Honeywell

ENTERPRISE
SECURITY
ASSURANCE

SECURITY ASSESSMENT REPORT

CONFIDENTIAL

HBT Sales – Jul 11th Rel

Report published on July 8, 2024

DISCLAIMER

This document contains Honeywell proprietary information. Information contained herein is to be used solely for the purpose submitted, and no part of this document or its contents shall be reproduced, published, or disclosed to a third party without the express permission of Honeywell International Sàrl. While this information is presented in good faith and believed to be accurate, Honeywell disclaims the implied warranties of merchantability and fitness for a purpose and makes no express warranties except as may be stated in its written agreement with and for its customer. In no event is Honeywell liable to anyone for any direct, special, or consequential damages. The information and specifications in this document are subject to change without notice.

The information presented in this document is provided as is and without warranty. Security assessments are a “point in time” analysis and random sampling; as such, it is possible that something in the environment could have changed or there may be undiscovered issues or emerging threats since the tests reflected in this report were run. For this reason, this report should be considered a guide, not a 100% representation of the risk threatening your systems, networks and applications.

DOCUMENT CONTROL

App/Project Name:	HBT Sales
SSR No:	SSR300870
Report Version:	v1
Published Date:	July 8, 2024
Report Classification:	CONFIDENTIAL
Report Authors:	Bhanutej , Pramod

ENTERPRISE SECURITY ASSURANCE CONTACT

Joe Kadisak
Sr. Director Cyber Security
Joe.Kadisak@honeywell.com

Leo Rethinasamy
Sr. Cyber Security Manager
Leo.Rethinasamy@honeywell.com

Bhantej Thota
Cyber Sec Assistant Manager
bhanutej.thota@honeywell.com

Pramod Kuthuru
Cyber Sec Archt/Engr II
pramod.kuthuru@honeywell.com

DOCUMENT DISTRIBUTION LIST

Mathew, Rony	Rony.Mathew@Honeywell.com
Tawde, Prachi (CSW)	Prachi.Tawde@Honeywell.com
Awasthi, Vipul (CSW)	Vipul.Awasthi@Honeywell.com
Pawar, Sumit (CSW)	Sumit.Pawar@Honeywell.com
Bongu, Ravi Kumar	RaviKumar.Bongu@Honeywell.com
Joshi, Vithika	Vithika.Joshi@Honeywell.com
Kanakadandi, Lakshmi Nivedita	LakshmiNivedita.Kanakadandi@Honeywell.com

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Enterprise Security Assurance team was engaged to assess the security posture of **HBT Sales**. The purpose of this engagement was to conduct targeted testing and to ensure broad coverage of the most common types of vulnerabilities as defined by Industry Standards and non-compliance to mandatory Honeywell Global Security Policy. This assessment provides insight into the resilience of the application to withstand attacks from unauthorized and authorized users and determines the trustworthiness of the application. Assessment report contains technical details of vulnerabilities discovered with proof of concept, steps to reproduce, risk mitigation recommendations that need to be implemented to ensure the systems are secure from risks arising due to the discovered vulnerabilities.

Overall Risk:
HIGH

Finding Counts:
0 Critical
2 High
2 Medium
5 Low
0 Informational

Assessment Type:
Grey Box

Assets in Scope:
Web Application

During the course of engagement, **9 vulnerabilities** were identified, which are of **2 High, 2 Medium & 5 Low** severity security vulnerabilities. Below table shows the in-scope assets and breakdown of findings by severity per asset. Please refer the appendix for more information on how the severity is calculated.

	Critical	High	Medium	Low	Info
Web Application	0	2	2	5	0

Top Vulnerabilities of the following kind were identified :

- A3 - Injection
- A5 - Security Misconfiguration
- A2 - Cryptographic Failures

These vulnerabilities represent the greatest immediate risk to **HBT Sales** application, therefore should be prioritized for remediation. As per HGS secure operations policy, all Critical, High, Medium issues must be remediated, Low issues are recommended; not mandatory.

ASSESSMENT OVERVIEW

Objective: Understand the **HBT Sales** application security posture. Identify as many security flaws as possible in the designated time and scope. Find missing controls and Honeywell secure policy violations. Provide recommendations for remediation and secure coding and configuration best practices.

Test Coverage: Compliance to Honeywell/Industry/OWASP Top 10 risks, but not limited.

In Scope:
Web Application

Out of Scope:
Social Engineering
Denial of Service
Intrusive Testing
Exploitation
Vulnerability Fixes
SAST/Code Reviews

Application URL
https://buildings.stage.honeywell.com/shop/honeywell/en/login

TEST EXECUTION

Assessment was conducted on non-prod environment (i.e., Stage) & executed using both automated and manual methods leveraging industry standard tools, security best practices, custom scripts and in-house tools including Nmap, ZAP, Wireshark, Burp etc. Also, verified business logics & workflows. Adopted hybrid approach for better test coverage and all the issues are validated manually to have zero false positives. This table shows the list of all high-level test categories and their execution status.

Test Category	Status
A1 - Broken Access Control	PASS
A2 - Cryptographic Failures	FAIL
A3 - Injection	FAIL
A4 - Insecure Design	PASS
A5 - Security Misconfiguration	FAIL
A6 - Vulnerable and Outdated Components	PASS
A7 - Identification and Authentication Failures	PASS
A8 - Software and Data Integrity Failures	PASS
A9 - Security Logging and Monitoring Failures	PASS
A10 - Server- Side Request Forgery	PASS

SUMMARY OF FINDINGS

In all, the **HBT Sales** application **FAILED** to meet the security objectives. **2 High, 2 Medium & 5 Low** risk vulnerabilities were identified during the assessment. These vulnerabilities represent the greatest immediate risk to the application and should be prioritized for remediation.

Below table shows the vulnerabilities statistics:

9 TOTAL TRUE POSITIVES	
9* OPEN	**0** CLOSED
	0 DEFERRED

** CRITICAL, HIGH, MEDIUM & LOW*

Severity	Vulnerability Count	Remediation Timeline #
CRITICAL	0	Immediate
HIGH	2	30 days
MEDIUM	2	30 days
LOW	5	Next Enhancement
INFO	0	Best Practices

Below table shows the list of vulnerabilities and severity classification.

#	VULNERABILITIES	Severity
1	Cross Site Scripting (XSS)	HIGH
2	HTML Injection	HIGH
3	Unvalidated Redirects	MEDIUM
4	Improper Input Validation	MEDIUM
5	Poor Error Handling	LOW
6	Unsafe HTTP Verbs Used	LOW
7	User Enumeration Possible	LOW
8	Banner Grabbing	LOW
9	Insecure Cookie Configuration	LOW

DETAILED LIST OF FINDINGS

1. Cross Site Scripting (XSS)

Cross-site scripting (XSS) attacks occur when an application is used to send malicious code, generally in the form of a browser side script, to a different end user and the input is not sanitized by the application which allows an attacker to inject JavaScript, VBScript, ActiveX or Flash into a vulnerable application. This could be used to gather data from the user without his knowledge.

HIGH RISK

CVSS Score: **8.2**

OWASP Category: **A3**

Observation:

Noted that when a file is uploaded, the JavaScript code within the filename is executed leaving the application vulnerable to Stored XSS.

Impact:

An attacker can control a script that is executed in the victim's browser and can typically fully compromise that user. Amongst other things, the attacker can:

- Steal session cookies, which allows the attacker to impersonate the victim.
- Perform any action within the application that the user can perform.
- Initiate interactions with other application users, including malicious attacks, that will appear to originate from the initial victim user.

Recommendation:

The best way to prevent cross-site scripting is to make sure that the application does not make use of user input in the rendered html pages, without validating it first.

- Escape all special characters when preparing the output.
- Any user input must be properly validated, special characters such as '"< >, * # % & / \ |.( ) : ; - etc. should be filtered out wherever not expected.
- Whitelisting approach which is found to be an effective means of enforcing strict input validation rules.
- In case special characters are required, these should be encoded using an appropriate encoding mechanism (e.g., HTML encoding) to ensure that these are not interpreted by the browser.
- Validation of all headers, cookies, and query strings, form fields, and hidden fields (i.e., all parameters) against a rigorous specification of what should be allowed.
- Use appropriate response headers such as use the X-XSS-Protection
- Never Insert Untrusted Data Except in allowed Locations. Any untrusted data should only be treated as displayable text & Implement a strong Content Security Policy (CSP).

References:

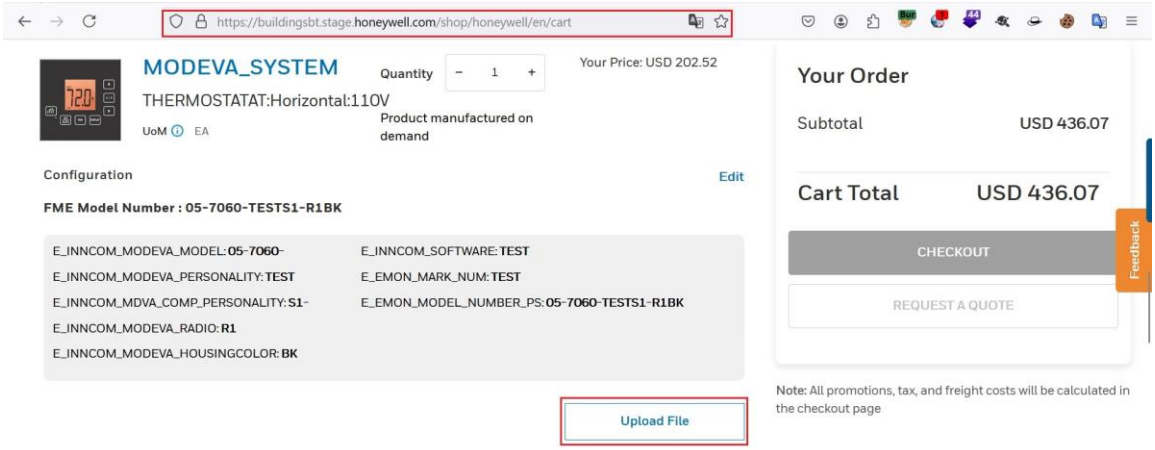
Industry Standard:
https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS)
https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Steps to Reproduce:

1. Login into web application
2. Navigate to cart page and upload '.jpg file.
3. Intercept above request using any proxy tool and add XSS payload as filename
4. Submit the above request from proxy
5. Observe the payload is executed successfully once the file is uploaded with xss payload in the name of file.

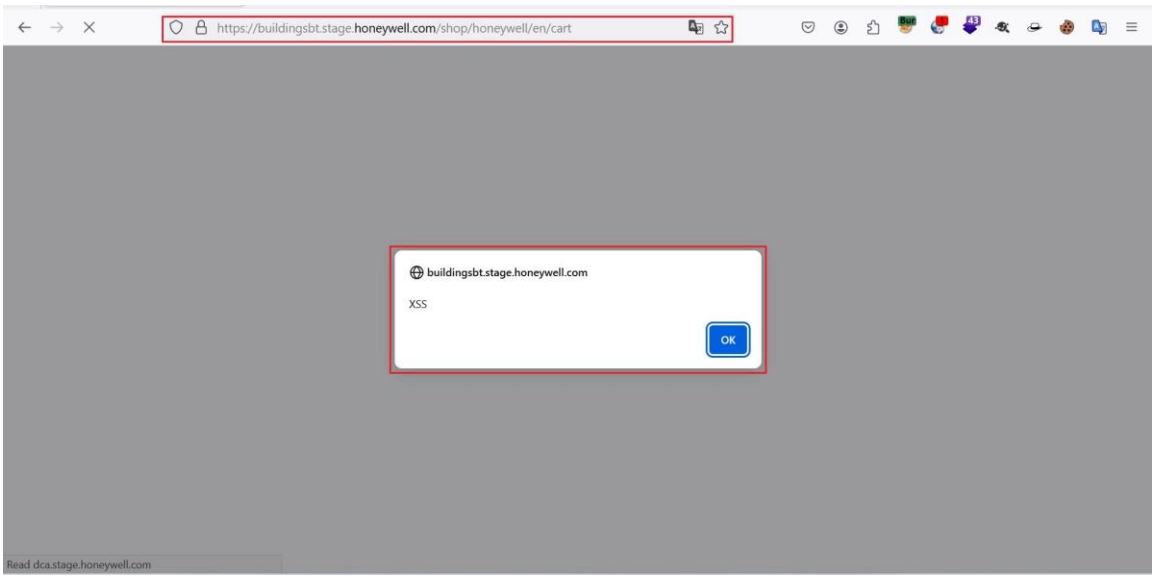
Proof of Concept:

Below figure shows file upload option at the client-side.



Below figure shows the filename replaced with an XSS payload using a proxy tool

Below Figure shows XSS Payload stored and executed successfully.



2. HTML Injection

Hypertext Markup Language (HTML) injection, also sometimes referred to as virtual defacement, is an attack on a user made possible by an injection vulnerability in a web application. When an application does not properly handle user supplied data, an attacker can supply valid HTML, typically via a parameter value, and inject their own content into the page. In addition, Link Injection flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation and escaping.

HIGH RISK

CVSS Score: **7.2**

OWASP Category: **A3**

Observation:

Noted that the application is allowing the users to inject HTML code within the application.

Impact:

Attacker can inject malicious links and perform phishing attacks or even deface the application page. This is an attack that is closely related to XSS. The difference is in the type of attack that leverages the vulnerability. While XSS uses script tags to run JavaScript, HTML injection simply uses HTML to modify the page for malicious reasons. Link Injection allows attackers to embed scripts in the victim's browser which can deface web sites or redirect the user to malicious sites.

Recommendation:

- Validate user input data using whitelists of allowed characters. For example, the character set for names should be restricted to alphabetic characters (plus hyphens, accented characters and apostrophes) and email addresses should be validated before further use.
- HTML encode HTML meta characters that could be used to create new HTML elements or attributes in the response, including “<”, “>”, “”” and “=”.
- In case special characters are required, these should be encoded using an appropriate encoding mechanism (e.g., HTML encoding) to ensure that these are not interpreted by the browser.
- Never insert untrusted data except in allowed Locations, instead.
  I. HTML escape before inserting untrusted data into HTML Element Content
  II. Attribute escape before inserting untrusted data into HTML Common Attributes
  III. JavaScript escape before inserting untrusted data into JavaScript Data Values
  IV. URL escape before inserting untrusted data into HTML URL Parameter Values

References:

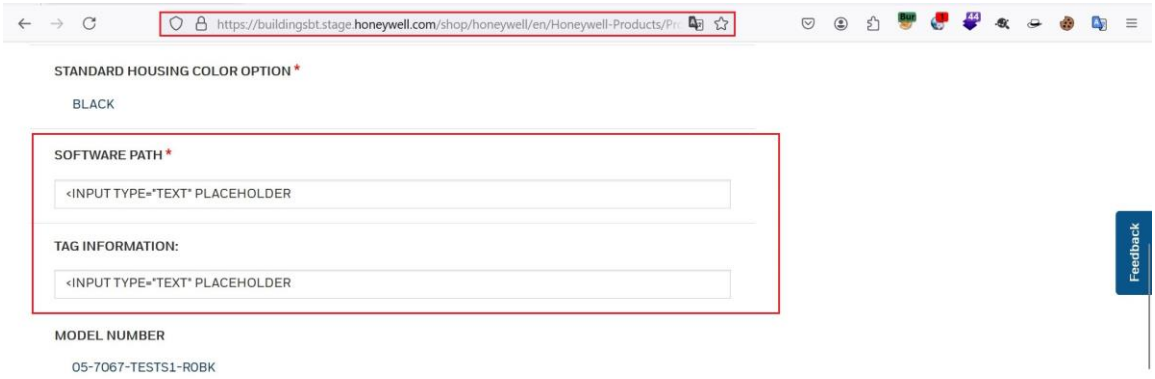
Industry Standard:
https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/11-Client-side_Testing/03-Testing_for_HTML_Injection

Steps to Reproduce:

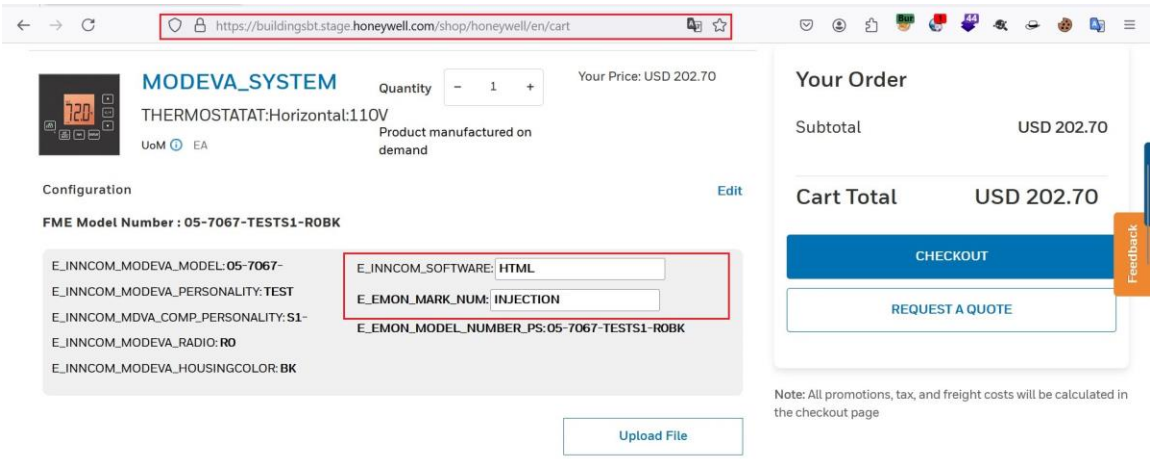
1. Login into the application
2. Search product (Ex: MODEVA_SYSTEM) & navigate to configurator Page.
3. File mandate fields > Enter the payload in ‘software path’, & click on update cart
4. Observe the payload is executed successfully.

Proof of Concept:

Below figure shows HTML injected input fields.



Below figure shows HTML code is injected successfully



3. Unvalidated Redirects

Unvalidated redirects and forwards are possible when a web application accepts untrusted input that could cause the web application to redirect the request to a URL contained within untrusted input. By modifying untrusted URL input to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials.

MEDIUM RISK

CVSS Score: **5.5**

OWASP Category: **A5**

Observation:

Noted that attackers could pass a malicious URL (for POC we used, www.bing.com) as the destination URL and redirect the users to a malicious/attacker's site.

Impact:

Such redirects may attempt to install malware or trick victims into disclosing passwords or other sensitive information. Unsafe forwards may allow access control bypass.

Recommendation:

- Safe use of redirects and forwards can be made in a few ways:
- Simply avoid using redirects and forwards.
- If used, do not allow the URL as user input for the destination. This can usually be done. In this case, you should have a method to validate a URL.
- Where possible, have the user provide short name, ID or token which is mapped server-side to a full target URL, this provides the highest degree of protection against the attack tampering with the URL, be careful that this does not introduce an enumeration vulnerability where a user could cycle through IDs to find all possible redirect targets
- If user input cannot be avoided, ensure that the supplied value is valid, appropriate for the application, and is authorized for the user.
- It is recommended that any such destination input be mapped to a value, rather than the actual URL or portion of the URL, and that server-side code translate this value to the target URL.
- Sanitize input by creating a list of trusted URLs (lists of hosts or a regex).
- Force all redirects to first go through a page notifying users that they are leaving your site and have them click a link to confirm.

References:

Industry Standard:
https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html

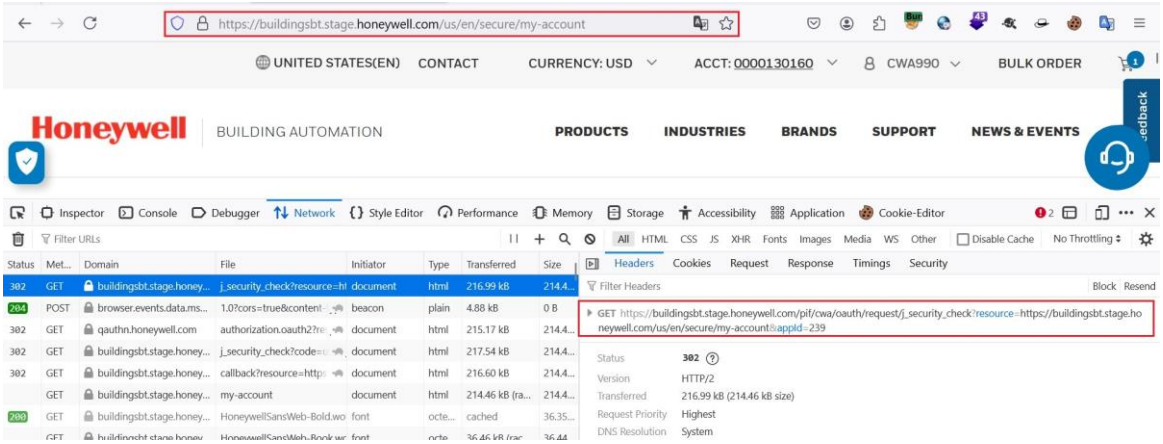
Steps to Reproduce:

1. Login into the application
2. Observe redirect URL in network tab using developer's tool, copy such URL
3. Browse the below modified resource and observe the response.

https://buildingsbt.stage.honeywell.com/pif/cwa/oauth/request/j_security_check?resource=https://bing.com&appId=239

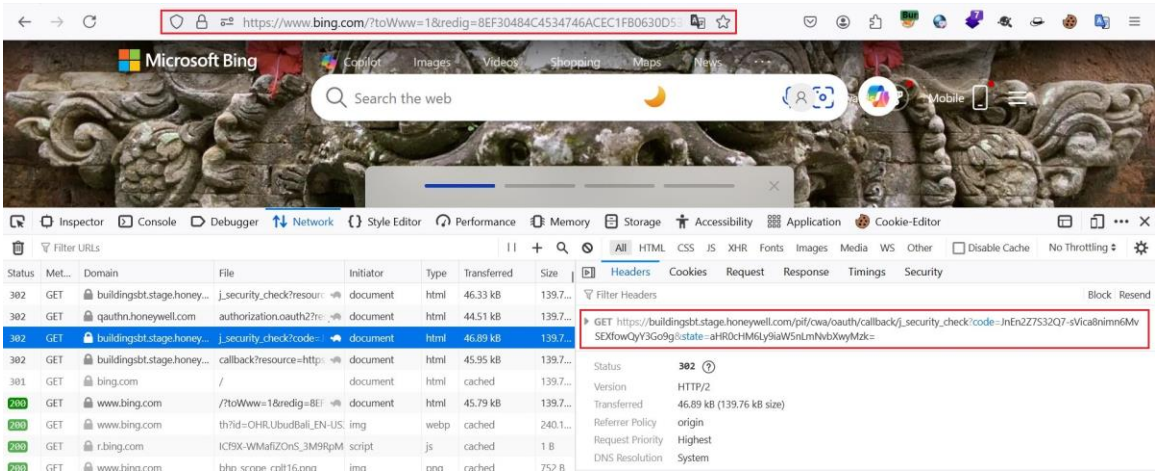
Proof of Concept:

Below figure shows that User logged in and redirect URL in network trace.



Below fig shows the URL with modified resource=attacker site (eg: https://www.bing.com/).

Below figure shows that user is redirected to modified site instead of actual site.



4. Improper Input Validation

MEDIUM RISK

CVSS Score: **5.4**

OWASP Category: **A3**

The most common security weakness is the failure to properly validate input coming from the client before using it. When an application does not validate input properly, an attacker can craft the input in a form that is not expected by the rest of the application. This will lead to parts of the application receiving unintended input, which may result in altered control flow, arbitrary control of a resource, or arbitrary code execution.

Observation:
Noted that the user input in this application not properly sanitized/validated. Potentially malicious characters are allowed in the input fields: '<>*^+%#!=~` []()

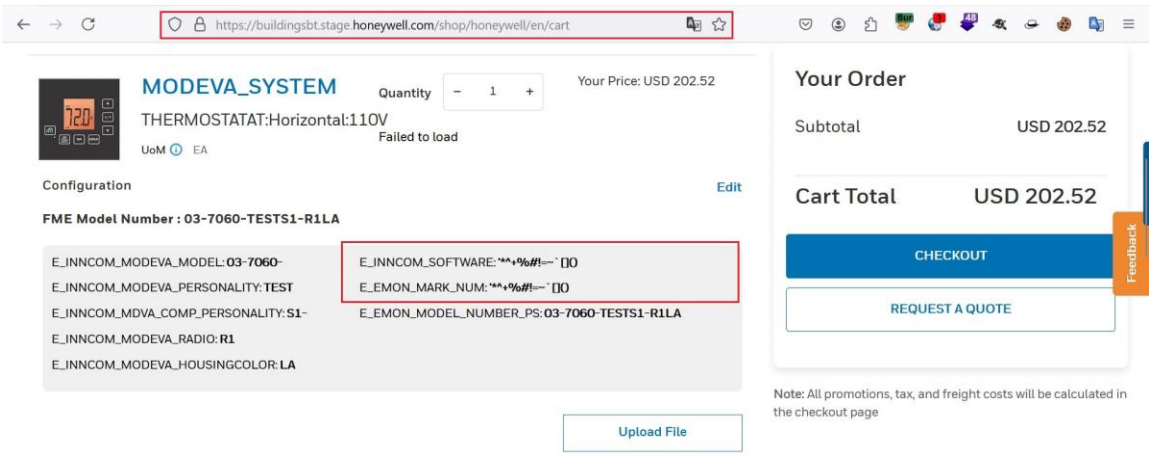
Impact:
Lack of input validation leads to almost all the major vulnerabilities such as XSS, SQL injection, interpreter injection, locale/Unicode attacks, file system attacks, and buffer overflows.

Recommendation:
- Allow special characters which are required for business in a whitelist - Use a whitelist filter with range, length and format, instead of a blacklist filter.
- Treat all parameters, objects, and other input data as untrusted.
- Check the request size and content length and type.
- Validate all inputs at the server, even if they are validated at the client.
- Identify special characters and escape them consistently during input validation.
- All user data controlled must be encoded when returned in the HTML page to prevent the execution of malicious data - Validate request parameters on the very first step, before it reaches to application logic.

References:
Industry Standard:
https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

Steps to Reproduce:
1. Login into the application
2. Search product (Ex: MODEVA_SYSTEM) & navigate to configurator Page.
3. Insert malicious characters into 'software path' in put fields.
4. Observe that malicious characters are successfully saved when click on update cart.

Proof of Concept:
Below figure shows malicious characters being inserted/displayed in the application.



6. Poor Error Handling

Improper handling of errors can introduce a variety of security problems for a web site. The most common problem is when detailed internal error messages such as stack traces, database dumps, and error codes are displayed to the user (hacker). These messages reveal implementation details that should never be revealed. Such details can provide hackers important clues on potential flaws in the application.

LOW RISK

CVSS Score: **1.8**

OWASP Category: **A2**

Observation:

Noted that the Application is displayed error message while sending the invalid string.

Impact:

Application error or warning messages may expose sensitive information about an application's internal workings to an attacker. Even when error messages do not provide a lot of detail, inconsistencies in such messages can still reveal important clues on how a site works, and what information is present under the covers.

Recommendation:

- Use a standard exception handling mechanism to be sure that your application properly handles all types of processing errors.
- All error messages sent to the user should contain only as little detail as necessary to explain what happened.
- Do not allow the application to throw errors up to the application container/the web application server.

References:

Industry Standard:
https://owasp.org/www-community/Improper_Error_Handling

Steps to Reproduce:

Browse the URL
https://buildingsbt.stage.honeywell.com/content/hbtbt/us/etc.txt
without authentication and observe the response

Proof of Concept:

Below figure shows the server information instead of a custom error message.

4. Unsafe HTTP Verbs Used

LOW RISK

CVSS Score: **1.8**

OWASP Category: **A5**

On most web servers, the default configuration includes insecure methods like TRACE, TRACK, PUT, DELETE, OPTIONS etc. Some of these methods potentially pose a security risk for a web application, as they can be used by an attacker for nefarious purposes like modifying files stored on the web server and steal the credentials of legitimate users.

Observation:
Noted insecure HTTP verbs are enabled.

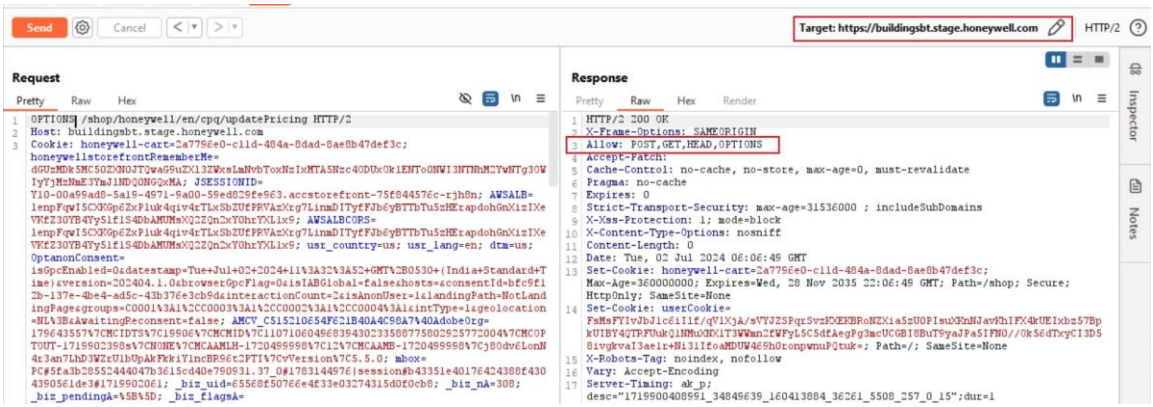
Impact:
An attacker can use these verbs to perform HTTP methods related attacks like Banner Grabbing, deleting files from the server or enumerating other HTTP methods enabled on the server.

Recommendation:
- Insecure HTTP method TRACE/TRACK, OPTIONS, DELETE, PUT should be disabled on the server. Only GET and POST methods are required in most of the applications.
- Ensure that only the required headers are allowed, and that the allowed headers are properly configured

References:
Industry Standard:
https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/06-Test_HTTP_Methods

Steps to Reproduce:
Capture the search request in any proxy tool and verify http methods.

Proof of Concept:
Below figure shows the unsafe HTTP verbs used in the web.



8. User Enumeration Possible

LOW RISK

CVSS Score: **1.8**

OWASP Category: **A5**

Username enumeration is a common application vulnerability which occurs when an attacker can determine if usernames are valid or not. Usually it occurs when a user-related form or application returns different results when a user exists than when no user exists.

Observation:
Noted that the application prompts an error like "This email address has already been registered. Please use a different email address or sign in with your existing email address" when user tries to re-register with an existing account, using which an attacker can determine if usernames exist/Valid or not.

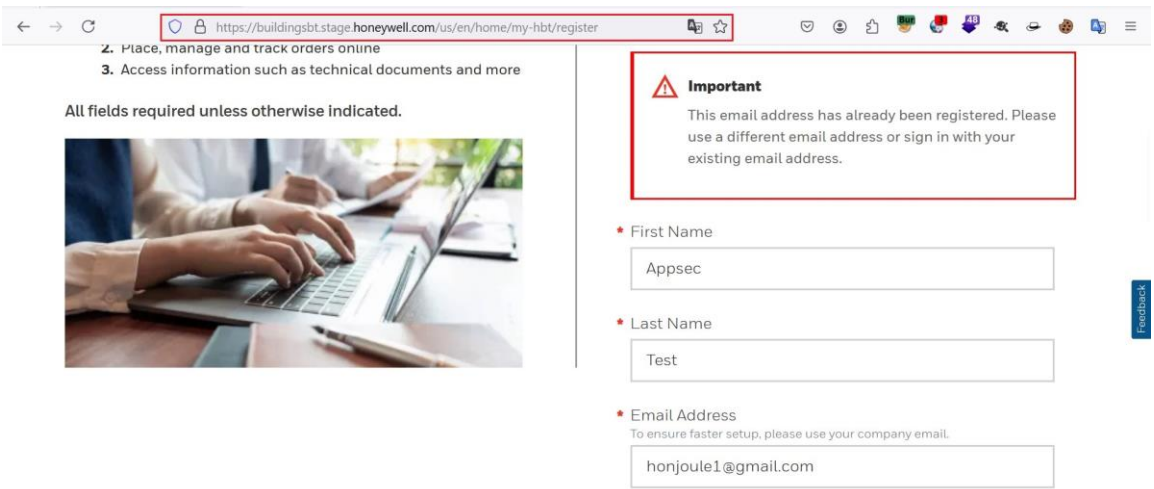
Impact:
User Enumeration leads to brute-force which can compromise the security of users. An attacker can steal the legitimate user account information and misuse it.

Recommendation:
Implement a custom message such as "Thankyou for registration. Upon verification you will receive and confirmation, look out for further information on your account."

References:
https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/03-Identity_Management_Testing/04-Testing_for_Account_Enumeration_and_Guessable_User_Account

Steps to Reproduce:
Create new account with existing email id and observe the response.

Proof of Concept:
Below fig shows the throws an error message.



7. Banner Grabbing

LOW RISK

CVSS Score: 1.8

OWASP Category: A2

Banner Grabbing is a technique used to gain information about a remote server. These types of issues are not exploitable in most cases but are considered as web application security issues because they allow malicious hackers to gather relevant information in order to achieve more than they could if they didn't get access to such information.

Observation:
Noted that the application displays server name information in the response.

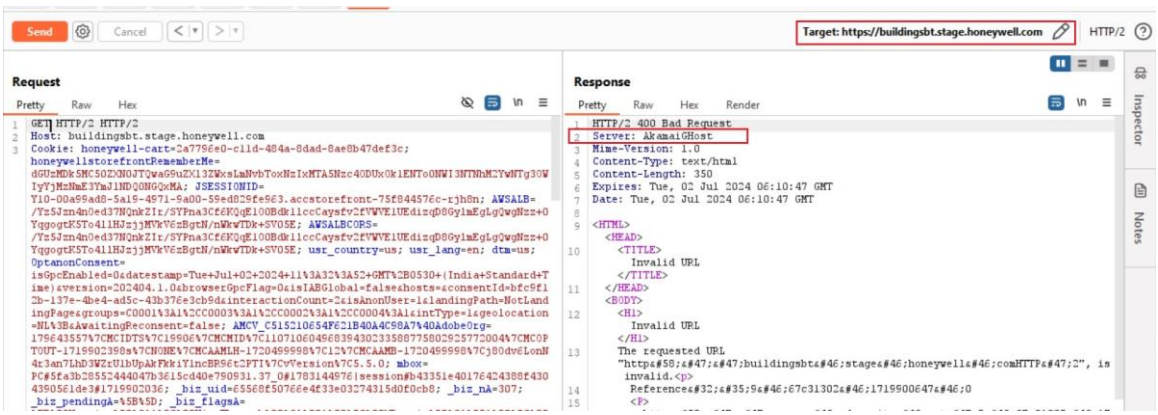
Impact:
Possible sensitive information disclosure, the attacker gets aware of the server details and can find some vulnerability linked to the versions of Server/ Technology. A dedicated attacker can use this information to find, and craft attacks specific to the system, or automated attacks may search for specific configurations w.r.t attack.

Recommendation:
- Mask the server/interface banner, Engine and Host name.
- Protect the presentation layer web server behind a hardened reverse proxy.
- Obfuscate the presentation layer web server headers.

References:
Industry Standard:
https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/01-Information_Gathering/02-Fingerprint_Web_Server
https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html

Steps to Reproduce:
Malform the URL and send the request in browser.

Proof of Concept:
Below fig shows that server name displayed as response



9. Insecure Cookie Configuration

Cookies are pieces of information stored on the client side that are primarily used for authentication & maintaining sessions. The below attributes, if set too loosely, then it could leave the application vulnerable to attacks by other applications on the same server. I.e., Secure, Path, HTTP Only & Expires.

LOW RISK

CVSS Score: **1.8**

OWASP Category: **A5**

Observation:

Noted that Path attribute is not configured in cookie parameters.

Impact:

A failure to specify proper attributes for cookies may result into stealing of cookie information through various attacks like Cross-Site Scripting (XSS) or a Man-In-The-Middle attack.

Recommendation:

- 'Path' attribute must be set to the directory in which the application is present such that the cookies are declared only valid for their respective app path.
- Set secure & HttpOnly flag to 'True' for all session cookies.
- Set a Same-Site attribute to a cookie by adding 'SameSite=Lax' or 'SameSite=Strict'.

References:

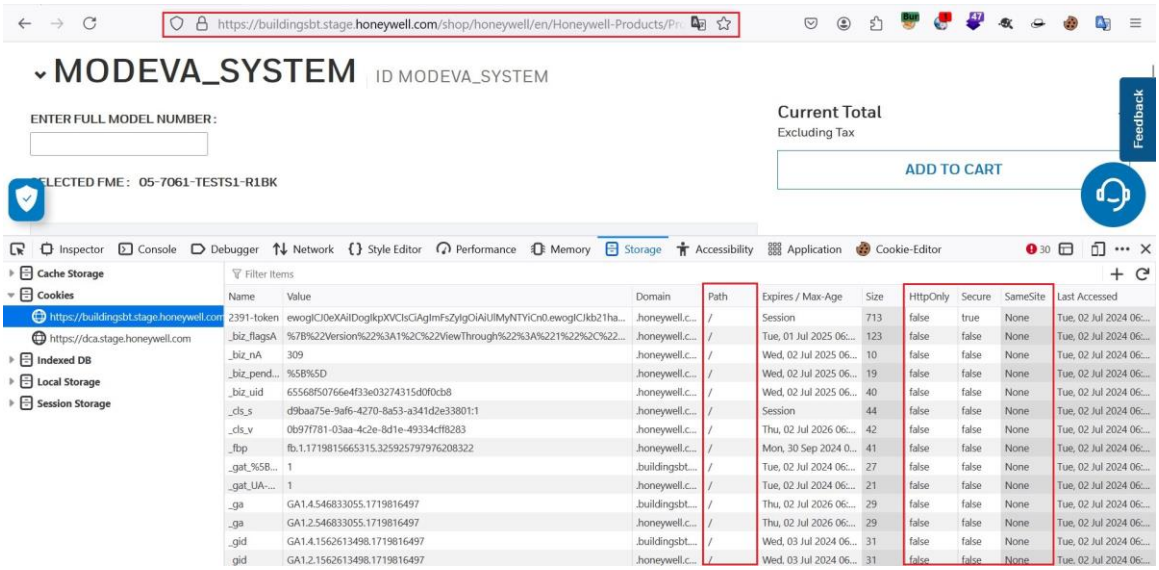
Industry Standard:
https://owasp.org/www-community/controls/SecureCookieAttribute

Steps to Reproduce:

Login to the application > Open Developer Tools. Observe cookie attributes.

Proof of Concept:

Below figure shows the Path attribute are not set in web app.



APPENDIX

Vulnerability Classification and Risk Rating:

To categorize vulnerabilities according to commonly understood vulnerability taxonomy, the **Enterprise Security Assurance** (ESA) team uses the industry standard **Common Weakness Enumerations** (CWE). CWE is a community-developed taxonomy of common security weaknesses which servers as a common language, a measuring stick for software security tools and as a baseline for weakness identification, mitigation, and prevention efforts.

To rate the severity of vulnerabilities, ESA uses the industry standard **Common Vulnerability Scoring System** (CVSS) to calculate severity for each identified security vulnerability. CVSS provides a way to capture the principal characteristics of a vulnerability, and produce a numerical score reflecting its severity, as well as textual representation of that score. To help prioritize vulnerabilities and assist vulnerability management process, ESA translates the numerical CVSS rating to a qualitative representation (such as Low, Medium, High and Critical).

Severity	CVSS rating	Description
CRITICAL	9.0 – 10	Issues that can be exploited in isolation, with no additional steps necessary, that may provide total compromise of the system.
HIGH	7.0 – 8.9	Severe issues that can easily be exploited to immediately impact the environment.
MEDIUM	4.0 – 6.9	Moderate security issues that require some effort to successfully impact the environment.
LOW	0.1 – 3.9	Security issues that have a limited or trivial impact to the environment.
INFO	0.0	These vulnerabilities significantly less risk and are informational in nature. These items can be remediated to increase security.

For more information:

Who We Are: https://go.honeywell.com/appsec

Security Assessment FAQs: https://go.honeywell.com/appsec-faqs

Secure Coding Best Practices: https://go.honeywell.com/appsec-bestpractices

OWASP Web Application Top 10 Risks: https://owasp.org/www-project-top-ten/

OWASP API Top 10 Risks: https://owasp.org/www-project-api-security/

CWE: https://cwe.mitre.org/

CVSS: https://www.first.org/cvss/