Poster: Multi-source data analysis for SQL injection detection

Kevin Ross, Melody Moh, Teng-Sheng Moh
Department of Computer Science
San Jose State University
San Jose, CA, USA

Jason Yao
Datiphy Inc.
San Jose, CA, USA

***Abstract*—SQL Injection continues to be one of the most damaging security exploits in terms of personal information exposure as well monetary loss. Injection attacks are the number one vulnerability in the most recent OWASP Top 10 report, and the number of these attacks is continuing to increase. Traditional defense strategies often involve static, signature-based IPS (Intrusion Prevention Systems) rules, which are effective primarily against previously observed attacks but not against unknown, or zero-day, attacks. Most current strategies involve collection of traffic coming into the web application either from a network device or from the web application host. Other strategies collect data from the database server logs. In this project, we collect traffic from two points: at the web application host, and at a Datiphy appliance node located between the webapp host and the associated MySQL database server. We then use machine learning techniques to analyze these two datasets, and another dataset that is correlated between the two. We have been able to demonstrate in the initial results an increase in accuracy with analysis of the correlated dataset.**

Keywords–Intrusion Detection, SQL Injection, Correlation Feature Selection, Machine Learning

I. Introduction

Web attacks such as SQL Injection have been around for decades, yet they continue to be a relevant and increasingly damaging cause of exposure of personal data as well as negative financial impact to business and governmental entities. This is true in particular as old attacks are modified and evolved, and new attack vectors continue to appear [1]. Industry and security firms devote a great deal of resources to mitigation of web attacks, and many present mitigation strategies have limitations that current research is continually striving to overcome [2].

An SQL injection detection strategy that is a current topic of research involves the use of machine learning techniques [3][8]. Popular techniques in this research are decision trees, rule-based learning techniques, and neural networks. A primary advantage of these techniques is that they are capable of detecting new attacks [3]. These techniques and others rely on the availability of good data.

Much current research uses web traffic captured coming in to the web application, or uses logs from the web application and/or web server. The strategy that we are proposing uses traffic captured inbound to the web application in combination with traffic captured between the web application and the associated database server.

II. System Design and Implementation

The approach that proposed in this paper uses machine learning techniques to classify incoming traffic as either normal or malicious. The system consists of a custom enterprise chat web application with a remote MySQL server backend. Data is captured in two places: we captured both HTTP traffic between the traffic generation server and the webapp server, and the resulting MySQL traffic between the webapp server and the remote database server. The resulting data is compared and correlated for increased accuracy. Machine learning is done using the Weka Machine Learning Framework [7].

A. Architecture

The architecture, referring to Fig. 1, consists of four server nodes, which are KVM virtual machines running on an HP server with dual quad-core processors and 64G of RAM. These nodes are: a webapp server, a traffic generation server, a database server, and a Datiphy MySQL data capture node. These are discussed briefly below.

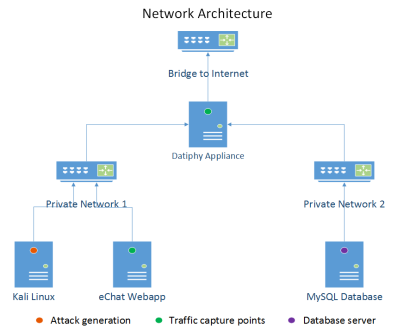

Fig. 1. Architecture

1) Web Application Server: The webapp server is running Ubuntu 14.04, with the custom web application installed in the webspace. This webapp has a MySQL backend located on the database server. This server is also running Snort for data capture and is one of the two points of data capture.

2) Traffic Generation Server: This server is used to generate both normal and malicious traffic, and is running the Kali Linux distribution. The normal and malicious traffic is generated with Python/shell scripts using the Beautiful Soup Python libraries [4]. This traffic consists of writes to the custom webapp, and these are done via HTTP POST methods.

3) Database Server: This server is also running Ubuntu 14.04 server version, and the database used is MySQL. It is set up for remote access to the database from the chat application on the webapp server, and all MySQL traffic for the webapp occurs between these two servers.

4) Datiphy MySQL Data Capture Server: This server consists of a Datiphy appliance VM provided for research by Datiphy Inc. [5]. This appliance allows for visibility of SQL traffic, among other types of database traffic. Traffic between the webapp and the MySQL database server is routed through the Datiphy appliance, thus allowing for visibility of all traffic in the Datiphy web interface. Once traffic has been generated, a report is created and used as the basis for what we call the Datiphy traffic in the following sections.

B. Process of Data Generation

The data generation process consists of three phases: traffic generation, capture, and processing. These are briefly discussed below.

1) Traffic Generation: The simulated normal and malicious traffic for our project is generated from the scripts located on the Kali Linux server as discussed previously. This traffic consists of HTTP POST requests from this server to the chat webapp, which then generates MySQL traffic between the webapp server and the database server. Normal traffic consists of simulated normal interaction with the chat web application, and malicious traffic differs in the inclusion of manually coded SQL injection attacks.

2) Traffic Capture: The traffic is captured at two points, at the webapp server, and at the Datiphy appliance. At the webapp server, we are capturing traffic using the Snort IDS tool [6]. The MySQL traffic resulting from the interaction between the webapp and the remote MySQL server is captured, as discussed, at the Datiphy appliance node. A report of this traffic is generated and saved in CSV format.

3) Data processing: Data processing for the webapp traffic consists of using TShark to process the pcap file generated by Snort; this is done by selecting the desired fields and processing these into a CSV file. The CSV file is then cleaned up with bash shell scripts for further processing. The data captured from the Datiphy appliance is also processed with shell scripts. In the final stage, these two datasets are processed into one file with shell scripts to create the correlated dataset. Once imported into Weka the numerical and nominal data is used as it is, and the string data is further processed into word vectors. In this phase we perform feature selection with the Correlation Feature Selection (CFS) algorithm.

C. Machine Learning

Our analysis of the processed data is done with Weka, which is a Machine Learning (ML) framework that includes many current ML techniques. We are currently using the J48 rule-based algorithm, the JRip rule-based algorithm; a later version will include Naive Bayes and Artificial Neural Networks (ANN) techniques. The results obtained are presented below.

III. Experiment and Results

A. Datasets

The datasets consist of normal and malicious traffic originating from the traffic generation server, and collected from both the web application server and the Datiphy appliance server. A third dataset is created from correlating and combining these two datasets. For the initial results the dataset consists of 1600 entries, 800 normal traffic and 800 malicious traffic. Certain features are unique to each of the datasets. The webapp dataset includes features of the incoming packets such as TCP and HTTP length. The Datiphy dataset includes information about the SQL statement, and also includes features related to the SQL Response and Result.

B. Preliminary Results

1) Results: Preliminary results are summarized in TABLE I.

TABLE I
Preliminary results

Dataset	Algorithm	Accuracy
Webapp	JRip	84.375%
	J48	87.188%
Datiphy	JRip	82.375%
	J48	84.938%
Correlated	JRip	98.438%
	J48	98.688%

C. Analysis

An intuitive understanding of this process would suggest that as there is more data available from the correlated dataset, as it is a combination of two datasets, the results obtained would be better in terms of classification accuracy. In our experiments so far this has been the case with JRip for example using features from both datasets in generating rules. An example rule from the correlated dataset includes both http.content_length from the webapp dataset, and SQL Statement length and Result length from the Datiphy dataset.

IV. Conclusion

SQL injection attacks, and web-based attacks in general, continue to be a major issue in the security of financial, health, and other important data. This problem only increases in importance as a growing number of societal processes become more dependent on the Internet [1]. In this project we have proposed a multi-source data analysis system for increased accuracy in detection of SQL injection attacks. Future works include adapting this system to detect other types of web-based attacks, as well as analysis of additional machine learning techniques for both accuracy and performance.

References

[1] OWASP Top 10, 2013, Top 10 2013-A1-Injection, https://www.owasp.org/index.php/Top_10_2013-A1-Injection

[2] Imperva Application Defense Center (ADC), 2015, 2015 Web Application Attack Report (WAAR), http://www.imperva.com/docs/HII_Web_Application_Attack_Report_Ed6.pdf

[3] Moh, M.; Pininti, S.; Doddapaneni, S.; Moh, T-S., Detecting Web Attacks Using Multi-Stage Log Analysis, 2016 IEEE 6th International Conference on Advanced Computing (IACC), Bhimavaram, 2016, pp. 733-738.

[4] Leonard Richardson, Sept. 29, 2015, Beautiful Soup 4.4.0, https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[5] Datiphy Data Sheet, Dec. 2016, Datiphy Inc., San Jose, CA, http://datiphy.com/resources/data-sheet/

[6] The Snort Project, Aug. 28, 2015, SNORT Users Manual 2.9.7.3, https://www.snort.org/documents/snort-users-manual

[7] Hall, M.; Eibe, F.; Holmes, G.; Pfahringer, B.; Reutmann, P.; Witten, I.; The WEKA Data Mining Software: An Update, SIGKKD Explorations, Volume 11, Issue 1, 2009

[8] Tan, Pang-ning, et al., Introduction to Data Mining, Addison Wesley, Pearson Education Inc., 2005