Motivation

- Cross-Site Scripting was the most prominent attack vector of hackers in 2019, making up nearly 40% of cyber-attacks globally.[1]
- NLP-based AEG is a new field that allows for automated exploit construction.
- This work could lead to non-cybersecurity practitioners being able to detect vulnerabilities in their own software.

Part-of-Speech Tagging

Part-of-Speech Tagging:
- We use the POS Tagging function in the spaCy[9] library in combination with regular expressions to identify URLs in intents. These URLs are marked so they can be ignored by the model which simplifies training.

Evaluating Code Generation Performance

- Esprima[10] is used to evaluate the syntax of JavaScript code in the datasets.
- Compile-Run[11] is used to evaluate the execution of JavaScript in the datasets.
- The XSS Attack Tester deploys and verifies the execution of XSS attacks in the Malicious datasets.

Approach

HIJaX is an NLP-based AEG tool that can convert English descriptions of code into JavaScript that is syntactically correct and executable.

Intent: English description of code.
Snippet: JavaScript code representation of an intent.

- PrePost processing includes Transpiling and POS Tagging.
- Snippets are given BLEU[6] and exact scores[7].
- The JavaScript Execution Tester and XSS Attack Tester calculate the syntax and execution scores for snippets.

HIXA Model

Stack Overflow Dataset:
- JavaScript-related questions and answers are mined from Stack Overflow[2] to create intent and snippet pairs.
- Benign Dataset:
  - Snippets are created from a JavaScript tutorial website[3] containing many examples of different JavaScript concepts.
  - Malicious Dataset:
  - Snippets are created with XSS payloads from a GitHub repository[4] ranging from pop-ups to attacks that steal personal data.

Transcribed Dataset:
- Intent and snippet pairs are created from transcribing the Python code in the CoNaLa[5] dataset into JavaScript.

Training Data

Transpiling

We use Transcrypt[8] to convert snippets in the CoNaLa Dataset from Python to JavaScript.

Conclusion

- We can successfully generate benign JavaScript code that compiles with high accuracy.
- We can successfully generate and deploy XSS attacks on penetration test websites.
- HIJaX requires datasets of at least 2,000 entries for accurate code generation.
- POS tagging website URLs increases the accuracy of code generation in smaller datasets.

Future Work

- Automating the deployment of XSS attacks and vulnerability patches.
- Generating XSS attacks for web-based mobile apps.
- Creating real-world datasets from social media posts, published CVEs, and online forums.

References


Work supported in part by National Science Foundation Grant No. 1566321

Yaw Frempong, Yates Snyder, Erfan Al-Hossami, Meera Sridhar, and Samira Shaikh
(ymfrempon, ysnnyder1, eaalhossa, mmsridhar, sshaikh2}@uncc.edu)