Sri Lanka Institute of Information Technology

Web Security – IE2062

Topic: Bug Bounty Report 8

Y2S2.WE.CS

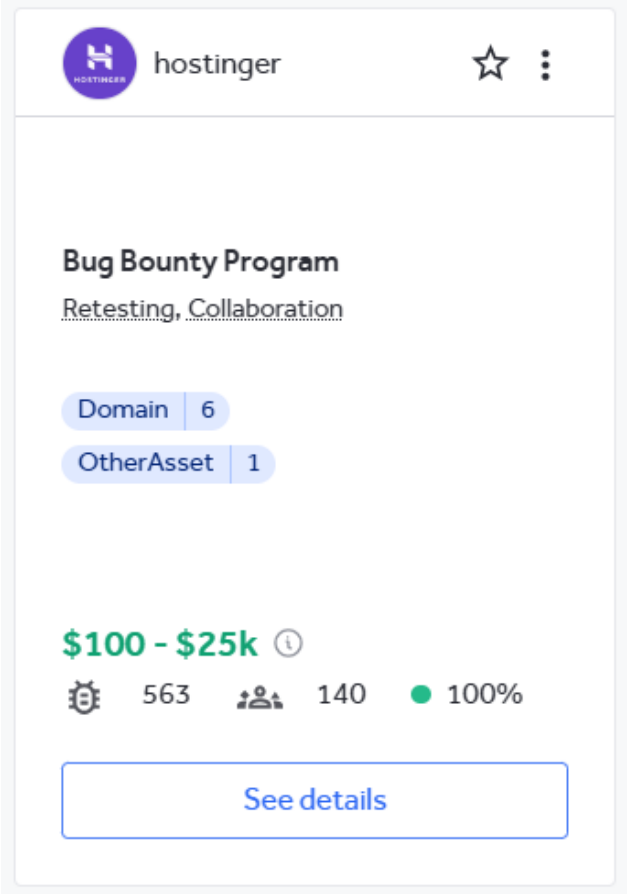
Name: S.D.W.Gunaratne

(IT23241978)

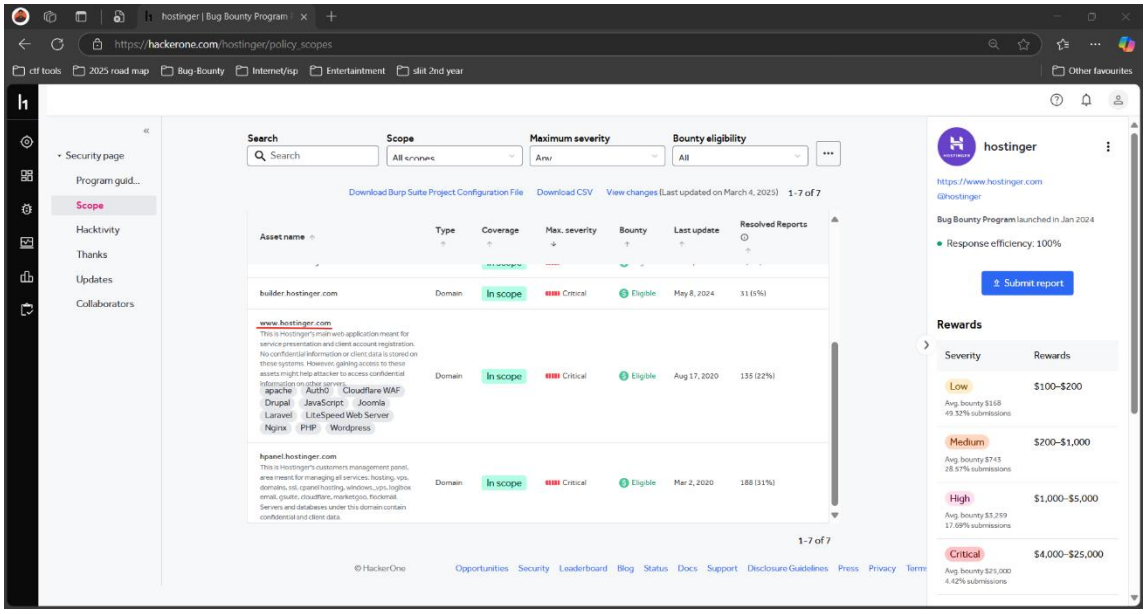
Table of Content

How I started?

1. Once I search from Hacker one, I saw a hostinger bug bounty program.

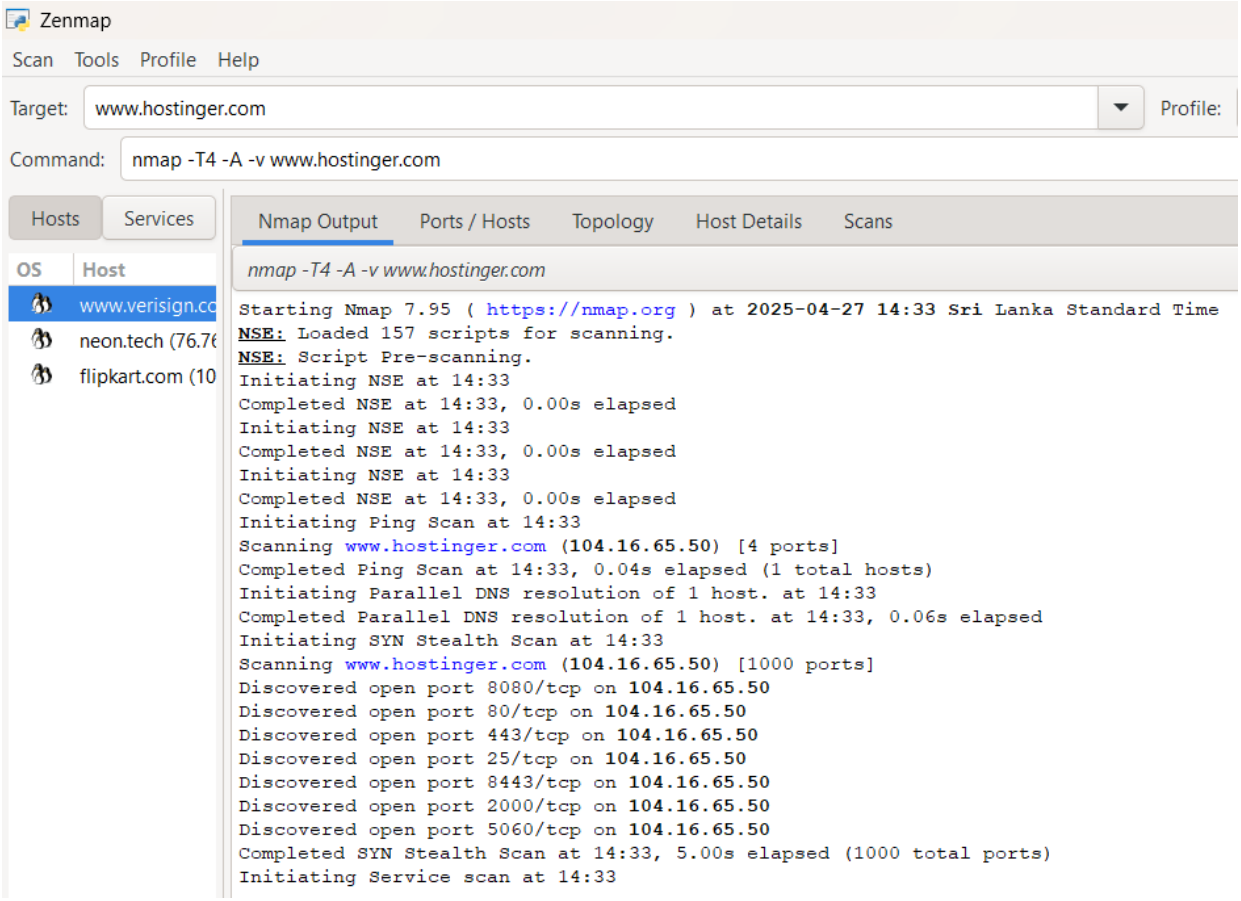


2. Then, I discovered full main domain allowed for scope, so that I choose https://www.hostinger.com .



3. I use several methods/tools to do penetration testing.

4. First, I used Nmap. It helps me to find what are the open ports, Identify the web technologies such as webservers.

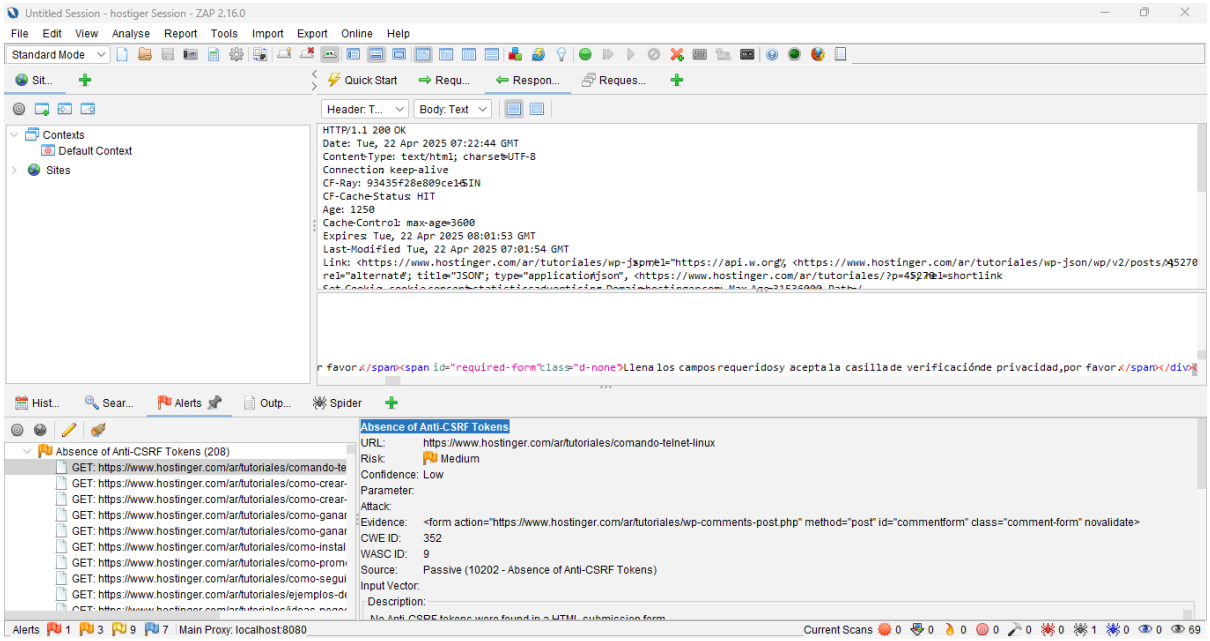


5. Secondly, I used Subfider tool to find hidden or forgotten web asserts. Because hidden web assert can have poor security, unpatched vulnerabilities.

6. Thirdly, I used Wafwoof tool to find website is protected by a WAF (web application firewall). Because if WAF is active, so pen testers do their test without blocked, and they can do their testing with bypass WAF.

```
File  Actions  Edit  View  Help

                    ~ WAFW00F : v2.3.1 ~

[*] Checking https://www.hostinger.com/
[+] The site https://www.hostinger.com/ is behind Cloudflare (Cloudflare Inc.
) WAF.
[~] Number of requests: 2

┌──(root㉿kali2025)-[~]
└─# wafw00f https://coda.io/

                    ~ WAFW00F : v2.3.1 ~
            ~ Sniffing Web Application Firewalls since 2014 ~
```

7.Finaly, I use OWASP zap to automatically find the vulnerabilities.



With getting these tool's support, I found below details about vulnerability.

2) Introduction

1.1 Domain	https://www.hostinger.com
1.2 Severity	• **Medium**

3) Vulnerability

3.1 Vulnerability title	Absence of Anti-CSRF Tokens OWASP_2021_A01 CWE-352
3.2 Vulnerability description	No Anti-CSRF tokens were found in a HTML submission form. A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf. CSRF attacks are effective in a number of situations, including: * The victim has an active session on the target site. * The victim is authenticated via HTTP auth on the target site. * The victim is on the same local network as the target site. CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.

3.3 Affected components	" https://www.hostinger.com/fr/tutoriels/wp-comments-post.php" form does not include an anti-CSRF token. Evidence: <form action="https://www.hostinger.com/fr/tutoriels/wp-comments-post.php" method="post" id="commentform" class="comment-form" novalidate>
3.4 Impact assessment	Risk of Unauthorized Actions: The Hacker can forge a malicious POST request that performs unwanted submissions or actions within the victim's authenticated session. Privilege Abuse: The session of the victim can be used for posting or submitting data without authorization when the user(victim) is logged in. Chained Attack Vector: In combination with XSS, the Hacker would be capable of bypassing CSRF protections completely and performing automated attacks within the same origin.
3.5 Steps to reproduce	Here are the steps to reproduce: 1. Go to ZAP tool or Zap generated report. 2. Then check the Response body

Response

▶ Status line and header section (1302 bytes)

▼ Response body (254174 bytes)

```
<!DOCTYPE html>
<html lang="en-US">
<head><meta charset="UTF-8">
```

3. by using CTRL+F search about CSRF in our respond header, we can't find any.

3.6 Proof of concept

▼ Response body (234901 bytes)

```
<!DOCTYPE html>
<html lang="fr-FR">
<head><meta charset="UTF-8">
<script>if(navigator.userAgent.match(/MSIE|Internet
```

Ctrl+f and search.

sliit 2nd year

<form action="https://www.h

```
class="d-none">Veuillez remplir les champs obligatoires et
accepter la case de confidentialité.</span></div><form
action="https://www.hostinger.com/fr/tutoriels/wp-comments-
post.php" method="post" id="commentform" class="comment-
form" novalidate><div id="comment-texarea-input-border"
class="new-input-border position-relative mt-20 mt-30-sm">
```

This form does not include a CSRF token like csrf_token or _csrf. ex:-

```
<input type="hidden" id="ak_js_1" name="ak_js" value="111"/>
<script
```

If this is secured it should be like this:

	<input type="hidden" name="csrf_token" value="d7g8f9we78fywe97f"> But this site doesn't use CSRF field, that means this site doesn't use CSRF. If it is there, they should mention it. So the danger is which would enable an attacker to form a malicious POST request that would be accepted by the server in case the user is authenticated.
3.7 Proposed mitigation or fix	Phase 01: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, use anti-CSRF packages such as the OWASP CSRFGuard. Phase 02: Implementation Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script. Phase 03: Architecture and Design Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330). Note that this can be bypassed using XSS. Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation. Note that this can be bypassed using XSS. Use the ESAPI Session Management control. This control includes a component for CSRF. Do not use the GET method for any request that triggers a state change. Phase 04 : Implementation Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because

	users or proxies may have disabled sending the Referer for privacy reasons.