Example 1: Reporting Stored XSS

`Title`: Stored Cross-Site Scripting (XSS) in X Admin Panel

`CWE`: CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

`CVSS 3.1 Score`: 5.5 (Medium)

`Description`: During our testing activities, we identified that the "X for administrators" web application is vulnerable to stored cross-site scripting (XSS) attacks due to inadequate sanitization of user-supplied data. Specifically, the file uploading mechanism at "Admin Info" -> "Secure Data Transfer" -> "Load of Data" utilizes a value obtained from user input, specifically the uploaded file's filename, which is not only directly reflected back to the user's browser but is also stored into the web application's database. However, this value does not appear to be adequately sanitized. It, therefore, results in the application being vulnerable to reflected and stored cross-site scripting (XSS) attacks since JavaScript code can be entered in the filename field.

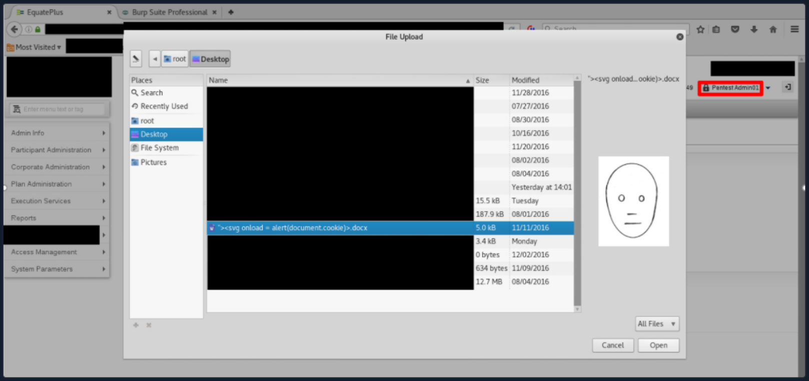
`Impact`: Cross-Site Scripting issues occur when an application uses untrusted data supplied by offensive users in a web browser without sufficient prior validation or escaping. A potential attacker can embed untrusted code within a client-side script to be executed by the browser while interpreting the page. Attackers utilize XSS vulnerabilities to execute scripts in a legitimate user's browser leading to user credentials theft, session hijacking, website defacement, or redirection to malicious sites. Anyone that can send data to the system, including administrators, are possible candidates for performing XSS attacks against the vulnerable application. This issue introduces a significant risk since the vulnerability resides in the "X for administrators" web application, and the uploaded files are visible and accessible by every administrator. Consequently, any administrator can be a possible target of a Cross-Site Scripting attack.

`POC`:

Step 1: A malicious administrator could leverage the fact that the filename value is reflected back to the browser and stored in the web application's database to perform cross-site scripting attacks against other administrators by uploading a file containing malicious JavaScript code into its filename. The attack is feasible because administrators can view all uploaded files regardless of the uploader. Specifically, we named the file, as follows, using a Linux machine.

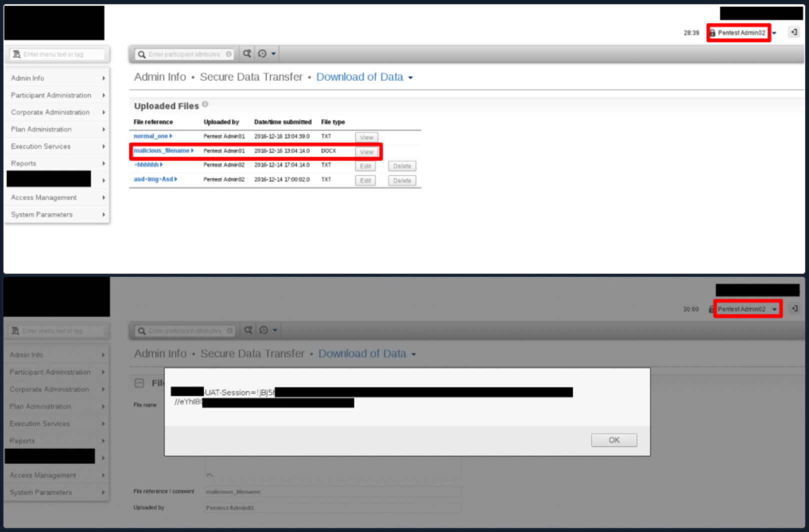
Code: javascript

```javascript
"><svg onload = alert(document.cookie)>.docx
```



Name	Size	Modified
[redacted]		11/28/2016
[redacted]		07/27/2016
[redacted]		08/30/2016
[redacted]		10/16/2016
[redacted]		11/20/2016
[redacted]		08/02/2016
[redacted]		08/04/2016
[redacted]		Yesterday at 14:01
[redacted]	15.5 kB	Tuesday
[redacted]	187.9 kB	08/01/2016
"><svg onload = alert(document.cookie)>.docx	5.0 kB	11/11/2016
[redacted]	3.4 kB	Monday
[redacted]	0 bytes	12/02/2016
[redacted]	634 bytes	11/09/2016
[redacted]	12.7 MB	08/04/2016

Step 2: When another administrator clicks the view button to open the abovementioned file, the malicious JavaScript code in the file's filename will be executed on the browser.



Admin Info • Secure Data Transfer • Download of Data

Uploaded Files

File reference	Uploaded by	Date/time submitted	File type		
normal_one	Pentest Admin01	2016-12-16 13:04:39.0	TXT	View	
malicious_filename	Pentest Admin01	2016-12-16 13:04:14.0	DOCX	View	
>hhhhhh	Pentest Admin02	2016-12-14 17:04:14.0	TXT	Edit	Delete
asd<img>Asd	Pentest Admin02	2016-12-14 17:00:02.0	TXT	Edit	Delete

```
[redacted]UAT-Session=!jBj5[redacted]
//eYhl8[redacted]
```

File name
File reference / comment: malicious_filename
Uploaded by: Pentest Admin01

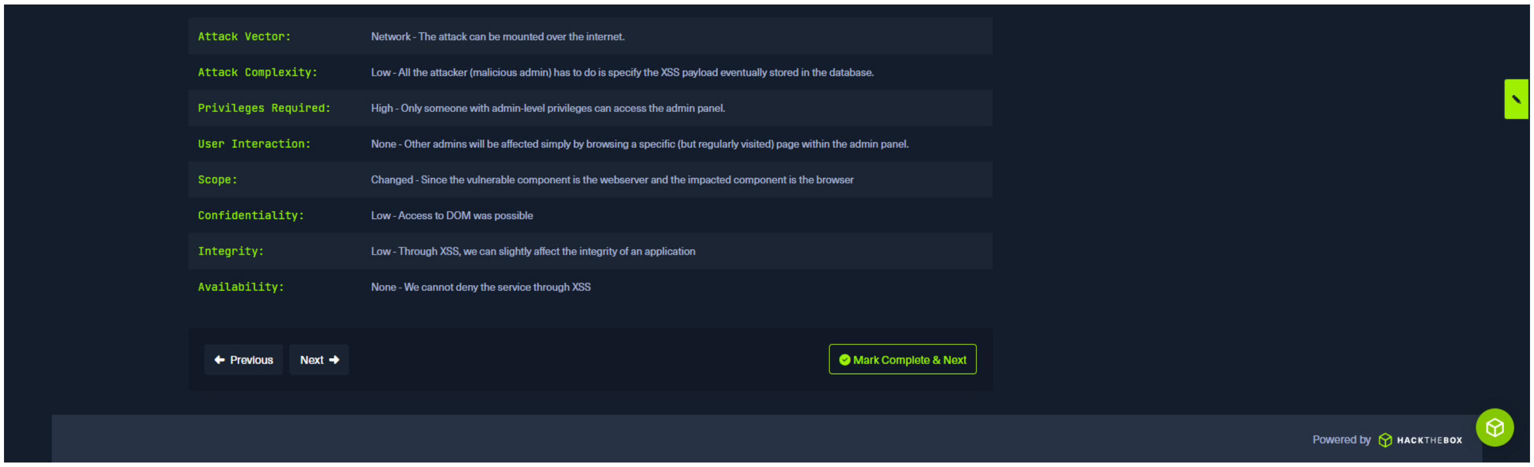
CVSS Score Breakdown

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left



`Attack Vector:`	Network - The attack can be mounted over the internet.
`Attack Complexity:`	Low - All the attacker (malicious admin) has to do is specify the XSS payload eventually stored in the database.
`Privileges Required:`	High - Only someone with admin-level privileges can access the admin panel.
`User Interaction:`	None - Other admins will be affected simply by browsing a specific (but regularly visited) page within the admin panel.
`Scope:`	Changed - Since the vulnerable component is the webserver and the impacted component is the browser
`Confidentiality:`	Low - Access to DOM was possible
`Integrity:`	Low - Through XSS, we can slightly affect the integrity of an application
`Availability:`	None - We cannot deny the service through XSS