COMS30038 — SECURITY BEHAVIOURS

LAB 2: INCIDENT ANALYSIS

In this session, you will be building and annotating attack trees in small groups, in order to gain an understanding of how you can identify weaknesses and likely attack paths in a hypothetical system. You will also be given some system logs to analyse, answering questions about suspicious network activity.

Finally, there is an exercise that challenges you to build a simple anomaly detection system, to give you some experience of one of the ways that intelligent systems can be deployed in cybersecurity.

1 ATTACK TREES

Split into smaller groups for brainstorming purposes. For this exercise you can use a multi-user online whiteboard like `https://witeboard.com`, or even just a piece of paper.

Consider an online banking system, like the one used by Barclays[1]. Create a group in the online tool and collaboratively build an attack tree towards the goal of *reading* a given user's bank account balance. If you need to make it more specific, let's say the person you're targeting is the new vice-chancellor of Bristol University.

Build your tree with breadth and depth, considering a range of both technical and social approaches to the problem. Use 'or' nodes to express alternatives, and 'and' nodes to express a set of actions that all need to be taken to achieve a parent (see the lecture notes for further guidance).

1. Once your tree is well-populated, create a `cost` attribute to express how expensive an action might be. Fill in your best guesses at the approximate cost for each *leaf node* – the child nodes with no children. Note how the costs percolate up the tree. Which of your top-level strategies would be the cheapest for an attacker?

2. Create a `probability` attribute to express the likelihood of an action being carried out successfully. Again, label the leaf nodes and identify which of your top-level strategies is the most probable for an attacker.

[1] If you like, select another system that your group is more familiar with.

3. Do cost and probability point towards the same strategies in your tree? Could you see a way to balance them?

2 LOG ANALYSIS

Let's now briefly look at things from the defensive perspective. Imagine you're working in a security operations centre for a large organisation (like a university). You think there has recently been a security breach, and your job is to investigate how it may have entered or spread through the organisation.

Working individually, inspect the SSH logs provided[2]. Your task is to answer the following questions:

1. How many failed connection attempts are there, as a proportion of all the data?

2. Which host is originating the most failed connection attempts?

3. Which host is receiving the most failed connection attempts? Where from?

4. Which origin hosts have attempted to connect to the most server IP addresses?

5. What is the UID for the most recent successful connection?

6. At what time of day did it occur?

[2] Using Python, or any other analysis tool you prefer.

3 ANOMALY DETECTION

The previous exercise had you manually inspecting logs for possible suspicious activity – this is investigative work that you might perform after an intrusion. As discussed in the lectures, one of the tools that you might use to *detect* an incident is an anomaly detection system, an approach which involves building a model of 'ordinary' system behaviour, and then detecting anomalous deviations from that normal pattern. You're going to explore one of the simplest anomaly detection approaches.

Start by downloading the nyc_taxi.csv data. The methods we're using could as easily be applied to the number of bytes being sent over a network, or some measure of activity on host machines, but here the data is actually the number of passengers in all New York City taxicabs in any given 30-minute period for a few months of 2014/2015. Your task is to design a model to identify any anomalies in the dataset.

You'll want to start by loading in the data and setting aside some of it to use as your 'baseline' – your level for what 'normal' activity for this system looks like. A typical period to use for this might be a week's data. Here's some code to get you started in Python, but feel free to use a different language or toolset if you're more comfortable with it.

```
import csv
import statistics

csvin = csv.DictReader(open('nyc_taxi.csv','r'))
all_data = []

for row in csvin:
    row['value'] = int(row['value'])
    all_data.append(row)

# 48 half-hours * 7 = 336
train = all_data[:336]
test = all_data[336:]
```

One of the simplest models possible is to look for deviations from the average. Calculate the mean value for the 'baseline' train data.

Once you have your mean, the next question is how you can define an 'anomalous' deviation from it. If you recall your introductory statistics, you'll know that in normally-distributed data, 68% of values should fall within 1 standard deviation of the mean, 95% should fall within 2 standard deviations, and 99.7% should fall within 3 standard deviations. Drawing on this[3] write a function that will look for values in test which are further than 3 standard deviations away from the baseline average.

With luck, you should now have identified your first anomalies in the

[3] And looking at the methods available in the statistics package.

dataset. The date associated with the points you've landed on should match the date of the 2014 NYC marathon – a big, disruptive event which clearly had knock-on effects on the taxi industry. Our very simple model was able to identify this as anomalous.

However, the simplest model isn't necessarily the correct one. If you look at the values within `train` you should be able to see a pattern repeating a few times – a daily cycle of life in the city, in half-hourly snippets. Some times regularly have far higher or lower values, and our current model's measure of variation is covering that range. But this means our model is missing out on abnormalities like traffic being 'normal for the dead of night' *during the day*. The half-hourly traffic normal distribution isn't a good enough model.

One way to improve the model is to aggregate to a different level of analysis. Rather than look at what is 'normal' for 30-minute intervals, could we not look at what is normal for days? Using the existing data divisions, construct a `train_daily` and `test_daily` that store the total passengers per day. Then use the new baseline mean and standard deviations to identify anomalous *days* in `test_daily`. If all goes well, you should find that you have recovered the date of the NYC marathon anomaly you found before, and also discovered a new one – the date of a particularly ferocious NYC blizzard.

Further Exploration

There are three more labelled anomalies in the dataset which aren't visible through the methods outlined above. If you're interested, you might want to try some of the following as part of a search for them:

1. Use a plotting library to explore the data visually.

2. Look for patterns at other levels of abstraction. A given Monday's traffic might be within the bounds for 'all days', but is it within the bounds of 'all Mondays'?

3. Explore other unsupervised anomaly detection approaches, such as the Elliptic Envelope, Isolation Forest or DBSCAN.

If you think you've found the right dates for the anomalies, check with your TAs, who have the list – but be prepared to show how you got your results.

4 WRITING PRACTICE

This section isn't part of the lab itself, but an opportunity for you to practice writing essay-style answers to (week-topical) questions. You can show your answer to your TA next week, or share in your group's Teams channel to get feedback on how you're answering questions.

1. Review the advanced persistent threat entry for APT28[4]. Look at the resources provided detailing investigations into this group. What do we know about them, and what is being assumed or extrapolated? Comment on how reasonable these extrapolations are. What can we say about their motives, in particular? [≈400 words]

[4] https://attack.mitre.org/groups/G0007/