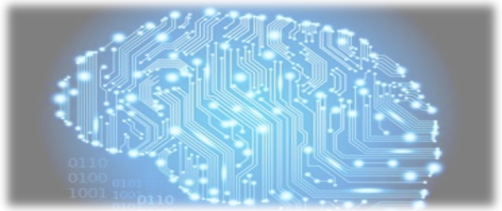
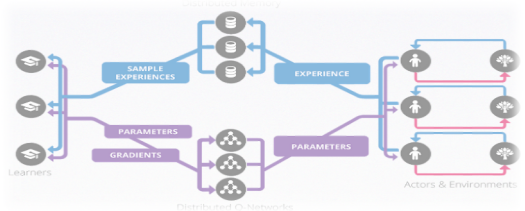

Restoration of Homoglyph Attack with BERT (2)

AutoML Lab

Hanyong Lee

2022. 2. 5.

Restoration of Homoglyph Attack with BERT

To Do List

- Last Week
  1. Defining the problem
  2. Data collection
  3. Searching prior studies
  4. Building BERT model
- This Week
  1. Related studies
  2. Setting the baseline

Restoration of Homoglyph Attack with BERT

1. Related studies

1. *Unicode Technical Standard #39,"Unicode Security Mechanisms," edited by Mark Davis and Michel Suignard. Version 14.0. 2021-08-25. (https://www.unicode.org/reports/tr39/#Confusable_Detection)*
   Data: http://www.unicode.org/Public/security/8.0.0/confusables.txt

- Recommended confusable mapping for IDN
  - **Translate table** for confusable characters
- It contains translations to non-ASCII characters.
  - e.g., (x → χ) COPTIC SMALL LETTER KHI → GREEK SMALL LETTER CHI
  - We need: (x → x) COPTIC SMALL LETTER KHI → LATIN SMALL LETTER X

Restoration of Homoglyph Attack with BERT

1. Related studies

2. *H. Suzuki, D. Chiba, Y. Yoneya, T. Mori, and S. Goto, "ShamFinder: An Automated Framework for Detecting IDN Homographs," Proceedings of the 19th ACM Internet Measurement Conference (IMC 2019), pp. 449--462, October 2019*

- Detecting IDN(Internationalized Domain Name) Homoglyph Attacks using **SimChar database**.

→ Restoring homoglyph with **translate table**.

→ Translate table can't consider the context.

∴ Vulnerability to context-sensitive cases.

- e.g., "γ (MATHEMATICAL ITALIC SMALL GAMMA)" can be translated into "**y**" or "**r**"
- "Who aγe γou?"

1. Related studies

- **SimChar database** example:


```
"ṗ": {
      "black_pixels": 25,
      "codepoint": "U+1e57",
      "decimal": 7767,
      "lang": "Latin Extended Additional",
      "name": "LATIN SMALL LETTER P WITH DOT ABOVE",
      "similar_char": [
        {
          "black_pixels": 24,
          "char": "p",
          "codepoint": "U+0440",
          "decimal": 1088,
          "delta": 1,
          "lang": "Cyrillic",
          "name": "CYRILLIC SMALL LETTER ER"
        },
                                    :
                                    :
```

1. Related studies

- **SimChar** uses following formula to calculate similarity of two bitmap images of characters.

$$\Delta = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |I_1(i,j) - I_2(i,j)|$$

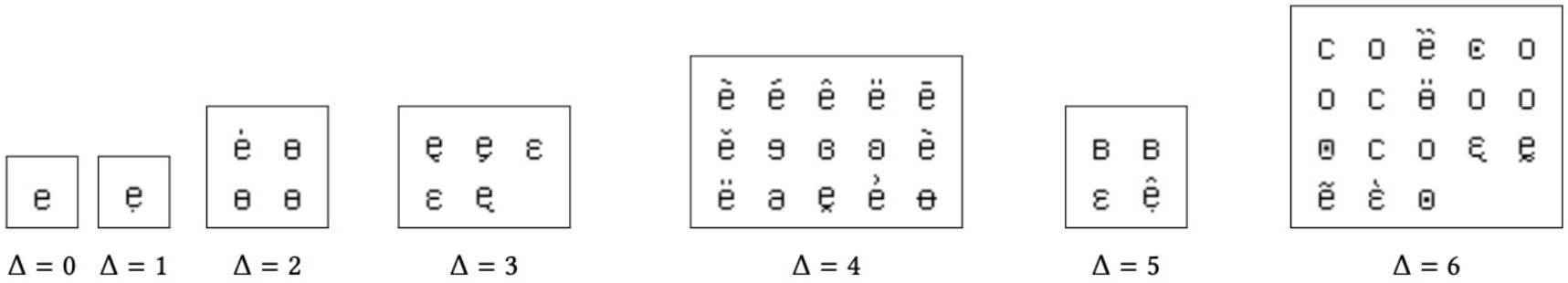

Figure 6: Basic Latin lowercase letter 'e' and characters under different values of the threshold Δ. In this work, characters with $\Delta \leq 4$ are detected as homoglyphs.

2. Limitation of related studies

1. Focusing only on IDN Homoglyph attacks.
2. Not considering context-sensitive cases.
   - e.g., "γ (MATHEMATICAL ITALIC SMALL GAMMA)" can be translated into "**y**" or "**r**"
   - "Who aγe γou?"
3. Insufficient translation list
   - e.g., The translation "**ß → B**", "**ĕ → e**", "**κ → K**" and "**ŏ → o**" don't exist in *SimChar Database* and *Unicode Confusable Characters Dataset*.

3. Setting the baseline

Baseline1: Homoglyph restoration using **SimChar database**

- Baseline Algorithm:

```
for char in original string:
    if char is non-ASCII:
        Ω ≝ A set of homoglyphs of char (similar_char[1])
        sort Ω by delta[1] in ascending order
        for ω in Ω:
            if ω is ascii and ω is alphabet:
                replace char to ω
                break
```

[1]See the SimChar database example at page 5.

- metric: accuracy
  - (# of correctly translated characters) / (length of non-ASCII characters)

3. Setting the baseline

Baseline2: Homoglyph restoration using **OCR**

- Baseline Algorithm:

```
for char in original string:
    if char is non-ASCII:
        replace char to OCR(char) output
```

- metric: accuracy
  - (# of correctly translated characters) / (length of non-ASCII characters)

3. Setting the baseline

- Making two **Translate Dictionary** files
  1. SimChar_dict.json
     - Refining from **SimChar Database** file
     - Extracting one ASCII character per one non-ASCII(homoglyph) character.

  2. OCR_dict.json
     - Python OCR API: **pytesseract**
     - Image files: **Unicode Images Database**
       - https://github.com/bbvanexttechnologies/unicode-images-database