Prototype application to detect malicious network traffic with case-based reasoning and SEASALT

Jakob M. Schoenborn[1,2,3,*], **Klaus-Dieter Althoff**[1,2]
[1]University of Hildesheim, Germany, schoenb@uni-hildesheim.de
[2]German Research Center for Artificial Intelligence (DFKI), Germany, klaus-dieter.althoff@dfki.de
[3]Exploit Labs GmbH, Germany, jakob@exploitlabs.de

1 Introduction

The amount of criminal online activities rises. Protective measures such as firewalls and intrusion detection systems are being actively developed. We accompany this development by offering a case-based reasoning prototype to detect similar attacks based on previous cases. The instantiation of the SEASALT framework allows us to distinguish between two different views on network traffic: the request itself, and the traffic overall. Here, the focus has been set on SQL-injections and cross site scripting - two of the most commonly used attack vectors in the last decade[1]. As we store cases containing these attacks, we are able to detect slightly similar attacks, which would be difficult to detect, for example, by a set of rules. Depending on the use-case, we identified up to 16 relevant attributes, predominantly text attributes. However, the similarity assessment needs improvement to reduce the rate of false-positives.

2 Development of a prototype

Analyzing the network traffic data requires reliable tools to monitor the ongoing network data. We gather the network data by using Wireshark[2] and Burp[3]. Both aim to capture, filter, and adjust network traffic and are standard tools to use in the domain, with slightly different foci. These tools allow us to generate structured text data (CSV and XML), containing all relevant information to automatically generate cases. Per default, seven columns are displayed: No., Time, Source, Destination, Protocol, Length, Info. Wireshark and Burp are the standard tools for network analysis across many different institutions due to their large variety of different observable protocols and continuing development by volunteer contributions. Fig. 1 illustrates the general process of the system. Every agent uses the myCBR[4] 3.0 SDK which is an open-source similarity-based retrieval tool.

[1] see the OWASP Top10: https://owasp.org/www-project-top-ten/
[2] For more information, see https://www.wireshark.org/
[3] For more information, see https://portswigger.net/burp
[4] For more information, see http://mycbr-project.org/

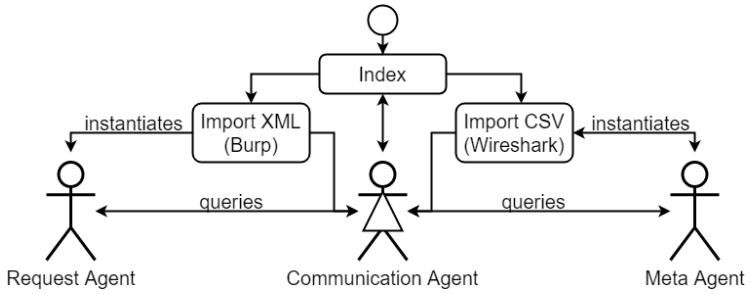

Fig. 1. Overview of the prototype. The workflow starts at index.jsp. By importing a Wireshark or Burp Export, the corresponding agent will be initialized. Afterwards, the user may send queries to the communication agent.

3 Experiment and Results

We distinguish between a *normal* and a *malicious* casebase. Initially, we fill the *normal* casebase our agents with XML/CSV training imports, which do not contain malicious traffic to establish a model of valid traffic. In a second step, we fill the *malicious* casebases with hand-crafted malicious requests and public lists of known SQLi/XSS attacks ('payloads') to establish a model of malicious traffic. By using Burp and the OWASP Juice Shop, we are located in a controlled environment without noisy data. In a third step, for training purposes, we import a CSV or a XML file and search for malicious traffic.

Our test is similar to the third step: We upload a XML file with six malicious requests out of a total of 795 requests. These malicious requests are manually created and are not part of the *malicious* casebase. We aim to automatically identify the malicious requests after comparing each of the 795 requests against cases in our *malicious* casebase. Basically, we query the CBR system automatically for each request against the *malicious* casebase which has been trained with malicious cases before. We expect to find similar cases with at least 90 % similarity. Indeed, we can identify an attack containing:

{*'email':' ' AND INSERT INTO users VALUES(1,2,3,4); –','password':'123'*}.

This attack revolves around guessing the number of columns the queued table contains by adding numbers in the round brackets after VALUES, which is a common testing approach. However, unfortunately, we only found one out of six attacks. Lowering the threshold to at least 85 % similarity sheds light onto the situation: now, we also do find more payloads which adds up to identifying four out of six attacks. Nevertheless, we also find false-positives in 11/16 cases (0,6875 %) and one redundant finding.

More information in the video!
https://www.youtube.com/watch?v=XBJYp6GG4tM