RuheIT

VULNERABILITY REPORT

THURSDAY, MAY 26, 2022

MODIFICATIONS HISTORY

Version	Date	Author	Description
1.0	05/26/2022	Jonathan	Initial Version

TABLE OF CONTENTS

GENERAL INFORMATION

SCOPE

undefined has mandated us to perform security tests on the following scope:

- https://ruheit.xyz
- https://intranet.ruheit.xyz
- https://edu.ruheit.xyz

ORGANISATION

The testing activities were performed between 05/20/2022 and 05/24/2022.

EXECUTIVE SUMMARY

VULNERABILITIES SUMMARY

Following vulnerabilities have been discovered:

Risk	ID	Vulnerability	Affected Scope
Critical	IDX-001	Authentication Bypass	https://intranet.ruheit.xyz
Medium	VULN-002	Cross-Site Scripting	https://edu.ruheit.xyz
Medium	VULN-003	HTML Injection	https://edu.ruheit.xyz

TECHNICAL DETAILS

AUTHENTICATION BYPASS

CVSS SEVERITY	Critical		CVSSV3 SCORE	9.1
CVSSV3 CRITERIAS	Attack Vector :	**Network**	Scope :	**Unchanged**
	Attack Complexity :	**Low**	Confidentiality :	**High**
	Required Privileges :	**None**	Integrity :	**High**
	User Interaction :	**None**	Availability :	**None**
AFFECTED SCOPE				
DESCRIPTION	This issue allow an attacker to bypass login panel using OAUTH method of authentication, but you need to have to know the users that are registered on the webpage. Then you can change on POST DATA, the email of another user, then when you send the data modified, you can enter with that user on intranet.			
OBSERVATION				
TEST DETAILS				
REMEDIATION	Make the verification mode, using UID not the email or username			
REFERENCES				

CROSS-SITE SCRIPTING

CVSS SEVERITY	Medium		CVSSV3 SCORE	6.1
CVSSV3 CRITERIAS	Attack Vector :	**Network**	Scope :	**Changed**
	Attack Complexity :	**Low**	Confidentiality :	**Low**
	Required Privileges :	**None**	Integrity :	**Low**
	User Interaction :	**Required**	Availability :	**None**
AFFECTED SCOPE				
DESCRIPTION	Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it. An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page. For more details on the different types of XSS flaws, see: Types of Cross-Site Scripting.			
OBSERVATION				
TEST DETAILS				
REMEDIATION	Preventing cross-site scripting is trivial in some cases but can be much harder depending on the complexity of the application and the ways it handles user-controllable data. In general, effectively preventing XSS vulnerabilities is likely to involve a combination of the following measures: • **Filter input on arrival.** At the point where user input is received, filter as strictly as possible based on what is expected or valid input. • **Encode data on output.** At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding. • **Use appropriate response headers.** To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can use the `Content-Type` and `X-Content-Type-Options` headers to ensure that browsers interpret the responses in the way you intend. • **Content Security Policy.** As a last line of defense, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur.			
REFERENCES				

HTML INJECTION

CVSS SEVERITY	Medium		CVSSV3 SCORE	6.1
CVSSV3 CRITERIAS	Attack Vector :	**Network**	Scope :	**Changed**
	Attack Complexity :	**Low**	Confidentiality :	**Low**
	Required Privileges :	**None**	Integrity :	**Low**
	User Interaction :	**Required**	Availability :	**None**
AFFECTED SCOPE				
DESCRIPTION	This vulnerability allows remote attackers to inject their own malicious HTML code into an imported snapshot, aka HTML Injection. Successful exploitation will allow attacker-supplied HTML to run in the context of the affected browser, potentially allowing the attacker to steal authentication credentials or to control how the site is rendered to the user. NOTE: the vendor disputes the risk because there is a clear warning next to the button for importing a snapshot.			
OBSERVATION				
TEST DETAILS				
REMEDIATION	The most common way of detecting HTML injection is by looking for HTML elements in the incoming HTTP stream that contains the user input. A naïve validation of user input simply removes any HTML-syntax substrings (like tags and links) from any user-supplied text. However, there are many instances where the application expects HTML input from the user. For example, this happens when the user submits visually-formatted text or text containing links to legitimate sites with related content. To avoid false positives, the security mechanism that detects possible injections and protects the application should learn in what application context user input is allowed to contain HTML. Also, it should be able to stop HTML input if it learns that such text is pasted as-is in web page generated by vulnerable application components.			
REFERENCES				