Babylon Cross-Application Scripting

A security advisory

Yair Amit <amityair@il.ibm.com> - Roee Hay <roeeh@il.ibm.com>
IBM Rational Application Security Research Group

November 10, 2010

1 Introduction

Babylon is a single-click computer online dictionary and translation software which is also capable of translating whole documents and web pages. The translation and dictionary results are presented to the user via the Trident layout engine (an in-app/embedded Internet-Explorer rendering engine).

2 Vulnerability

Babylon fails to sanitize user input before rendering it on the Trident control, effectively leading to a Cross-Application Scripting vulnerability.
The user's input can originate from the following sources:

1. Babylon's main translation interface: When searching for a non-existent term Babylon pushes the string "No matches were found for '*non-existent term*'. Search the web for *non-existent-term*". Unfortunately, '*non-existent term*' is not HTML encoded or validated before it is written on the embedded browser (the translation process transparently validates translatable input).
2. Document translation: Documents (such as PDF files) may contain text which cannot be translated (e,g. JavaScript code), in such case it is rendered without HTML encoding or validation.
3. Web site translation: Same as the document translation case. Text which is not translated is simply rendered on the Trident control.

In order to trigger an attack, the victim has to be lured into translating malicious text, either in a document or on a web page. Although the scenarios require some social-engineering, some of them are feasible and realistic.

3 Impact

The Trident control runs in Local Machine Zone (LMZ) which is not Locked down. This allows a malicious script to perform the following actions:

1. Interact with other websites in the background while using persistent cookies to impersonate the victim: This can be done using XHR. The payload can leak the response's body which may contain sensitive information in case the victim has been authenticated on the site prior to the attack. The payload can also perform actions on the site on behalf of the victim.

2. Collect information from local files: This can be done using XHR or by a hidden iframe with the filename's as the source. Since the payload runs in a non-Locked-down LMZ, the XHR response or iframe's body can be inspected and sent back to the attacker.

3. Code execution: System commands can be executed by using the `Wscript.Shell` ActiveX object to practically take over the whole system.

4 Vulnerable versions

Versions prior to 8.0.7 are susceptible to this vulnerability.

5 Acknowledgments

We would like to thank the Babylon team for their quick responses and the efficient way in which they handled this security issue.

6 References

1. Babylon's homepage: http://www.babylon.com/

2. Enhanced Browsing Security:
http://technet.microsoft.com/en-us/library/bb457150.aspx