Proceedings of the Language Technologies for All (LT4All) , pages 137–140
Paris, UNESCO Headquarters, 5-6 December, 2019.

Spoof-Vulnerable Rendering in Khmer Unicode Implementations

Joshua Horton, Ph.D., Makara Sok, Marc Durdin, Rasmey Ty
National Polytechnic Institute of Cambodia
Phnom Penh, Cambodia
joshua_horton@sil.org, makara@keyman.com, marc@keyman.com, rasmeyt2@npic.edu.kh

Abstract

While there are established conventions for typing Khmer using the Unicode Standard, existing systems provide little assistance to users in following the conventions which are thus often ignored. When typing Khmer text, users find that words can be constructed in multiple ways, all of which look 'correct' on-screen. Furthermore, some aspects of Khmer as implemented by common operating systems deviate from the Unicode Standard. This leads to a number of negative outcomes, including phishing and spoofing security risks, poor searchability and complications with natural language processing. This paper identifies issues in the encoding of the Khmer language with the Unicode Standard.

Keywords: Khmer script, text input, Unicode, security, character ordering

Résumé

ទោះបីជាមានគោលការណ៍សម្រាប់ការវាយអក្សរខ្មែរដោយប្រើ "យូនីកូដស្តង់ដារ" ក៏ដោយ ក៏ជំនួយដែលបានផ្តល់ឱ្យអ្នកប្រើប្រាស់ក្នុងការអនុវត្តតាមគោលការណ៍ទាំងនោះ នៅមានតិចតួចនៅឡើយ ។ នេះជាហេតុនាំឱ្យអ្នកប្រើប្រាស់មិនសូវចាប់អារម្មណ៍អើពើ ។ ពេលវាយអត្ថបទជាភាសាខ្មែរ អ្នកប្រើប្រាស់យល់ថាគេអាចវាយពាក្យបានច្រើនរបៀប លើយវាមើលទៅត្រឹមត្រូវដូចគ្នានៅលើអេក្រង់ ។ ជាងនេះទៅទៀត ផ្នែកខ្លះនៃភាសាខ្មែរដែលត្រូវបានអនុវត្តដោយប្រព័ន្ធប្រតិបត្តិការពេញនិយម មានលក្ខណៈល្បែងពីការកំណត់របស់ "យូនីកូដស្តង់ដារ" ។ បញ្ហានេះនាំឱ្យមានលទ្ធផលអរិជ្ជមានជាច្រើន ដូចជា ៖ ហានិភ័យផ្នែកសុវត្ថិភាពដោយសារការឆបោកនិងក្លែងបន្លំ លទ្ធភាពក្នុងការស្វែងរកមានកម្រិតទាប និង ភាពស្មុគស្មាញក្នុងដំណើរការភាសាបែបធម្មជាតិ ។ ការស្រាវជ្រាវនេះរកឱ្យឃើញនូវបញ្ហានានាក្នុងការអ៊ិនកូដនៃភាសាខ្មែរដោយ "យូនីកូដស្តង់ដារ" ។ ឃីមែន (Keyman) មានដំណោះស្រាយចំពោះបញ្ហាទាំងនេះសម្រាប់ភាសាខ្មែរនិងភាសាផ្សេងទៀតផងដែរ ។

1. Introduction

According to *How to Type Khmer Unicode* (Open Forum of Cambodia, 2004), the Khmer (Cambodian) language has 33 consonants, 14 independent vowels, 16 dependent vowels, and 13 diacritics. These are assigned individual code points in the Unicode Standard in the range "U+1780" through "U+17FF". However, given the highly complex structure of the Khmer writing system, Unicode Standard implementers have encountered some ambiguity in how words are constructed from those component codepoints, which could result in vulnerability to spoofing. In this paper, 'Spoof-vulnerable rendering' is used to describe how incorrectly-encoded clusters can be rendered in a manner that could easily be mistaken for correctly-encoded clusters, whether identical pixel-by-pixel or subtly different.

This section illustrates problematic cases this paper seeks to explore. Each example shows a correct encoding, followed by incorrect encodings of the same word rendering identically on common operating systems. The examples show the Unicode codepoints, sample output from Google Chrome 58.0 on Android 6.0.1, and the Google Search results for those encoding.

Unicode codepoints for Khmer characters always begin with "U+17", so only the last two digits will be displayed in our examples.

Case #1: (Subscript + Vowel)

A word with a subscript and a vowel in different orders look exactly alike on-screen. (See Table 1).

(1a)	81	D2	98	C2	9A	ខ្មែរ 29M
(1b)	81	C2	D2	98	9A	ខ្មែរ 175K

Table 1: ខ្មែរ 'Khmer'

Case #2: Subscript + [D2+9A]

Where there are two subscripts in a word and one of them is [D2+9A], either order of the two are rendered identically (See Table 2).

(2a)	9F	D2	8F	D2	9A	B8	ស្ត្រី 4790K
(2b)	9F	D2	9A	D2	8F	B8	ស្ត្រី 471K

Table 2: ស្ត្រី 'woman'

Case #3: Subscript + Consonant Shifter

Error! Reference source not found. shows a consonant shifter placed before a subscript (3b) and after a subscript (3a), producing near identical display.

(3a)	98	D2	99	C9	B6	84	ម្យ៉ាង 452K
(3b)	98	C9	D2	99	B6	84	ម៉្យាង 464K

Table 3: ម្យ៉ាង 'one kind'

Case #4: Consonant Shifter + Vowel

Error! Reference source not found. shows a word which could be encoded in four ways, with identical rendering results.

(4a)	9F	CA	B8	ស្ុី	6,160K
(4b)	9F	B8	BB	ស្ុី	117K
(4c)	9F	BB	B8	ស្ុី	129K
(4d)	9F	C9	B8	ស្ុី	7,860

Table 3: ស្ុី 'to eat (for animals, children)'

Case #5: {[BB] or [B6]}+[C6]

The Nikahit sign [C6] will render in a visually similar way when encoded either before or after a dependent vowel (See **Error! Reference source not found.**).

(5a)	80	BB	C6	កុំ	2,860K
(5b)	80	C6	BB	កុំ	4,830
(5c)	85	B6	C6	ចាំ	2,110K
(5d)	85	C6	B6	ចាំ	5,640

Table 4: កុំ '(negation)' and ចាំ 'to wait'

Case #6: [C4] or {[C1]+[B6]} or {[B6]+[C1]}

As shown in Table 6, the example of លោក can be correctly encoded with a single vowel codepoint [C4] but also renders identically when two vowel codepoints are used, even when ordered in reverse.

(6a)	9B	C4	80		លោក 21,900K
(6b)	9B	C1	B6	80	លោក 24K
(6c)	9B	B6	C1	80	លោក 392

Table 5: លោក 'Mr./Sir'

Case #7: [BE] or {[C1]+[B8]} or {[B8]+[C1]}

As with **Case #6**, this composite vowel will render identically whether encoded correctly as a single codepoint, or incorrectly with two codepoints.

(7a)	9F	CA	BE	94		ស៊ើប 619K
(7b)	9F	CA	C1	B8	94	ស៊ើប 3
(7c)	9F	CA	B8	C1	94	ស៊ើប 1

Table 6: ស៊ើប 'to investigate'

Case #8: [D2+8A] and [D2+8F]

In Khmer orthography, these two subscript consonants are the same character. However, in the Unicode Standard, two separate code sequences have been assigned (See **Error! Reference source not found.**).

(8a)	80	8E	D2	8A	B6	9B	កណ្តាល 967K
(8b)	80	8E	D2	8F	B6	9B	កណ្តាល 4,260K

Table 7: កណ្តាល 'Kandal'

Before exploring these cases further, this paper will look at existing research and give a brief introduction on Khmer typography in Unicode.

2. Khmer Typography

Khmer script has been included in the Unicode Standard since the 3rd edition, 1999. In earlier encoding schemes, multiple characters were frequently combined to form what is now a single Unicode character; characters were encoded according to their visual presentation rather than their function. Khmer subscript consonants were assigned individual code-points and many diacritics had multiple codepoints. For users of these encodings, if a word looked correct on-screen, it was considered to be valid.

The Unicode Standard 3.0 took a different route, encoding characters according to their function rather than their presentation. Codepoints could also display differently according to their context in the text store.

In the release of Unicode Standard 4.0 in 2003, the order of consonant shifter and subscripts were reversed. This led to incompatibilities for which a compromise ordering was proposed. *Issues in Khmer Unicode 4.0* (Solá, 2004) suggested allowing a consonant shifter either before or after the subscript: **B {R | {{Z} C}} {S}* {{Z} C} {{Z} V} {O} {ZJ S}**. To date, Solá (2004) has not been incorporated in the Unicode Standard.

Khmer Character Order	**V.**
B {S}* {C} {V} {O}[1]	3.0
B {R | C} {S {R}}* {{Z} V} {O} {S} • B - a base character • R - a robat ([CC]) • C - a consonant shifter • S - a subscript • V - a dependent vowel • Z - a zero width (non-)joiner • O - any other sign • * - (occur more than once)	4.0 5.0 6.0 7.0 8.0 9.0

Table 8: Unicode Standard Versions

3. Data and Methodology

14 words and their corresponding alternatives susceptible to spoof-vulnerable rendering were selected from the top one thousand words in the "Khmer Word Frequency List"[2]. Altogether, 62 items were examined and divided into two groups, valid and invalid text inputs, as defined by conformity to the existing Unicode rules and constraints seen below.

- **Rule #1:** No more than one dependent vowel codepoint is permitted in a syllable (Open Forum of Cambodia, 2004).
- **Rule #2:** A vowel can never be encoded before a subscript (Open Forum of Cambodia, 2004); it can only follow a consonant, a shifter or a Robat sign (Khmer Generation Panel, 2016).

[1] The abbreviations from version 4.0 of the Unicode Standard 4.0 are used here for comparison purposes.

[2] The list is available at: http://sealang.net/project/list/.

- **Rule #3:** When there are two subscripts and one of them is [D2+9A], the other subscript should come first, per spelling order (Open Forum of Cambodia, 2004).
- **Rule #4:** The shifters Triisap[3] [CA] and Muusikatoan[4] [C9] are always typed after the consonant and any subscript but before a vowel.[5] When the shifter is followed by one of the vowels ([B7], [B8], [B9], [BA], [BE], and [B6+C6]), the shifter is rendered as symbol resembling [BB]. Because [BB] itself (Open Forum of Cambodia, 2004) is always a vowel, it should not be used in place of a shifter in this situation, despite the identical appearance.
- **Rule #5:** Robat [CC] must always be inserted after the consonant above which it will be placed, before any vowels. A syllable that has [CC] cannot have any subscript, nor vowels or signs (Open Forum of Cambodia, 2004).
- **Rule #6:** Subscript marker [D2] must occur in between two consonants in a cluster (Khmer Generation Panel, 2016).
- **Rule #7:** When [BB] or [B6] is used with [C6], [C6] must be inserted after [BB] or [B6] (Open Forum of Cambodia, 2004).
- **Rule #8:** Subscripts [D2+8F] and [D2+8A] are usually placed after [93] and [8E] respectively (Sok, 2016). There are some ambiguities as to the identity of the subscript. This needs further discussion.

4. Results and Discussion

The results were rendered in four separate browsers running on Mac OS X 10.12 Sierra and Windows 10 (Safari 10.0, Mozilla Firefox 53.0, Microsoft Edge 38, and Google Chrome 58.0) and four separate platforms (iOS 10, Ubuntu 16, Windows 10 and Android 6.0.1) in order to examine cross-platform consistency.

The ambiguities that arise from different encodings and orderings in these cases leads us to recognize the potential for what we term *spoof-vulnerable rendering*, as it becomes impossible for an individual to visually distinguish whether or not the text is encoded correctly. Thus, there is potential for spoofing online where multiple rendering aliases exist for significant portions of an identifier such as a URL.

For example, the bank ACLEDA អេស៊ីលីដា can be aliased in a manner similar to **Case #4**. A malicious actor could use spoof-vulnerable renderings to deceive users into visiting their fraudulent web property in place of a legitimate site. While similar examples have been found with other scripts (Zheng, 2017), the volume of potential aliases in Khmer magnifies the problem.

Furthermore, spoof-vulnerable rendering increases the complexity of natural language processing (NLP). To be effective, NLP must be preceded by a normalisation phase to remove the redundancy that spoof-vulnerable renderings introduce to the text. This normalisation phase may be difficult to achieve because it must be based on a visual examination of a word. This also results in difficulties with searching content.

While outside the scope of this study, cross-platform inconsistencies are worse with incorrect encodings, hampering document portability.

4.1 Rendering Tests

Since some browsers and/or platforms may have the same text rendering performance, only those which have different rendering performance will be examined in greater detail.

In browsers on Windows and Mac OS X, the results demonstrate that Chrome and Firefox rendered identically: 87% spoof-vulnerable rendering; while Edge and Safari have fewer: 46% and 52% respectively (See Figure 1).

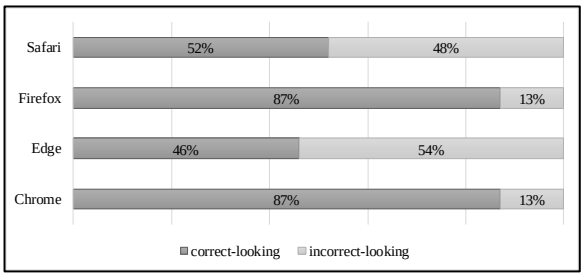

Figure 1: Browser spoof-vulnerable rendering rates

In various platforms, the data shows that iOS achieved 52%, the same as Safari. Ubuntu has the same results as Firefox and Chrome, at 87%. Windows has a result identical to the Edge browser, 46% spoof-vulnerable rendering. Finally, Android surprisingly performs differently from Chrome, suggesting that the rendering systems for Khmer text differ on these two platforms.

In most cases, the valid or correctly-encoded text input is found most frequently online, with the exception of **Case #8,** which had 4,260K results of wrong spelling versus 967K results of correct spelling.

In **Case #2**, the word can possibly be encoded in 8 different ways; and each of them break at least one rule and/or constraint. For instance, (2b) breaks **Rule #3**; (2c) **Rule #8**; (2d) **Rule #3 & #8**; (2e) **Rule #2, #6 & #8**; (2f) **Rule #2 & #6**; (2g) **Rule #2, #3, #6 & #8**; and (2h) breaks **Rule #2**, **#3, & #6.**

[3] A Triisap indicates that vowels accompanying consonants of the first series are to be pronounced as if they accompanied consonants of the second series.

[4] A Muusikatoan indicates that vowels accompanying consonants of the second series should be pronounced as if they accompanied consonants of the first series.

[5] Given the incompatibility between Unicode 3.0 and Unicode 4.0, the character ordering for shifters given precedence by this study is adopted from version 3.0, as (a) it is more consistent with Khmer spelling and pronunciation (Open Forum of Cambodia, 2004), and (b) it is used an order of magnitude more widely in practice.

(2a)	9F	D2	8F	D2	9A	B8	ស្ដ្រី 4,790K
(2b)	9F	D2	9A	D2	8F	B8	ស្ដ្រី 471K
(2c)	9F	D2	8A	D2	9A	B8	ស្ដ្រី 554K
(2d)	9F	D2	9A	D2	8A	B8	ស្ដ្រី 22K
(2e)	9F	D2	8A	B8	D2	9A	ស្ដ្រី 12
(2f)	9F	D2	8F	B8	D2	9A	ស្ដ្រី 3,060
(2g)	9F	D2	9A	B8	D2	8A	ស្ដ្រី 2,470
(2h)	9F	D2	9A	B8	D2	8F	ស្ដ្រី 151K

Table 9: 8 Possible Encoding for ស្ដ្រី

This paper will now examine how **Case #2** is rendered in iOS, Chrome and Edge, and of course Android. Text rendered with unreadable combinations of characters (or dotted circles) is displayed in shaded cells.

	iOS	Chrome	Edge	Android
(2a)	ស្ដ្រី	ស្ដ្រី	ស្ដ្រី	ស្ដ្រី
(2b)	ស្ដ្រី	ស្ដ្រី	ស្ដ្រី	ស្ដ្រី
(2c)	ស្ដ្រី	ស្ដ្រី	ស្ដ្រី	ស្ដ្រី
(2d)	ស្ដ្រី	ស្ដ្រី	ស្ដ្រី	ស្ដ្រី
(2e)	ស្ដ្រី	ស្ដ្រី	ស្ដ្រី	ស្ដ្រី
(2f)	ស្ដ្រី	ស្ដ្រី	ស្ដ្រី	ស្ដ្រី
(2g)	ស្ដ្រី	ស្ដ្រី	ស្ដ្រី	ស្ដ្រី
(2h)	ស្ដ្រី	ស្ដ្រី	ស្ដ្រី	ស្ដ្រី

Table 10: Rendered Texts of 2a-2h

(2a) show correct encodings for the two words used in this example. (2b) to (2h) show alternate, invalid encodings for those words: they break the rules laid out in **Section 3**. Our data shows that Android is most susceptible to spoof-vulnerable rendering, as all invalid encodings were indistinguishable from the valid encodings. Chrome is also highly susceptible to security attack for rarely does it render text in a way that gives users feedback on the incorrect encoding.

5. Conclusion

This paper has noted several cases in which the Unicode Standard's Khmer script rules can be bypassed while producing visually identical results on many standard browsers and operating systems, which results in spoof-vulnerable renderings. This also has impacts for NLP and document portability.

In order to address spoof-vulnerable rendering, rendering engines in all browsers and platforms should be updated to ensure that invalid encodings do not render identically to valid ones. As the majority of diacritics and vowels in Khmer render with a dotted circle when isolated (to indicate the lack of a base character for combining), we advise that invalid representations be likewise rendered with isolated combining characters to facilitate security and prevent spoofing.

6. Limitations and Future Research

The words were chosen for their potential to demonstrate spoof-vulnerable rendering interactions based on a visual analysis and thus do not represent a comprehensive sample of rendering and encoding issues in Khmer script.

There are a number of other languages that use the Khmer script; these have not been analyzed for this paper and may introduce additional complexities or ambiguities.

While some preliminary rules capable of addressing the example spoof-vulnerable readings of this paper may be quickly identified, development of a more complete set of rules will require further research and testing.

7. Bibliographical References

Open Forum of Cambodia. (2004). *How to Type Khmer Unicode*, Version 1.0:7–14.

Solá, J. (2004). *Issues in Khmer Unicode 4.0. Open Forum of Cambodia*, Version 2.0:6-7.

Khmer Generation Panel. (2016). Association for computing machinery. *Proposal for Khmer Script Root Zone Label Generation Rules*, Version 1.5:15.

Sok, Makara. (2016). *Phonological Principles and Automatic Phonemic and Phonetic Transcription of Khmer Words.* Master's Thesis: 35. Retrieved from: http://inter.payap.ac.th/wp-content/uploads/linguistics_students/Makaras-Thesis.pdf

Zheng, Xudong. (2017). *Phishing with Unicode Domains*. April 14. Online: https://www.xudongz.com/blog/2017/idn-phishing/