**Security Fundamentals to assess Application and Network Security**

Semester Project Report : Monsoon-2017

Faculty : Prof. Shatrunjay Rawat.

International Institute of Information Technology-Hyderabad

Hemal N. Thakkar

Roll No. 201350890

hemal.thakkar@students.iiit.ac.in

Anil Kumar

Roll No.

anilkumar.n@students.iiit.ac.in

# **ABSTRACT**

Technology is everywhere and it's growing so rapidly. Unfortunately, despite the advantages that all of this new technology offers, it also comes with risk. The cybercrime victims per year is over 566 million victims. Equals out to be 1.5 million victims per day or 18 victims per second. The cost of cybercrime includes not only the effects to businesses, but also to hundreds of millions of people globally. By end of 2017 the global cyber security market is expected to go from $64 billion in 2011 to $120 billion. To prevent such cybercrime on our application/network, we need to think like an attacker/hacker and assess our own security before someone else finds and exploits its vulnerabilities.

# **Introduction**

Security Assessment is not a one-time process, but an on-going process. As rightly quoted “The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards.”

For every security fix, as technology advances and with time sooner or later some vulnerabilities will be exposed which can be exploited. The Idea is to ourselves detect such vulnerabilities before anyone else does and exploits it.

A penetration test, is an attempt to evaluate the security of an IT infrastructure by safely trying to exploit vulnerabilities. These vulnerabilities may exist in operating systems, services and application flaws, improper configurations or risky end-user behavior. Such assessments are also useful in validating the efficacy of defensive mechanisms, as well as, end-user adherence to security policies.

Penetration tests are typically performed using manual or automated technologies to systematically compromise servers, endpoints, web applications, wireless networks, network devices, mobile devices and other potential points of exposure. Once vulnerabilities have been successfully exploited on a particular system, testers may attempt to use the compromised

System to launch subsequent exploits at other internal resources – specifically by trying to incrementally achieve higher levels

Of security clearance and deeper access to electronic assets and information via privilege escalation.

# **Objective**

As a part of this project we intent to study/document the top security vulnerabilities of 2017. Learn the required tools and with approval carry out Penetration scans to detect vulnerabilities and suggest corrections wherever applicable.

# **Keywords**

SQL Injection, Denial of Service, Distributed Denial of Service, Cryptography, Browser Security Headers, Buffer Overflow, Hacking Wireless network, Hacking Web Servers, Security Headers, Evading IDS, Firewalls, Honeypots, Session Hijacking, Malware Threats, System Hacking, Network Scanning.

# **Some Facts:**

* Around 230,000 malware being sampled every day.
* Estimated cost of cybercrime globally is $100 billion/year
* The most expensive computer virus of all time “MyDoom” have caused an estimated financial damage of $38.5 billion.
* Social Media – a hacker’s favorite target. More than 600,000 Facebook accounts are compromised every single day.
* There is a [Real Map](http://hp.ipviking.com/) that shows live cyber-attack.
* Social Engineering, in the context of information security, refers to psychological manipulation of people into performing actions or divulging confidential information, is cyber criminal’s favorite.
* Oracle Java, Abode Reader which almost 99% computer has can be exploited.

# **Browser Security headers**

Temporal data can be broadly classified as stream data, time-series data and sequence data.

Stream data are those that are temporally ordered, fast changing, and potentially infinite. Examples of stream data can be data generated by a satellite-mounted remote sensor. Other examples include telecommunications data, transaction data from the retail industry, and data from electric power grids.

A sequence dataset consists of sequences of ordered elements or events, recorded with or without a concrete notion of time. Sequential pattern mining is the discovery of frequently occurring ordered events or sub-sequences as patterns. An example of a sequential pattern is “*Customers who buy a Canon digital camera are likely to buy an HP color printer within a month.*”

A time-series database consists of sequences of values or events obtained over repeated measurements of time. Applications involving time-series data include economic and sales forecasting, utility studies, and the observation of natural phenomena (such as atmosphere, temperature, and wind). A time-series database can thus be of two types,

* It can be similar to a stream data with multiple, possibly infinite, online stream data inputs coming in. An example can be the seismology data for different regions.
* Or it can be a dataset containing multiple finite or eventually finite time series entries. An example can be the ball by ball data of a cricket match. Each cricket match can be considered as a time series data and the database will be considering data of multiple cricket matches.

Hence, we will first review the challenges and algorithms available for stream data, and then the case of multiple finite time series dataset.

# **STREAM DATA**

Traditional OLAP and data mining methods typically require multiple scans of the data and are therefore infeasible for stream data applications because of the following reasons.

* Streams typically have massive volume, and it is often not possible to store the data explicitly on disk. Therefore, the data needs to be processed in a single pass, in which all the summary information required for the clustering process needs to be stored and maintained. The time needed to process each record must be small and constant. Otherwise, the model construction process would never be able to catch up with the stream.
* The patterns in the data stream may continuously evolve over time. From a stream mining perspective, this implies that the underlying cluster models need to be continuously updated. A usable model must be available at any time, because the end of stream computation may never be reached, and an analyst may require results at any point in time.
* Different domains of data may pose different challenges to data stream clustering. For example, in a massive domain of discrete attributes, it may not be possible to store summary representations of the clusters effectively without increasing the computational complexity of the problem significantly. Therefore, space-efficient methods need to be designed for massive domain clustering of data streams.

However, there are some clustering algorithm specifically designed for working with stream data. They can be subdivided as below based on the methods they use.

## Methods based on Partitioning Representatives

In these techniques, the clusters are defined by a set of data points, which are drawn either directly from the data set or otherwise. The cluster membership of the remaining points are defined by assigning them to their closest representative

### The STREAM Algorithm

The core idea is to break the stream into chunks, each of which is of manageable size and fits into main memory. Thus, for the original data stream D, we divide it into chunks D1 ...Dr ..., each of which contains at most m data points. The value of m is defined on the basis of a pre-defined memory budget. This Algorithm is not particularly sensitive to evolution in the underlying data stream

### CluStream: The Micro-clustering Framework

To provide such a flexibility in computing clusters over different kinds of time horizons the process is separated out into an online micro-clustering component and an offline macro-clustering component. The online microclustering component requires a very efficient process for storage of appropriate summary statistics in a fast data stream. The offline component uses these summary statistics in conjunction with other user input in order to provide the user with a quick understanding of the clusters whenever required. CluStream clustering algorithms are incompetent to find clusters of arbitrary shapes and cannot handle outliers

## Density based Stream Clustering

The DenStream algorithm approach combines micro-clustering with a density-estimation process for effective clustering. The algorithm uses an online component which maps each input data record into a grid and an offline component which computes the grid density and clusters the grids based on the density.

# **FINITE TIME-SERIES DATA SET**

The problem of clustering of time-series data is formally defined as follows:

“***Definition****: Given a dataset of n time-series data D = , the process of unsupervised partitioning of D into C =, in such a way that homogenous time-series are grouped together based on a certain similarity measure, is called time-series clustering. Then, Ci is called a cluster, where and for .”*

Clustering of time-series data can be classified into three categories, “whole time-series clustering”, “subsequence clustering” and “time point clustering”. Whole time-series clustering is considered as clustering of a set of individual time-series with respect to their similarity. Here, clustering means applying clustering on discrete objects, where objects are time-series. Subsequence clustering means clustering on a set of sub-sequences of a time-series that are extracted via a sliding window, that is, clustering of segments from a single long time-series. Time point clustering is clustering of time points based on a combination of their temporal proximity of time points and the similarity of the corresponding values. This approach is similar to time-series segmentation. However, it is different from segmentation as all points do not need to be assigned to clusters, i.e., some of them are considered as noise.

|  |  |  |
| --- | --- | --- |
| Representation Method | Complexity | Comments |
| Discrete Fourier Transform (DFT) | O(n(log(n)) | Usage: Natural Signals  Pros: No false dismissals.  Cons: Not support time warped queries |
| Discrete Wavelet Transform (DWT) | O(n) | Usage: stationary signals  Pros: Better results than DFT  Cons: Not stable results, Signals must have a length |
| Singular Value Decomposition (SVD) | O(Mn2) | Usage: text processing community  Pros: underlying structure of the data |
| Piecewise Linear Approximation (PLA) | O(n log n) complexity for “bottom up” algorithm | Usage: natural signals, biomedical  Cons: Not (currently) indexable, very expensive O(n2N) |
| Adaptive Piecewise Constant Approximation (APCA) | O(n) | Pros: Very efficient  Cons: complex implementation |

Clustering of a finite time-series data set will fall in the category of whole time-series clustering. Reviewing existing works in the literature, it is implied that essentially time-series clustering has four components: dimensionality reduction or representation method, distance measurement, clustering algorithm, prototype definition, and evaluation. Usually, data is approximated using a representation method in such a way that can fit in memory. Afterwards, a clustering algorithm is applied on data by using a distance measure. We will go through them briefly.

## Time series representation

Time series representation can be formally defined as below.

“Definition: Given a time-series data , representation is transforming the time-series to another dimensionality reduced vector , where and if two series are similar in the original space, then their representations should be similar in the transformation space too.”

Choosing an appropriate data representation method can be considered as the key component which effects the efficiency and accuracy of the solution. Table 1 above shows various time series representation methods and their time complexity.

|  |  |
| --- | --- |
| Distance Measure | Method |
| Dynamic Time Warping (DTW) | Shape – based |
| Euclidean distance (ED) | Shape – based |
| Piecewise probabilistic | Compression based dissimilarity |
| Hidden Markov models (HMM) | Model based |
| Longest Common Sub-sequence (LCSS) | Shape-based |
| Short time-series distance (STS) | Feature based |
| Spatial Assembling Distance (SpADe) | Model based |
| Compression-based dissimilarity measure (CDM) | Compression based dissimilarity |

Table 2: Distance Measures and their type.

## Similarity measurement

In traditional clustering, distance between static objects is exactly match based, but in time-series clustering, distance is calculated approximately. In particular, in order to compare time-series with irregular sampling intervals and length, it is of great significance to adequately determine the similarity of time-series.

There are different distance measures designed for specifying similarity between time-series. The Hausdorff distance, modified Hausdorff (MODH), HMM-based distance, Dynamic Time Warping (DTW), Euclidean distance, Euclidean distance in a PCA subspace, and Longest Common Sub-Sequence (LCSS) are the most popular distance measurement methods that are used for time-series data.

The choice of a proper distance approach depends on the characteristic of time-series, length of time-series, representation method, and of course on the objective of clustering time-series to a high extent, i.e. whether we need to find similarity in time, shape or similarity in change. A shape based similarity is normally used if the length of time-series is small, where as a model based similarity is used for a long time-series and a compression based similarity is suitable for both short and long time series. Table 2 below shows different distance measures and their type.

## Clustering Algorithm

Once the representation and similarity measurement is decided, a classical clustering algorithm can be used directly or with slight modifications to cluster the time series together. Generally classical clustering can be broadly classified into six groups: Partitioning, Hierarchical, Grid-based, Model-based, Density-based clustering and Multi-step clustering algorithms. However, some class of algorithms are preferred over other based on the type of time-series data, its representation and similarity measurement chosen, and the objective of clustering.

For example, partition based clustering algorithms like k-means or k-mediods needs the number of clusters to be formed as input. But it will be impossible to predict the number of clusters owing to the large dataset and will be get natural clusters. On the other hand, these algorithms are generally very fast compared to say hierarchical clustering algorithms and hence preferred in many implementations. Often, to increase the accuracy and at the same time keeping the time complexity in check, hybrid algorithms combining algorithms of multiple groups are also used.

# **Time series forecasting**

Time series forecasting, or time series prediction, takes an existing series of data and forecasts data values. The goal is to observe or model the existing data series to enable future unknown values to be forecasted accurately. Time series forecasting are being done by either linear prediction or non-linear prediction. In linear prediction, there exists a large number of linear models, such as Auto Regression (AR), Autoregressive Moving Average (ARMA), Autoregressive Integrated Moving Average (ARIMA) etc. ARMA and ARIMA are the popular approaches widely used in many time series analysis.

Forecasted value using an Autoregressive Moving Average model can be written as below.

where

- parameters of the AR model.

- amplitude of signal.

- white noises.

But linear regression models fail to accurately forecast the value in case if there is a non-linear change in the data. In non-linear prediction, the most popularly used approach is the use of Artificial Neural Networks (ANN). However, approaches based on neural networks are very complex to build and takes a huge amount of training time.

# **PREDICTING A CHANGE IN CLUSTER**

In case of STREAM Algorithm since each chunk fits in main memory, a variety of more complex clustering algorithms can be used for each chunk. Like from each of the chunks *K* representative such that each point in that chunk is assigned to its closest representative. The goal is to pick the representatives in such a way, so as to minimize the Sum of Squared Error (SSQ) of the assigned data points from these representatives. The incoming Stream data can be related to these representative points to conclude whether this new point is a part of that chunk.

In case of CluStream the micro-clustering framework is designed to capture summary information about the data stream, in order to facilitate clustering and analysis over different time horizons. The micro-clusters are stored at snapshots in time which follow a pyramidal pattern. This pattern provides an effective trade-off between the storage requirements and the ability to recall summary statistics from different time horizons. Consider for example, the case when the current clock time is tc and the user wishes to find clusters in the stream based on a history of length h. Then, the macro-clustering algorithm will use some of the additive properties of the micro-clusters stored at snapshots tc and (tc −h) in order to find the higher level clusters in time horizon of length h.

For Density based Clustering, dense regions of smaller granularity are defined in the form of core micro-clusters. A core micro-cluster is defined as a set of data points with weight at least µ and for which the radius of the micro-cluster about its center is less than ε. Similarly we have concept of potential core micro-cluster and outlier micro-cluster. In the former case, the micro-cluster contains a weight of at least β · µ (for some β ∈ (0,1)), and in the latter case, it contains a weight less than β · µ.

Representative points in the potential-micro-clusters are found and the Stream data can be related to these representative point.

If the first step fails than try to relate it with similarly selected representative points from outlier-micro-cluster to conclude if the new point coincides with the existing cluster or not.

For Time Series, if the dataset, , is a finite time-series dataset of time-series data, at any time , we can classify the dataset into clusters. Then using linear or non-linear time series forecasting algorithms, we should be able to forecast the time series values for any time , say with an error bound of .

From the list of clusters and the data points in each cluster, we will also be able to determine the size of the cluster, and the distance between each clusters, in the same similarity measure, where

is the size of cluster in

is the distance between clusters and in

Now, on getting the actual values of the time series at any time , say , intuitively, we should be able to make a model M in such a way that can predict with an error bound whether the new values obtained at time can result in a change of cluster.

# **EXPERIMENTS**

We have taken the earth quake feed for the past 30 days from <https://earthquake.usgs.gov/earthquakes/feed/v1.0/csv.php> and are trying to perform clustering on them using tools available in R programming.

Similarly, for the finite time-series dataset, we are using the “Cricket Statistics, 2013” dataset from <http://knoema.com> and trying to cluster various players based on the indicators available like matches played, runs scores, batting average etc. Initially we will try to cluster for all players till a particular year, say 2010. Then we are trying to use this cluster information and the actual and forecasted values for 2011 to determine if any player will be changed from one cluster to another or if any new cluster can be formed.

Besides this we are using time-series forecasting algorithm provided in WEKA and work is in progress.

# **CONCLUSION**

From the readings and analysis and trials done, we believe that we will be able to make a model that can predict a change in cluster for the incoming data of the Data Stream. But we were not able to implement because of our limited knowledge and expertise in using the mining tools and the lack of time.

# **REFERENCES**

1. C. Antunes, A.L. Oliveira, Temporal data mining: an overview, in: KDD Workshop on Temporal Data Mining, 2001, pp. 1–13.
2. Jain, R. Dubes. Algorithms for Clustering Data, Prentice Hall, New Jersey, 1998.
3. L. Kaufman, and P. Rousseuw. Finding Groups in Data- An Introduction to Cluster Analysis. Wiley Series in Probability and Math. Sciences, 1990.
4. Aggarwal. On Change Diagnosis in Evolving Data Streams. In IEEE TKDE, 17(5), 2005.
5. Charu C. Aggarwal, Chandan K. Redd. Data Clustering: Algorithms and Applications, Chapman and Hall / CRC Press, 2014, ch. 10.
6. Charu C. Aggarwal, Jiawei Han, Jianyong Wang, Philip S. Yu. A Framework for Clustering Evolving Data Streams. Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003
7. C. Ratanamahatana, E. Keogh, A.J. Bagnall, S. Lonardi, A novel bit level time series representation with implications for similarity search and clustering, in: Proceedings of 9th Pacific-Asian International Conference on Knowledge Discovery and Data Mining (PAKDD’05), 2005, pp. 771–777.
8. Yixin Chen, Li Tu, Density-Based Clustering for Real-Time Stream Data
9. D. Barbar´a. Requirements for clustering data streams. SIGKDD Explorations Newsletter, 3(2):23–27, 2002.