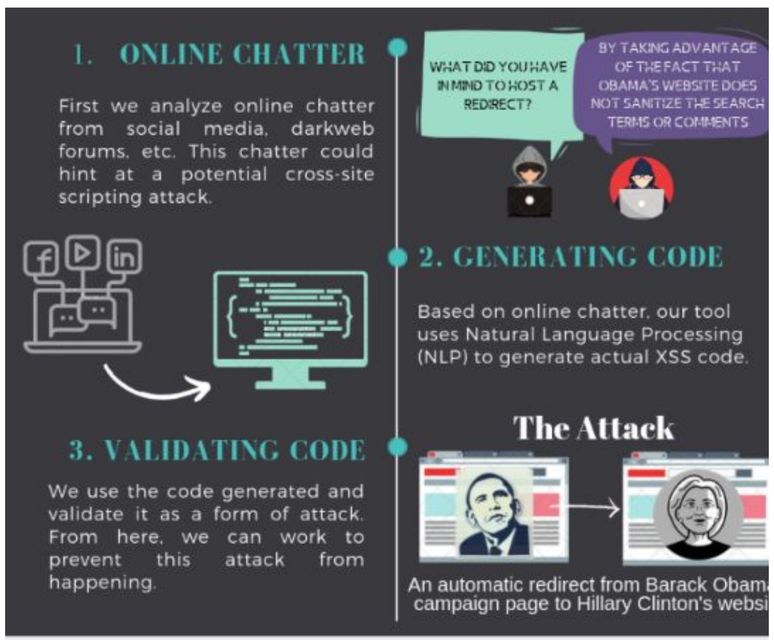
# HIJaX: Human Intent to JavaScript XSS Generator

Yaw Frempong, Yates Snyder, Erfan Al-Hossami, Meera Sridhar, and Samira Shaikh
(yfrempon, ysnyder1, ealhossa, msridhar, sshaikh2)@uncc.edu

## 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 allow for automated exploit construction.
- This work could lead to non-cybersecurity practitioners being able to detect vulnerabilities in their own software.

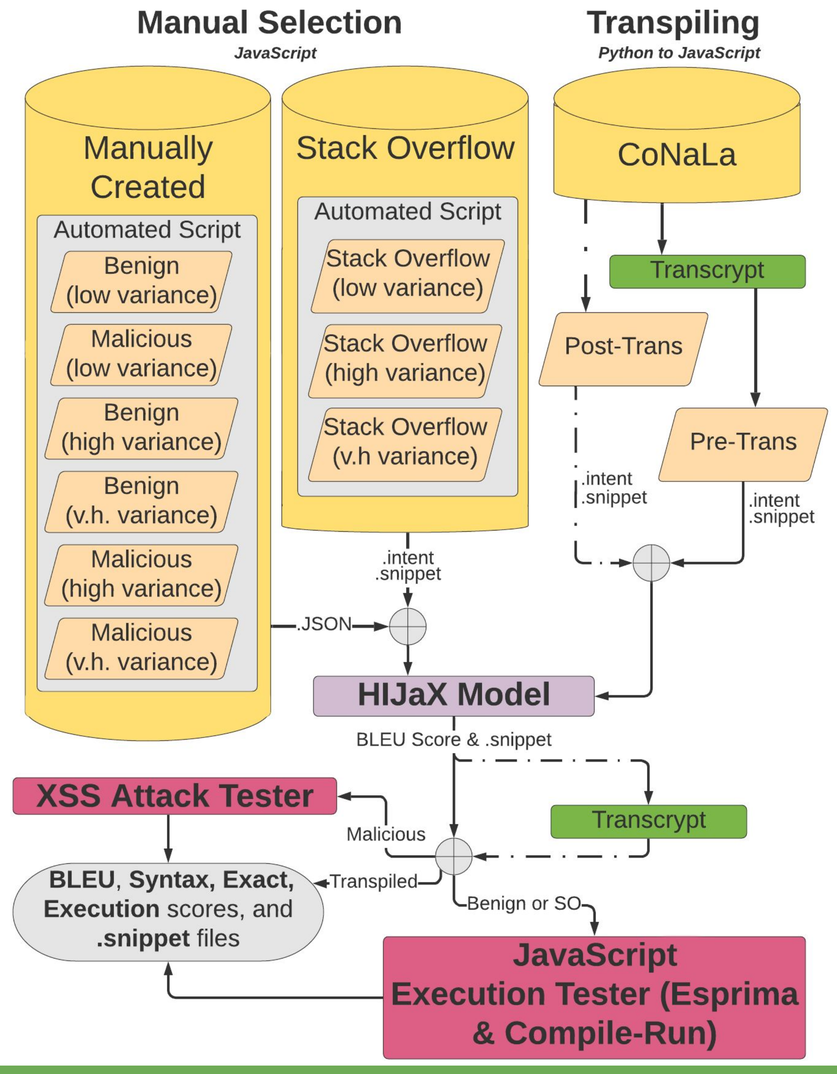



## Approach

**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.

- **Pre/Post processing** includes **Transpiling** and **POS Tagging**.
- **Snippets** are given **BLEU**[6] and **exact scores**[7].
- The **JavaScript Execution Tester** and **XSS Attack Tester** calculates the **syntax** and **execution scores** of snippets.




## Training Data

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.

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

## Transpiling

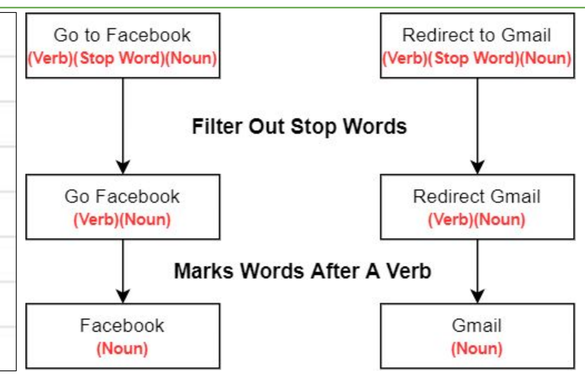
We use Transcrypt[8] to convert snippets in the CoNaLa Dataset from Python to JavaScript.

```
{
  "question_id": 36875258,
  "intent": "copying one file's contents to another in python",
  "rewritten_intent": "copy the content of file 'file.txt' to file 'file2.txt'",
  "snippet": "shutil.copy('file.txt', 'file2.txt')",
}
```

## Part-of-Speech Tagging

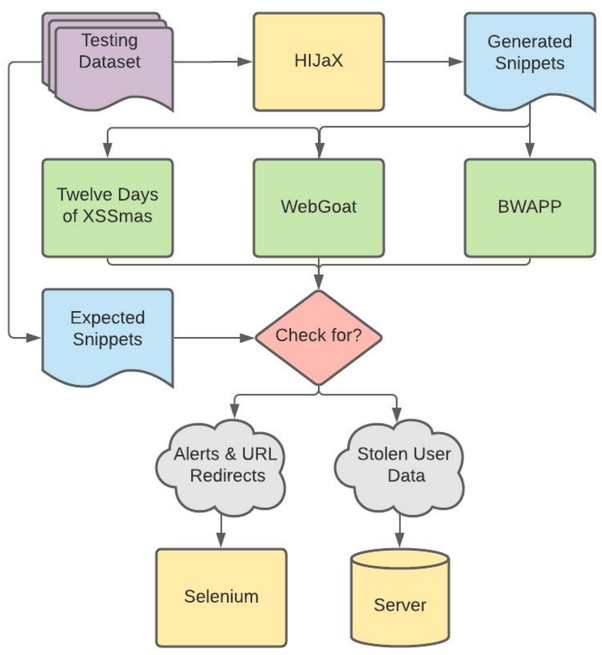
Part-of-Speech Tagging:
- We use the POS Tagging function in the spaCy[9] library in combination with regular expressions to identify URLs in intents. These URLs are marked so they can be ignored by the model which simplifies training.

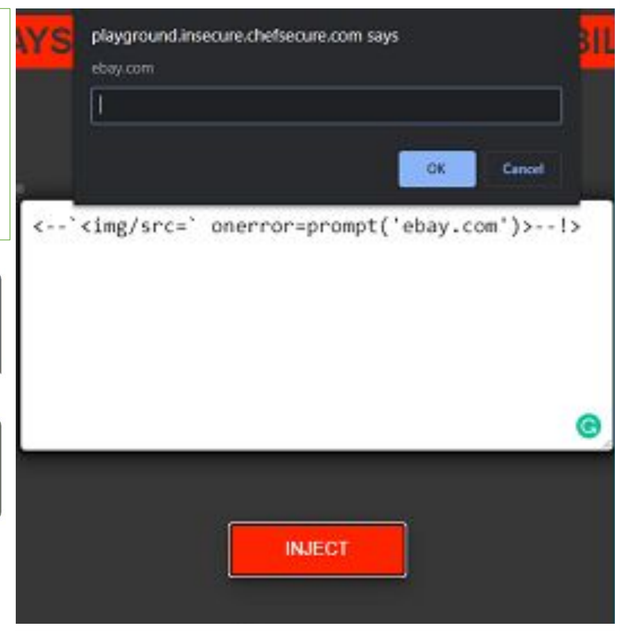
| Intent |
|---|
| redirect script to linkedin |
| redirect script to linkedin.com |
| redirect script to www.linkedin.com |
| script to redirect to linkedin |
| script to redirect to linkedin.com |
| script to redirect to www.linkedin.com |
| redirect to linkedin |

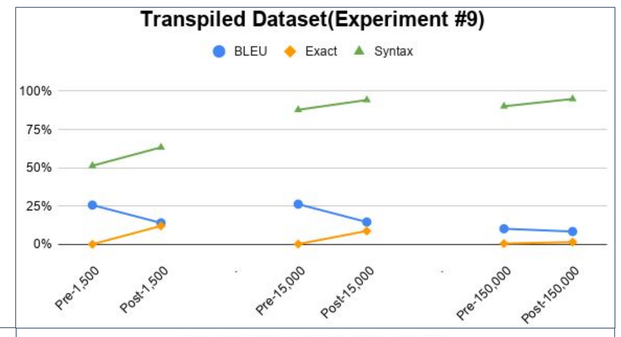


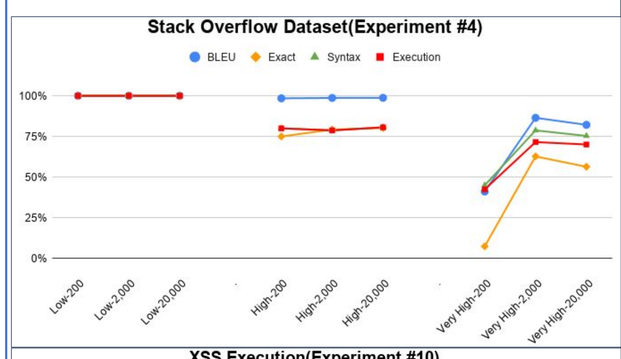

## Evaluating Code Generation Performance

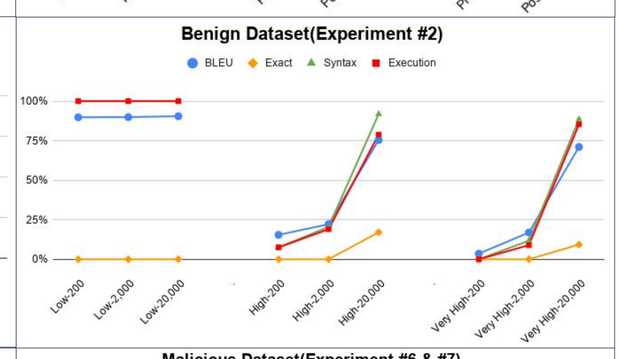
- Esprima[10] is used to evaluate the syntax of JavaScript code in the datasets.
- Compile-Run[11] is used to evaluate the execution of JavaScript in the datasets.
- The XSS Attack Tester deploys and verifies the execution of XSS attacks in the Malicious datasets.



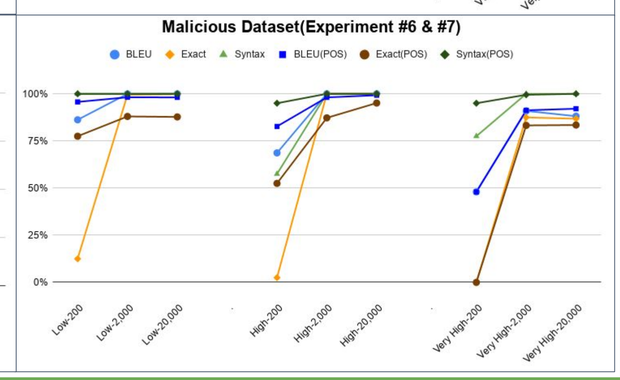








## Conclusion

- 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

[1] Jastra Ilic. 2019. Cross-Site Scripting (XSS) Makes Nearly 40% of All CyberAttacks in 2019. https://www.precisesecurity.com/articles/cross-site-scripting-xss-makes-nearly-40-of-all-cyber-attacks-in-2019/.
[2] Stack Overflow. 2020. "Stack Overflow: Where Developers Learn Share & Build Careers". https://stackoverflow.com/.
[3] HTMLCheatSheet. 2019. JS CheatSheet. https://htmlcheatsheet.com/js/. Retrieved 6-2-2020.
[4] Payload Box. 2019. Cross Site Scripting (XSS) Vulnerability Payload List. https://github.com/payloadbox/xss-payload-list. Retrieved 6-2-2020.
[5] Carnegie Mellon University. 2019. CoNaLa: The Code/Natural Language Challenge. https://conala-corpus.github.io/. Retrieved 03-10-2020.
[6] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. InProc. of the 40th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 311–318.
[7] Pengcheng Yin and Graham Neubig. 2019. Reranking for neural semantic parsing.InProc. of the 57th Annual Meeting of the Association for Computational Linguistics.4553–4559.
[8] Transcrypt. 2016. https://www.transcrypt.org/docs/html/index.html.
[9] spaCy. 2020. spaCy·Industrial-strength Natural Language Processing in Python. https://spacy.io/.
[10] Ariya Hidayat. 2020. Esprima: ECMAScript parsing infrastructure for multipurpose analysis. https://esprima.readthedocs.io/en/latest/syntactic-analysis.html.Retrieved 05-8-2020.
[11] Vibhor Agrawal. 2019. compile-run - npm. https://www.npmjs.com/package/compile-run. Retrieved 6-10-2020.

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