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

Northern Arizona University
School of Informatics, Computing, and Cyber Systems

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

Methods

The method adapted is to utilize the Amazon Machine Learning software, it will be trained on a dataset comprised of normal user situations, crafted mistakes, and malicious activity. After providing the training dataset, the software will be instructed 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 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.

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 my laptop. Second is a list of commands assembled from my research. Finally is the UNIX User Dataset from Purdue University. Due to the nature of of the two gathered datasets I wrote simple programs to remove unnecessary lines. Due to the lack of open data showing malicious and mistake commands, the vast majority of data points falls into the normal category.

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')
```

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
```

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 agian.')
```

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(com)
```

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")
```

Results

After completion of the prediction it has been shown that the model is off by about 20% when predicting malicious or mistake commands.

The results of the high-fidelity simulation show that not only is the software I wrote ineffective as a terminal but that it is also easy to bypass. the participants were able to escape it and after that became untraceable to my system.
Overall the process has not proved anything of statical significance.

Conclusions

Based on the results of the tests it has been determined that the model isn't that accurate, 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.

Another problem identified during this study is the programming needed to accurately capture information from a live terminal. While I was able to create a fake prompt it was severely limited and that skewed the data from the simulation.

Future research will be needed in order to determine how to best approach the problem of practical proactive intruder detection. I theorize that larger datasets comprising of more points such as timestamps and keyboard information would help better track user behavior in an attempt to establish recognizable patterns by neural networks.

Acknowledgements

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

Refrences, Source Code, Material

http://github.com/Skraelingjar/cyberml