

## Methods

The method utilizes the Amazon Machine Learning software. It will be trained on a dataset comprised of normal user situations, crafted mistakes, and malicious activity. The software will use the training dataset, to make predictions against similar datasets to verify accuracy. It will then be tested against a human actor to test predictions in a high-fidelity simulation. A testing dataset of 100 commands will be given to the model for batch prediction and will be evaluated for statistical accuracy.

## Motivation and Introduction

Insider threats are on the rise. Current commercial software can monitor, log, and prevent access to designated files and directories. However, it remains difficult to predict and prevent unauthorized insider usage. Due to the gaps in research in the area, the focus of this study is to more accurately predict insider threats in a server environment.



Northern Arizona University - Prescott Valley  
School of Informatics, Computing,  
and Cyber Systems

## Why the linux terminal?

Linux has been chosen because of the ubiquity of the operating systems based on it are connected to the internet in servers around the globe. It is also transferable to all POSIX-compatible terminal environments that are in use today.

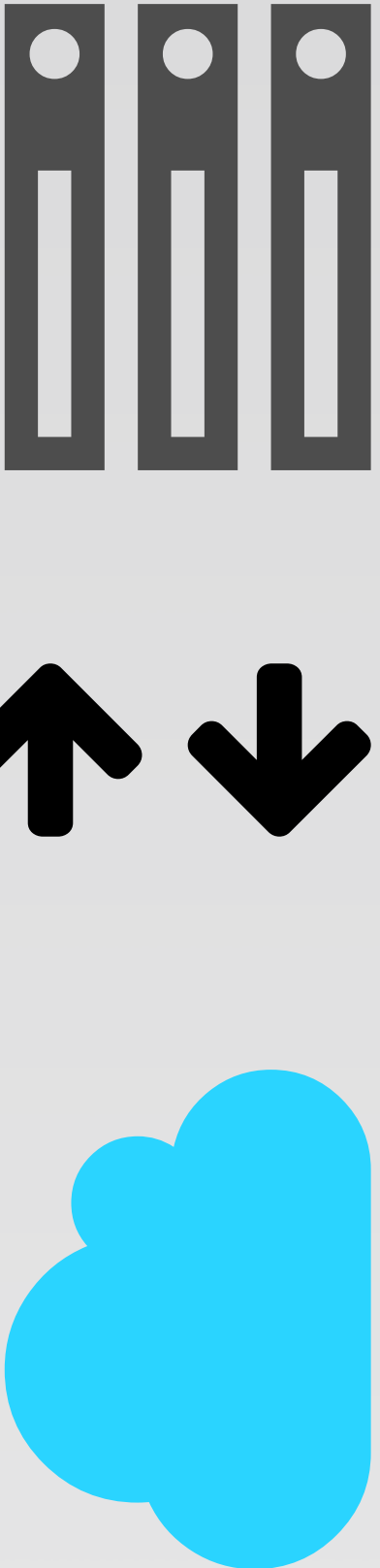
## Data Sources

There are three main sources of data used for the machine learning model. The first is an un-edited history file from the test system. Second is a list of commands assembled during this research. The third is the UNIX User Dataset from Purdue University. Simple programs were developed to remove unnecessary lines from the two gathered datasets.

## Results

The batch predictions have shown that the model is 100% accurate at predicting normal use and has a %30 rate of false negatives for malicious and mistake commands. The results of the high-fidelity simulation indicate that the software is partially effective at capturing all input. The participants were able to escape it and after that became untraceable to my system. This indicates that further work is needed in this area. Overall the process has indicated that a prototype like this can capture near 80% of the commands and suggests that further work on this model can produce far more reliable results.

## Send to AWS for Machine Learning Evaluation



```
def get_prediction(request):  
    # Correctly format request  
    request = {"Var1":str(request)}  
  
    # Send JSON-formatted request  
    response = client.predict(  
        MLModelId=modelid,  
        Record=request,  
        PredictEndpoint=endpoint  
    )  
  
    return response
```

## Conclusions

The results of the test indicate the model isn't sufficiently accurate in detecting intrusion even when given special weighting. This is likely due to the lack of data in the malicious and mistake categories. Future research in the area needs more open data. One solution is to ask organizations to publish the cleaned logs after an attack.

This study also indicated that further improvements are needed to the program to accurately capture information from a live terminal. While the program created a fake prompt it was severely limited in what it could capture and that skewed the data from the simulation.

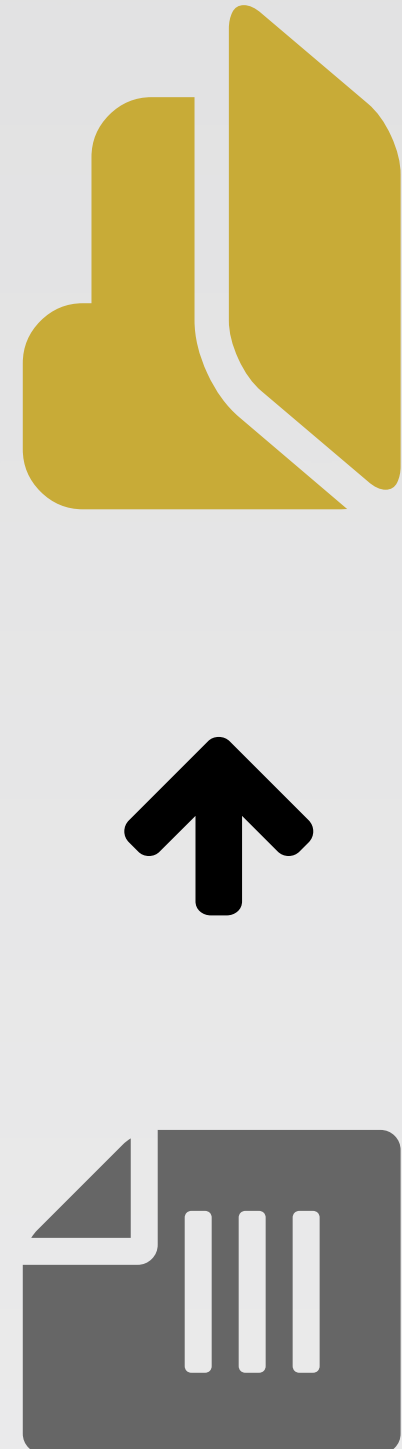
Future research is needed in order to determine how to best approach the problem of practical proactive intruder detection.

As other researchers have indicated, larger datasets comprising of more points such as time-stamps and keyboard information would help better track user behavior in an attempt to establish recognizable patterns by neural networks. This and other research indicate that more data is needed.

## Data Sources

There are three main sources of data used for the machine learning model. The first is an un-edited history file from the test system. Second is a list of commands assembled during this research. The third is the UNIX User Dataset from Purdue University. Simple programs were developed to remove unnecessary lines from the two gathered datasets.

## Log Command and Evaluation



```
def log(command, evaluation):  
    with open('logfile', 'a') as f:  
        f.write(datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')  
            + ", " + str(command) + ", " + str(evaluation) + "\n")
```

Joshua Bowen  
jdb567@nau.edu  
<http://joshuabowen.info>

## Warn of Mistake

```
# if mistake, do not execute command  
elif ev <= 0.75 and ev >= 0.25:  
    print('\nI think your command may  
        be or contain a mistake, please try again.')
```

## Prevent Malicious Activity

```
# if determined malicious, stop session  
elif ev < 0.25 and ev >= 0:  
    print('\nIntruder detected!')  
    sys.exit(1)
```

## Allow Normal Use

```
# if good, allow command  
if ev > 0.75:  
    os.system(cmd)
```



## Get User Input

```
def get_input():  
    # Update username, hostname, and working dir from terminal  
    username = os.environ.get('USER')  
    hostname = os.environ.get('HOSTNAME')  
    wdir = os.environ.get('PWD')  
  
    # Build command prompt  
    prompt = str(username).rstrip() + '@' + str(hostname).rstrip() +  
        ':' + str(wdir).rstrip() + '#'  
  
    # Get user input  
    cmd = input(prompt)  
  
    # Convert to and understndable string  
    cmds = str(cmd)  
  
    return cmds
```

## Setup Connection

```
# Set up Amazon Web Services connection  
client = boto3.client('machinelearning')
```



# A Practical Application of Machine Learning-Based Classification Techniques to Proactively Identify Insider Threats

**References, Source Code, and Material are available at:**  
<http://github.com/Skraelingjar/cyberml>

## Acknowledgements

Faculty Advisor - James A. Subach, Ph. D  
Academic Credit Advisor - Nancy L. Jensen  
Code Help - Ian Harvey on Stack Overflow  
Data Sources - Purdue University  
Poster Template - Felix Breuer  
The NAU Cyber Security Team  
The Embry-Riddle [Ethical] Hackers Club  
Amazon Web Services  
Font Awesome  
GitHub