CP-8320 Project Proposal: DeeperXSS

Dawson Brown, 500780579
Matthew Bush, 500770831

Ryerson University

1 Introduction

Cross-site scripting (XSS) attacks persist as a major issue for web applications despite their root causes being well understood. The Open Web Application Security Project consistently ranks XSS in the top ten vulnerabilities on the web. For this reason, there exists a large body of work aimed at automating the detection and prevention of XSS attacks; this includes approaches using Machine Learning techniques to detect both attacks and vulnerabilities. DeepXSS is an LSTM classifier that is meant to detect XSS payloads; it was designed by Fang *et. al* and is purported to have very high precision and recall rates. For our project, we intend to recreate and extend the work of DeepXSS by Fang *et. al* [2]. We want to more clearly outline and comment on the strengths and weaknesses of their architecture as well as replicate their purported results.[2].

2 Motivation

The primary motivation for this work is to verify Fang *et. al*'s DeepXSS method of detecting cross-site scripting (XSS) attack payloads [2]. Given that XSS is a significant problem for many web applications, the need for further research into the detection of XSS attack payloads is apparent. Unfortunately, we found that this paper lacked detail and failed to address key questions related to the work. Given DeepXSS' promising results, it would be very useful to address the lack of detail, and attempt to replicate the methods used by the authors. In verifying these results we could further our understanding of why DeepXSS was so effective (or why it was not as effective as it seemed) and apply lessons to future research.

3 Project Outcomes

To begin, we will simply recreate the architecture outlined in the original DeepXSS paper [2]. This will include:

1. creating two custom scrapers to pull malicious and legitimate URLs from two online repositories (see below)
2. creating a custom recursive decoder that can decode several common encoding schemes (including base64, hex, URL, UTF-7, etc.) and can decode multi-encoded strings

3. building a generalizer to remove string literals, numbers, and control characters (as per the paper's method).
4. building Javascript tokenizer
5. Training a CBOW model against a Corpus of Javascript payloads
6. Training an LSTM-classifier with sequences of CBOW vectors representing sequences of Javascript tokens in a payload.

We will used the same datasets that were used in the original DeepXSS paper. For malicious samples we are using the the XSSed database [4], and for legitimate samples, we are using the DMOZ database [1]. In total there were 33,426 malicious samples and 31,407 legitimate samples in the original paper. We expect to use a similar number in our study.

In order to extend on this work, we will be comparing the accuracy of the LSTM classifier using CBOW to vectorize the tokens (this is the DeepXSS architecture) with an LSTM classifier that doesn't use CBOW and instead uses nondescript token IDs instead of learned vector representations. This new LSTM classifier will also be evaluated with 10-fold cross validation. In this way we hope to show that using CBOW is, or is not, an important step in the efficacy of the classifier.

Addition to gauging the usefulness of CBOW in the DeepXSS architecture, we also want to gauge the usefulness of the LSTM. Both LSTM models above (with and without CBOW) will be compared to a traditional XSS detector that will work directly with the decoded XSS (skipping the tokenization, generalization, and training). Performing these three experiments in tandem should shed light on which parts of DeepXSS are valuable contributions and which are not.

3.1 Division of Work

Dawson will be doing steps 1-4; that is all the data collection, data sanitization, decoding, and JS tokenization.

Matthew will be responsible with the machine learning side of things–that is training the LSTM and CBOW models and performing the 10-fold cross validation for the two LSTM models.

Both of us will work to compare the deep learning methods to a conventional XSS detector that doesn't using deep learning. We will use the whitelist-list based tool Bleach (which wrapper for the html5lib Python library) [3].

References

1. Dmoz. http://dmoztools.net/, accessed: 2021-11-1
2. Fang, Y., Li, Y., Liu, L., Huang, C.: Deepxss: Cross site scripting detection based on deep learning. In: Proceedings of the 2018 International Conference on Computing and Artificial Intelligence. pp. 47–51 (2018)
3. gguthe-moz, Kahn-Greene, W.: Bleach, https://pypi.org/project/bleach/, accessed: 2021-11-3
4. Kevin, F., Pagkalos, D.: Xssed.com.xss (cross-site scripting) information and vulnerable website archive, https://www.xssed.com/archive, accessed: 2021-11-1