BibTrek Prototype: A Graph Visualization Tool and Search Engine for Cybersecurity Papers

1 Introduction

1.1 Problem

As a result of the rising number of cyberattacks, it is becoming hard to find white-hat hackers that know how to thwart these kinds of attacks effectively. However, with the increased availability of information on the web it is becoming easier to study and learn how these attacks work. The cutting edge techniques used to carry them out is mostly found on the latest scientific research papers. Due to the many iterations of scientific knowledge the most recent work builds upon the work that has been done on the past. Because of this, to understand the most recent concepts it is important the reading of the referenced papers that made the writing of the current one possible. With this being said the most relevant paper of a certain subject becomes the paper with more references to it, thus making it an obligatory read. Having said that, the reality is that performing these kinds of bibliography surveys is time-consuming and inefficient.

1.2 Solution

The visualization of information using graphs is a benefit for our learning process and the construction of mental conceptual maps that makes it easier for us to retain information and relate it with everything we have learned so far on a certain topic. With BibTrek we propose a graph visualization tool and search engine with the goal of relating the many different research papers in the area of cybersecurity and presenting that in an informative easy-to-browse graph using Neo4J. The use of this graph is therefore useful to relate the papers and their references making the workflow of the survey process much more efficient but also enjoyable. BibTrek would permit us to start a search from a set of either authors, publication titles or keywords dynamically producing a reference graph of with the most relevant scientific papers therefore permitting us an easy navigation and understanding of the relationships between every of its nodes.

1.3 Overview

In BibTrek to obtain the required information to populate our graph database we make use of the APIs provided to us by the major computer science libraries that host scientific research publications. After researching what the appropriate libraries were given their terms of service and API functionalities we have chosen as our means of obtaining information the libraries of: ArXiV, DBLP and IEEEXplore. Our app is presented through a simple command line Java program that through the use of user provided queries, crawls the libraries mentioned above using their APIs and adds to the Neo4J graph database all the available hosted information about either publications, institutions, authors and subjects. This graph database can also later be queried through the use of console commands to infer useful information about the state of the system. A simple example execution would follow these steps. Let us say we want to learn about the Spectre attack. We would choose the option to query one of the mentioned above libraries, choose the option to search for a publication and then type the word "Spectre" on the console terminal. After choosing the Spectre publication (written by), the Spectre paper its references, authors and its subject will be added to the database and related with the other papers that were previously

added by the user. After the fact, the user can simply watch the graph visualization created by BibTrek, this way finding which papers are the most referenced and therefore understand which shall be the one that he shall tackle firstly.

2 Related Work

Some similar work has been done using both graph databases and their references. This work is listed here:

- **Cross Ref** is a platform on which the registered publishers are given the option to store and relate their publication references with other available database publications. BibTrek stores publications as nodes in the database and relates every available node through their references. While Cross Ref does not visually represent its data, BibTrek does so using a graph database, also, differently to Cross Ref, is given to us the option to add user input publications that have been dynamically fetched from different libraries.
- **Book recommendation using Neo4j graph database in BibTeX book metadata** is an application that using several books' BibTeX metadata and a Neo4J graph representation, through the use of user input Cypher queries, about either the author's name or book type, recommends us a book. Similarly to BibTrek, a Neo4J graph representation is used after parsing a book's metadata. The main difference being that BibTrek is focused around authors, keywords, publications and subjects whilst this application being focused exclusively on books. Finally, similarly to this application, BibTrek's Cypher queries are done to fetch the most referenced paper given a single author, keyword or publication title, making the scope different.
- **VIS - Google Scholar Citation Visualisation** is an application that is used to create a graph based on a geographic map that has information about a publication, like for example, where it has been written, relating it, like BibTrek, through its references and citations with other publications all over the said map. While BibTrek does not use a map representation, on the other hand, unlike VIS, that only uses Google Scholar, BibTrek uses different libraries to fetch its information. Finally, BibTrek is more versatile given the fact that it does not base its graph around a single publication, but every node that has been added to the system by the user so far.

3 Solution

3.1 Design

The BibTrek application was written using the Java programming language mainly due to the fact that it is ubiquitous on modern-day computing systems. The database module uses Neo4J and requires us to have either a local or remote instance running it. The connection of the JVM to the Neo4J is made using the BOLT protocol on the network port: 7687. The JSON format was chosen as the way to retrieve the information from the DBLP database using the API due to it being lightweight and having overall great flexibility. The Neo4J database is queried using " Cypher" which is its standard query language. The application was built using the Maven tool and uses the Neo4J driver and JSON maven repositories. Finally, we use the java.net package in order to establish an HTTP connection with the DBLP API.

3.2 Implementation

At the present time, BibTrek only uses the information obtained from querying DBLP API to populate our server. In the current implementation after choosing one of the available options to query the database, the user inputs either an author's name or publication title that is scanned and properly parsed into a query. Queries are done through the composition of a specifically formatted URL link and an HTTP request. After the query is successfully processed by the DBLP database the JSON response is properly parsed and displayed

in the terminal console. After choosing whichever papers are appropriate for its research the parsed JSON response is transformed into a Cypher query and written into a file. This file will later be used to create nodes and the relationships between them in our Neo4J database. This file is processed by a thread that periodically searches the current folder for newly written update files and is responsible for storing this information onto the graph database. Once the thread has finished its update responsibilities these files are then moved to a different folder in order not to be re-written into the database. An example execution of a query would be the following. The user starts by choosing the option to query the libraries for a publication by its title. Let us say he chose to query the system for publications about the Spectre attack. Because of this a query is properly formatted in the following way: "https://dblp.org/search/publ/api/?q=Spectreformat=json". The "q=Spectre" parameter in the URL link means that we are querying the system for Spectre subject; "search/publ/api/" means we will be using its API and search for a publication; "format=json" means we want the fetched data in the JSON format. After obtaining the HTTP response and parsing its JSON the information is then displayed to the user. Here every available publication about the Spectre attack is shown on the console in order for the user to choose one or more of them adding them to the Neo4J database. Afterwards the chosen information is then parsed into a Cypher query and stored. The thread responsible with the Neo4J communication and data storage will search the BibTrek file system for newly added files. If a new file is found the most recent data is then stored in the Neo4J database. The Neo4J Java thread moves the most recently written file to a different folder than the one mentioned before. Finally, the constructed graph can then be observed using the Neo4J browser by the user and infer useful information about the Spectre attack its authors and other related papers.

4 Evaluation

4.1 Tests

We have tested our application for a vast array of author and publication queries whilst locally running Neo4J using a Lenovo T400 computer with the following specs:

- **OS:** 64-bit Ubuntu 18.04.2 LTS GNU/Linux
- **CPU:** Intel Core 2 Duo P8700 @ 2.534GHz
- **GPU:** Intel Mobile 4 Series Chipset
- **RAM:** 4096MB @ 1066Mhz

BibTrek successfully processed and inserted all nodes and relations into the Neo4J database. We were able to search for any added data with ease. However, the data retrieved from the DBLP database might present a variable number of field attributes. Since we are following a fixed attribute structure for our node data, some of these fields are ignored. Future versions should consider adding these fields dynamically, we further these considerations on the Future Work section of this article.

4.2 Functionalities

At the current moment we can choose only to retrieve data from the DBLP publication. Nonetheless, future versions will include an option to query the ArXiV library, IEEE Xplore and any combination of these. Choosing the DBLP option we are able to pick whichever option suits our interests the most. At the current time we are able to query by author's name, author's publications and publication title. Once queried, BibTrek presents us the data that it has been able to retrieve. The user is then given the option to mark whichever of the retrieved information will be stored in the Neo4J graph database. The marked options are then stored on the graph with their appropriate relations. It is possible at any given moment to abort any of the above mentioned procedures.

5 Future Work

The reader of this article has to keep in mind that this project was developed in one month throughout the writer's internship with INESC-ID. Because of this there are many aspects of the current BibTrek implementation that ought to be further engineered and developed. To begin with, one of the obvious aspects to further develop would be to fetch information from more than one web based library. At the current moment only the DBLP is being used. Finally, while the ArXiV API is free to use, IEEE Xplore requires the use of a key attributed to its developers. Another aspect that should be considered is the use of dynamic attributes since, at the moment, any block of data that is fetched has some of its attributes unused because, we adhere to a fixed attribute standard when adding data to the database. At the current moment the only nodes being added to the database are either Authors or Publications. We should consider adding Companies, Foundations, Organizations, Universities and other Institutions that authors are or have been linked with in the past, Keywords/Subjects and any other relevant information for the reader. It is worth being said that nodes being created ought to be the most common set of information between the data that is being stored in the different databases that BibTrek will be using. Finally, along with more diverse nodes, each one of them should have a vast array of arguments in order to better categorize the information being added. Furthermore, at the current moment BibTrek allows the user to add duplicated information. Also, to infer better relations between the nodes already stored in the database we should use their URL attribute to download the publication's PDF and use a text reading tool such as iText in order to read its references and through them relate which node references each node. This way we can have a richer graph and understand which of the papers is the most important building block to learn about a certain subject. Obviously a check should be added in the Java code that compares if the information that is trying to be added has been put there already. Finally, we are not able to query our Neo4J graph database through the terminal console. The work to be done regarding this topic is developing the option of querying the database through a user input choice from a vast array of specific functions that manipulate the information present there, displaying relevant information about the current status of the data to the user we the intent on aiding its research process. An example of a query would be getting the most referenced paper by all the papers presently on the database with respect to a certain subject or given the current paper title that references one of the said papers.

6 Conclusion

The BibTrek prototype the way it was implemented allows us to fully query the DBLP database and acquire diverse information about the subject matter. Currently it presents a fully functional rich user interface allowing the user to insert all obtained information into the Neo4J graph database. We are able to construct a graph with some relevant node information and the relations between these nodes being able to infer which authors wrote each paper. This project's potential is related with the strength of us being able to use different databases to our advantage. This way we would be able to add even more data and create a very complex but informative graph. There is also some work to be done in inferring more information than the one being currently inferred about the retrieved data from DBLP adding more nodes and attributes to the graph representation. Finally, we should crawl the database publications' PDF texts in order to add a relationship between publication nodes that reference one another in the database.