional data. The Random Forest algorithm runs data against randomly created groupings of variables to identify the most important variables and reduce generalizations in its predictions.

This project develops an IDS using SSH honeypot data as input to the Random Forest machine learning algorithm to predict if the activity is malicious and alert the system administrator accordingly. The process was automated to check for new connection attempts, parse the data, and create a list of predicted malicious IP addresses for the system administrator to blacklist.

## Materials and Methods

The IDS application was developed to monitor SSH connection activity on Ubuntu Linux machines using C++ and the Python Random Forest library. The base system is a cloud server running Ubuntu rented from DigitalOcean™ Inc. with the Cowrie SSH honeypot. The honeypot logged remote connection data such as the attempted username and password, IP address, input commands, and duration of the connection. Code was written to parse the output from the honeypot into a Comma Separated Values (CSV) file to be read by the machine learning (ML) algorithm. The machine learning code was then written in Python with the SciPy™ library to read the CSV file, run the data through the Random Forest algorithm, and output the predictions to another CSV file. If the connection attempt was predicted to be malicious, it also output the prediction to the console. The logic for the main loop is shown in Figure 1.

## Materials and Methods (cont.)

The ML algorithm provides a list of potential intruders to the system administrator, which they would then have to blacklist manually. The blacklisting could have been done by the software if it was given root privileges, but doing so would open up more potential vulnerabilities.

In addition to anonymous hits from the internet, data collection was performed by having SMA students log into the honeypot. Depending on whether the student was assigned to be a malicious or benign user, they would behave as such to test the algorithm's effectiveness. To do so they would perform certain actions that were mostly performed by malicious actors, such as attempting to change their privileges or download files from outside networks.

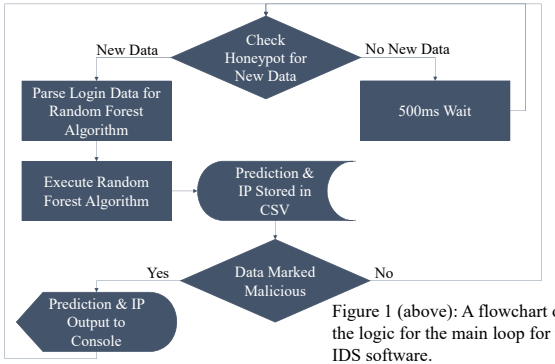
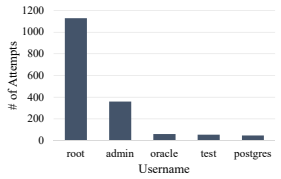


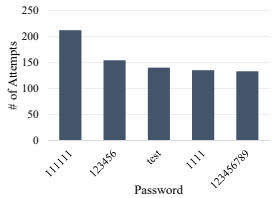

Figure 1 (above): A flowchart of the logic for the main loop for the IDS software.

## Results

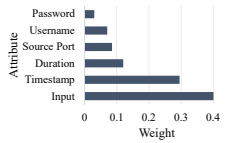
Top 5 usernames attempted




Top 5 passwords attempted




Importance by attribute




Graph 1 (upper left): The 5 most common usernames attempted by attackers on the honeypot. Graph 2 (upper right): The 5 most common passwords attempted by attackers on the honeypot. Graph 3 (lower left): The ranking of each attribute and how much it contributed to the Random Forest algorithm.

## Results (cont.)

| Classification of Predictions | Observed | Test Proportion | Expected | Contribution to $\chi^2$ |
|---|---|---|---|---|
| Correct | 145 | 0.5 | 87.5 | 37.7857 |
| Incorrect | 30 | 0.5 | 87.5 | 37.7857 |

Table 1 (left): Table of correct and incorrect prediction counts and their $\chi^2$ values.

A $\chi^2$ goodness-of-fit test was performed to examine if the application was randomly classifying variables, and it was determined that classification was not random. $\chi^2(1, n = 175) = 75.5714, p < 0.001$, as shown in Table 1 (above). Graphs 1 and 2 (bottom left) show the 5 most common usernames and passwords attempted by attackers on the honeypot. Graph 3 demonstrates the value each attribute contributed to the algorithm.

## Conclusions

This project offers developers and system administrators a new method to protect their servers from malicious login attempts. Table 1 (above) shows that the algorithm correctly classified 84% of the connections selected for testing, However, as with any machine learning software, results can be improved with more data. The algorithm had issues with generalizing its predictions to the training data, which was very similar to the testing data as it was all collected in the same way and from the same source. For the software to be implemented on another network, data would have to be collected to train the algorithm, as traffic is often vastly different between networks. The application could be improved mainly by furthering the capabilities of the IDS, such as by trying a different algorithm to better improve the behavioral analysis of intrusion attempts. It was observed that while the benign users were connecting via the default SSH port (port 22), many of the malicious users connected through a different port. A way to protect against this would be to prohibit SSH connections from ports other than port 22. It was also observed that many of the malicious logins were entered very quickly, so access to the network could also be restricted in the event that a login or commands were input faster than a human could type. This increased capability would turn it into an Intrusion Prevention System.

## References

Cao, P., Badger, E., Kalbarczyk, Z., Iyer, R., & Slagell, A. (2015). *Preemptive intrusion detection: Theoretical framework and real world measurements*. Retrieved from https://publish.illinois.edu/science-of-security-lablet/files/2014/06/Preemptive-Intrusion-Detection-Theoretical-Framework-and-Real-World-Measurements.pdf

Kaggle. (2013). *Titanic: Machine learning from disaster*. Retrieved from https://www.kaggle.com/c/titanic

Oosterhof, M. (2017). Cowrie (2017, April 27) [Software]. Available from https://github.com/micheloosterhof/cowrie