Forecasting Cyber Hacking Breaches using Time Series Analysis

G.S.C. Sirisha, V. Sai Karthika, Dr. M. Radhika Mani

Abstract: Analyzing cyber incident data sets is an important method for deepening our understanding of the evolution of the present threat situation. This is a relatively new research topic, and many studies remain to be done. In this paper, we report a statistical analysis of a breach incident data set corresponding to 15 years (2005–2019) of cyber hacking activities that include malware attacks. We show that, in contrast to the findings reported in the literature, both hacking breach incident inter-arrival times and breach sizes should be modelled by stochastic processes, rather than by distributions because they exhibit autocorrelations. Then, we propose particular stochastic process models to, respectively, fit the inter-arrival times and the breach sizes (total number of records breached). We show that these models successfully predict the breach sizes. In order to get deeper insights into the evolution of hacking breach incidents, we conduct both qualitative and quantitative trend analyses on the data set. We draw a set of cyber security insights, including that the threat of cyber hacks is indeed getting worse in terms of their frequency, but not in terms of the magnitude of their damage.

Index Terms: Breach, Time Series, ARIMA, forecast, Seasonality, Trend

I. INTRODUCTION

The strength of time series modelling is generally not used in almost all current intrusion detection and breach prediction systems. By having time series models, system administrators will be able to better plan resource allocation and system readiness to defend against malicious activities.

Time Series Analysis

Time series are one of the most common data types encountered in daily life. Financial prices, weather, home energy usage, and even weight are all examples of data that can be collected at regular intervals. Almost every data scientist will encounter time series in their daily work and learning how to model them is an important skill in the data science toolbox. One powerful yet simple method for analyzing and predicting periodic data is the additive model. The idea is straightforward: represent a time-series as a combination of patterns at different scales such as daily, weekly, seasonally, and yearly, along with an overall trend. Your energy use might rise in the summer and decrease in the winter, but have an overall decreasing trend as you increase the energy efficiency of your home. An additive model can show us both patterns/trends and make predictions based on these observations.

A time series is a sequence of numerical data points in successive order. In investing, a time series tracks the movement of the chosen data points, such as a security’s price, over a specified period of time with data points recorded at regular intervals. There is no minimum or maximum amount of time that must be included, allowing the data to be gathered in a way that provides the information being sought by the investor or analyst examining the activity.

Time series analysis can be useful to see how a given asset, security, or economic variable changes over time. Time series forecasting uses information regarding historical values and associated patterns to predict future activity.
Time-series forecasting has been widely used for prediction of cyber-attacks. Using machine-processed attack data from network telescopes and honeypots, the number of cyber-attacks over time at minute and hour intervals are predictable over the time period of up to one day. Also, cyber-attacks were modelled at different levels (attacker IP address, targeted network ports, and victim IP address). The use of different levels for attacks were extended to an early warning system by modelling multiple timeseries for attack penetration and the number of victims.

**Auto Regressive Integrated Moving Average (ARIMA)**

Cyber-attack processes exhibit long range dependence and in order to investigate such properties, 'Auto Regressive Integrated Moving Average'(ARIMA) can be used.

An autoregressive integrated moving average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity. The AR part of ARIMA indicates that the evolving variable of interest is regressed on its own lagged (i.e., prior) values. The MA part indicates that the regression error is actually a linear combination of error terms whose values occurred contemporaneously and at various times in the past. The I (for "integrated") indicates that the data values have been replaced with the difference between their values and the previous values (and this differencing process may have been performed more than once). The purpose of each of these features is to make the model fit the data as well as possible.

Non-seasonal ARIMA models are generally denoted ARIMA(p,d,q) where parameters p, d, and q are non-negative integers, p is the order (number of time lags) of the autoregressive model, d is the degree of differencing (the number of times the data have had past values subtracted), and q is the order of the moving-average model. Seasonal ARIMA models are usually denoted ARIMA(p,d,q)(P,D,Q)m, where m refers to the number of periods in each season, and the uppercase P,D,Q refer to the autoregressive, differencing, and moving average terms for the seasonal part of the ARIMA model. When two out of the three terms are zeros, the model may be referred to based on the non-zero parameter, dropping "AR", "I" or "MA" from the acronym describing the model.

For example,

- ARIMA (1,0,0) is AR (1)
- ARIMA (0,1,0) is I (1)
- ARIMA (0,0,1) is MA (1).

**II. METHODOLOGY**

**SYSTEM ARCHITECTURE**

![System Architecture Diagram](image-url)
Fig 3: System Flow chart

III. DATASET AND IMPLEMENTATION
The main objective of our project is to model and predict cyber hacking breaches. We made use of time series analysis for achieving the above-mentioned objective. In particular, in time series analysis, a popular and widely used statistical method for time series forecasting is the ARIMA. It is an acronym that stands for autoregressive integrated moving average. It is a class of model that captures a suite of different standard temporal structures in time series data. It provides a simple yet a powerful method for making simple time series forecast. ARIMA is a generalization of the simpler auto regressive moving average (ARMA) and adds the motion of integration.

DATA PREPROCESSING
Data preprocessing is a data mining technique that involves transforming raw data into an understandable format.

DATASET DESCRIPTION
BREACH DATASET
This dataset describes the following attributes.
1. datemadepublic
2. company
3. city
4. state
5. type of breach
6. type of organization
7. total records
8. description of incident
9. information source
10. source URL
11. Year of breach
12. Latitude
13. Longitude
From the above attributes, we have considered datemadepublic and total records (<30000).

PREPARING AND TRAINING THE MODEL
We are preparing the data for further processing. We have considered only those records whose type of breach is HACK. We have dropped all the NULL values and considered the records for which total records is less than 30000. We have dropped all the records for which the total records are 0. Re-sampling of data is performed. Missing values are eliminated. Furthermore, we
made use of statistical methods to decompose data into SEASONALITY and TREND. We generate all possible p,d,q values to fit into ARIMA models. We perform grid search for optimal values. different ARIMA models with different AIC(Akaike Information Criteria) are generated. It is a widely used measure of a statistical model. It basically quantifies 1) the goodness of fit, and 2) the simplicity/parsimony, of the model into a single statistic. When comparing two models, the one with the lower AIC is generally “better”.

Now, we model the most optimal parameters i.e order and seasonal_order which is obtained from the above ARIMA model with least AIC value. This will be the final required model.

IV. TESTING

TESTING OBJECTIVES

SOFTWARE TESTING is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is Defect free. It involves execution of a software component or system component to evaluate one or more properties of interest. Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools.

Software Testing has different goals and objectives. The major objectives of Software testing are as follows:

1. Finding defects which may get created by the programmer while developing the software.
2. Gaining confidence in and providing information about the level of quality.
3. To prevent defects.
4. To make sure that the end result meets the business and user requirements.
5. To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications.
6. To gain the confidence of the customers by providing them a quality product.

Software testing helps in finalizing the software application or product against business and user requirements. It is very important to have good test coverage in order to test the software application completely and make it sure that it’s performing well and as per the specifications. While determining the test coverage the test cases should be designed well with maximum possibilities of finding the errors or bugs. The test cases should be very effective. This objective can be measured by the number of defects reported per test cases. Higher the number of the defects reported the more effective are the test cases.

Once the delivery is made to the end users or the customers, they should be able to operate it without any complaints. In order to make this happen the tester should know as how the customers are going to use this product and accordingly, they should write down the test scenarios and design the test cases. This will help a lot in fulfilling all the customer’s requirements.

Software testing makes sure that the testing is being done properly and hence the system is ready for use. Good coverage means that the testing has been done to cover the various areas like functionality of the application, compatibility of the application with the OS, hardware and different types of browsers, performance testing to test the performance of the application and load testing to make sure that the system is reliable and should not crash or there should not be any blocking issues. It also determines that the application can be deployed easily to the machine and without any resistance. Hence the application is easy to install, learn and use.

TESTCASES

In our paper, we test our model with the breach data corresponding to different consecutive years as below.

The Visualization report for the records breached between 2017-2020 is as follows:

![Fig 4: Plot for data between 2017-20](image-url)
The Visualization report for the records breached between 2018-2020 is as follows:

![Plot for data between 2018-20](image)

**Fig 5: Plot for data between 2018-20**

The Visualization report for the records breached between 2019-2020 is as follows:

![Plot for data between 2019-20](image)

**Fig 6: Plot for data between 2019-20**

V. RESULTS AND DISCUSSIONS

**PREPROCESSING THE DATA**

Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data Pre-processing is a proven method of resolving such issues.

![General task of Data Preprocessing](image)

**Fig 7: General task of Data Preprocessing**

We have considered only those records whose type of breach is **HACK**. We have dropped all the NULL values and considered the records for which ‘total records’ is less than 30000.
The following figure shows data preprocessing (Visualization report):

**Fig 8: Data Preprocessing**

**USING STATISTICAL MODELS TO DECOMPOSE DATA INTO SEASONALITY AND TREND**

Trend: It is a general systematic linear or (most often) nonlinear component that changes over time and does not repeat.

Seasonality: It is a general systematic linear or (most often) nonlinear component that changes over time and does repeat.

Noise: It is a non-systematic component that is not Trend/Seasonality within the data.

The following figure shows the decomposition of data into seasonality and trend using seasonal_decompose method:

**Fig 9: Decomposition of data into Seasonality and Trend**

**GENERATING ALL POSSIBLE p,d,q VALUES TO FIT INTO ARIMA MODEL**

ARIMA: ARIMA (Autoregressive integrated moving average) is a generalization of an autoregressive moving average (ARMA) model. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting).

SEASONAL ARIMA: The seasonal ARIMA model incorporates both non-seasonal and seasonal factors in a multiplicative model. One shorthand notation for the model is

\[
\Phi(B^S) \Psi(B) (x_t - \mu) = \Theta(B^S) \Theta(B) \omega_t 
\]

with \( p = \text{non-seasonal AR order, } d = \text{non-seasonal differencing, } q = \text{non-seasonal MA order, } P = \text{seasonal AR order, } D = \text{seasonal differencing, } Q = \text{seasonal MA order, } \) and \( S = \text{time span of repeating seasonal pattern.} \)

Without differencing operations, the model could be written more formally as
The non-seasonal components are:

AR: \( \Phi(B) = 1 - \Phi_1B - \cdots - \Phi_qB^q \)

MA: \( \Theta(B) = 1 + \Theta_1B + \cdots + \Theta_qB^q \)

The seasonal components are:

Seasonal AR: \( \Phi(B^S) = 1 - \Phi_1B^S - \cdots - \Phi_qB^{qS} \)

Seasonal MA: \( \Theta(B^S) = 1 + \Theta_1B^S + \cdots + \Theta_qB^{qS} \)

The following figure shows examples of parameter combinations \((p,d,q) \text{ and } (P,D,Q_S)\) for seasonal ARIMA:

![Examples of parameter combinations for Seasonal ARIMA...](image)

**DIFFERENT ARIMA MODELS WITH CORRESPONDING AIC VALUES**

Grid search is performed for different optimal values. Different ARIMA models with different AIC values are generated.

**AIC:** AIC is an acronym for AKAIKE INFORMATION CRITERIA. It is a widely used measure of a statistical model. It basically quantifies 1) the goodness of fit, and 2) the simplicity parsimony, of the model into a single statistic. When comparing two models, the one with the lower AIC is generally “better”.

![ARIMA Models and Corresponding AIC values](image)

**ARIMA MODEL WITH LEAST AIC VALUE**

When comparing two models, the one with the lower AIC is generally “better”. The following figure clearly shows ARIMA \((1,1,1) \times (0,1,1,12)_{12}\) with least AIC value of 2900.
Fig 12: Least AIC Value

PLOTTING THE FORECAST OF ARIMA AND OBSERVED RESULTS

We model the most optimal parameters i.e., order and seasonal_order which is obtained from the above ARIMA model with least AIC value. Prediction of data records from 2017 to 2020 is done and compared with the original plot.

The following figure shows a One-step ahead forecast:

---

EVALUATING THE MODEL USING METRICS MAE, RMSE, MSE

The following performance metrics are used to evaluate this model. They are:

1. **MAE**: Mean Absolute Error (MAE) is a measure of errors between paired observations expressing the same phenomenon.
2. **RMSE**: Root Mean Squared Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are.
3. **MSE**: The Mean Squared Error (MSE) or Mean Squared Deviation (MSD) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors.
4. **Error Percentage**: Percent Error is a special case of the percentage form of relative change calculated from the absolute change between the experimental (measured) and theoretical (accepted) values, and dividing by the theoretical (accepted) value.

The formulae for evaluating above metrics are as follows:

\[
MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|
\]

\[
MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2
\]

\[
RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}
\]

\[
R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}
\]
\[ E = \frac{MAE}{100} \]

Where,
- \( \hat{y} \) - predicted value of \( y \)
- \( \bar{y} \) - mean value of \( y \)
- \( E \) - Error percentage

The following figure shows the values of MSE, RMSE, MAE and Error percentage for our model:

![Fig 13: MSE, RMSE, MAE and Error Percentage values for our model](image1)

VI. CONCLUSION

We analyzed a hacking breach dataset from the points of view of the incidents inter-arrival time and the breach size. The statistical models (ARIMA) developed in our project show satisfactory fitting and prediction accuracies for data records having <30000 total records. Statistical tests show that the methodology proposed in our paper is better than those which are presented in the literature. This methodology can be adapted to analyze datasets of a similar nature. One can perform better data cleaning operations and even extend the model to make it work on more than 30000 breached records.

There are many open problems that are left for future research. For example, it is both interesting and challenging to investigate how to predict the extremely large values and how to deal with missing data (i.e., breach incidents that are not reported). It is also worthwhile to estimate the exact occurring times of breach incidents. Finally, more research needs to be conducted towards understanding the predictability of breach incidents (i.e., the upper bound of prediction accuracy).

REFERENCES