Hiram - cyber security architecture tool

Jonathan Isakov
College of tel aviv jaffa
tel aviv jaffa, Israel
jonathan.isakov@gmail.com

ABSTRACT

A major issue for cyber security architects and cyber security advisors is the design of the systems. During this process, the similarities between previous solutions aiming at the same purpose become apparent. This paper will suggest the usage of ML with the fuzzy matching between a known labeled good (supervised learning), and the solution is drawn by the "architect" to suggest the best similar architecture.

KEYWORDS

ACM proceedings, LaTeX, text tagging

1 INTRODUCTION

During the integration and design process, a cyber security architect/consultant will ask the developer or the IT administrator to draw the proposed architecture of the system they are suggesting. Later the cyber security personnel will use his best knowledge and review well-known security best practices to suggest improvements. This process ends up costing all the involved parties time, money, and flexibility in the design and development cycle. To solve this issue, an ML process that takes the suggested design and finds the best matching pre-approved design can be a helpful tool, thus, allowing the developers to receive instant feedback and free up time for the cyber security personnel. Hiram will use the Draw.IO free design tool to recreate a matrix of connections between all the different components of the system. He will suggest the best matching connection between the proposed architecture and a previously drawn architecture. As a next step, the system will try to categorize the architecture by purpose to increase the learning base further.

2 METHODOLOGY

Hiram will follow the following methodology:

(1) The data will be received as an XML saved from the DRAW.IO.
(2) The DRAW.IO saves the XML using the Deflate compression algorithm.
(3) The next step will be using the DRAW.IO provided tool to decompress this in the following URL https://jgraph.github.io/drawio-tools/tools/convert.html.
(4) The last step is converting the decompressed XML to a 2-dimensional matrix showing any connections between different components.

After finishing the normalization of the data into a format that allows ML fuzzy learning to take place.

(1) Transform the 2-dimensional matrix into a single binary word.
(2) Use the Levenshtein Distance algorithm to match the best-known word to this new word

3 POSSIBLE IMPROVEMENTS

To best categorize the new "architectures," we will use a grouping algorithm where we will divide each component into a general category and count the number of components of each category. We will place the new vectors resulting from this on an N-dimensional graph and choose the nearest distance to a pre-categorized architecture (one of the architectures used for fuzzy matching).

4 SCOPE

The scope of this work will hold seven different items being described in the XML in the shape attribute of the object. These will be:

(1) thin item client
(2) firewall
(3) xenapp server (will represent file server)
(4) cache server (will represent web server)
(5) cellphone
(6) xenclient synchronizer (will represent back end server)
(7) chassis (will represent SQL server)
(8) a direction line (from source to destination, meaning who sent the first packet, either syn in TCP or packet number 0 in QUIC/UDP).

SYSTOR'21, June 2021, Haifa, Israel

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

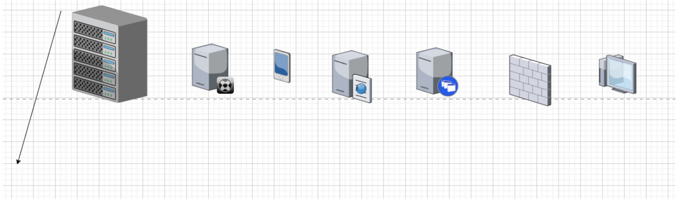

Figure 1: possible objects from right to left

With these in mind, we will mix and match about 12 different architectures to learn from each 3 of them from a different "domain," these being:

(1) web applications
(2) backup applications
(3) mobile applications
(4) A logging server (either a SIEM collector or something with the same line of thought)

5 GOALS

The project will be considered successful if, by the end of the project, it can receive any architecture from Draw.io containing the "in scope" objects and offer a better (more secure) alternative while keeping in mind the specific intention of the original design. Meaning if a web application architecture is detected, it will be with a high level of confidence the web application and the offered architecture will be able to perform the same operations. This will be tested with another three architectures from each "domain," testing its reliability.

6 RELATED WORK

In this section, we review previous research related to our study. We first provide an overview of the Literature on automatic architecture review. Finally, we summarize studies that have been investigated.

6.1 Literature on automatic architecture review

(1) Nemesis: Automated Architecture for Threat Modeling and Risk Assessment for Cloud Computing BY Patrick Kamongi, Mahadevan Gomathisankaran and Krishna Kavi 2014. speaks about checking a cloud configuration for vulnerabilities by enumerating the different components and checking each one for vulnerabilities. Sadly this doesn't solve the issue that correct architecture eliminates many risks by making the vulnerabilities un-exploitable.
(2) ARES: Automated Risk Estimation in Smart Sensor Environments BY Athanasios Dimitriadis, written in July 2020, has a similar issue. Though showing a model for risk assessment, it does check a proposed architecture doesn't offer a better architecture but provides a list of problems with the architecture drawn in a specific, not industry-well-known tool.
(3) A Fuzzy Probability Bayesian Network Approach for Dynamic Cybersecurity Risk Assessment in Industrial Control Systems by Qi Zhang and Chunjie Zhou in 2018 is an excellent intro to the power of ML with probability to examine a model. Sadly here, the examination of the model isn't automatic and is used with predefined wights that might be susceptible to a strong correlation instead of causation. A good architecture might have made the vulnerability not exploitable. Still, since it simply looks for its presence, we might assume the exposure is why the architecture is malformed.