Universal Acceptance

Work Item: UA-Readiness of Open-Source Code (Pilot)

Ver.: 2019-10-17

Purpose

The proposed work is building upon foundational knowledge published in UASG018 (Review Programming Languages and Frameworks for Compliance with Universal Acceptance Good Practice)[1] and UA Assessment of Programming Language LIbraries[2].

The work aims to determine the usage of *domain name/email address* validation procedure (library or ad-hoc code) by applications available in open-source code repositories. Specifically:

1. If no procedures or libraries are found, attempt to isolate the code doing validation to study it
2. If procedures or libraries are found, identify the type of support provided

- Coordination Group proposing the work item: UA Measurements WG
- Reference to the Action plan: FY20
- Reference to work item(s): M4
- Suggested budget: USD 40,000 (for the entire work; not just the pilot)

This work identifies the UA readiness but does not mitigate it. The results of this work will be shared with the UA Tech WG to prioritize, plan and undertake remediation, which will be done in the next phase of this work.

Description of Work

The work entails developing a crawler which will automatically check for usage of *domain name/email address* validation procedure (library or ad-hoc code) by applications available in open-source code repositories, with the following scope:

1. Programming language: `Python, Java`
2. Source Code repository: `Github`
3. Tests:
    a. Domain name (i18n and non-i18n) vis-a-vis UA-actions:
        i. Must determine `{acceptance, validation, process}`
        ii. Nice to have `{storage}, {display}`
    b. Email address (i18n and non-i18n) vis-a-vis UA-actions:
        i. Must determine `{acceptance, validation, process}`

[1] UASG018 - https://uasg.tech/wp-content/uploads/documents/UASG018-en-digital.pdfhttps://uasg.tech/wp-content/uploads/documents/UASG018-en-digital.pdf

[2] https://uasg.tech/software/

ii. Nice to have `{storage}, {display}`

4. Test Cases: Use domain name classes and email address classes per UASG004[3]

Deliverables

Following is the itemized list of deliverables expected from this work item.

1. Record findings per application in template: see Table 1
    a. If no procedures or libraries were found, attempt to isolate the code doing validation to study it.
    b. If procedures or libraries were found, identify the type of support provided.
2. Provide parameters, such as create date, last update or count of downloads or any other appropriate data point to assist the sorting of the data to help prioritization.
3. Undertake an alternate method to validate findings. A sample is acceptable as per the following details:
    a. Sample size: 30
    b. Sample description: sample should cover all test scenarios
4. Develop a report which includes recommendations to:
    a. Quantitative results and their validity based on the alternate method
    b. Prioritize UA-readiness mitigation work
    c. Understand how code is being used to offer solutions (e.g. Table 2)
    d. Develop categories of which UA issue occurs based on the automated checks. Identify applicable categories of ua issue for each application checked to help substantiate future outreach to developers and maintainers
    e. Identify and prioritize of functionality or use relevant for UA (e.g. beyond validating a domain names and email addresses)

Table 1: Application Assessment Form

Topic	Examples	I18n/EAI support	ASCII validation
Application	`Accounting software`	Yes/No?	
Libraries used by Application	`to process hostname and email addresses`	Yes/No?	
Programming Language used by Application and library	`Python (version x.y), Java (version x.y)`	Yes/No?	
Operating System	`Linux`	Yes/No?	
Mitigation recommendation			
Maintainer contact information			

[3] UASG004 - https://uasg.tech/wp-content/uploads/documents/UASG004-en-digital.pdf

Table 2: Issue and Solution Matrix

possible cases for apps to use well-known libs for UA	**Possible conclusion**	**Next possible steps**
app do not have any signature of a UA lib	most likely not supporting UA (because UA is difficult). it may be possible that they develop their own code, but most likely not.	ask maintainers if they are aware of UA?
app does have a signature of using a well-known old (for example idna2003) UA lib	most likely not supporting UA, since the library they are using is not supporting UA properly	tell maintainers to use a better lib
app does have a signature of using a known (good) UA lib	most likely supporting UA, but they may use it wrongly.	if we have time, test it? or read the code?

Timeline

- Tentative start date: 15 November 2019
- Tentative end date: 30 January 2019

History (if any)

The work builds on earlier work in UASG018.