

Sri Lanka Institute of Information Technology



IT21299902 (ZAKY M.S.M.A)

Bug bounty Report 03.

Domain: MalwareBytes.com

Web security – IE2062

B.Sc. (Hons) in Information Technology Specialization in
cyber security.

Declaration:

- I hereby certify that no part of this assignment has been copied from any other work or any other source. No part of this assignment has been written/produced for me by another person.
- I hold a copy of this assignment that I can produce if the original is lost or damaged.

1. Using components with known vulnerabilities.

- Possible BREACH attack.

I hope this message finds you well. I am writing to report a vulnerability I discovered under the domain of <https://www.malwarebytes.com/business/pricing?quantity=%2527> during my participation in the bug bounty program. Please find below the details of the vulnerability for your review and further action.

 [\[Possible\] BREACH Attack Detected](#) GET <https://www.malwarebytes.com/business/pricing?quantity=%2527> 7 MEDIUM

Vulnerability title: Possible BREACH attack on pricing page

Severity: Medium

OWASP classification 2013: A9

CVSS 3.0 score: 6.5

CVSS 3.1 score: 6.5

CVSS string: CVSS:3.1/AV: N/AC: L/PR: N/UI: R/S: U/C: H/I: N/A: N

Effected components: User accounts.

Server infrastructure.

Data confidentiality.

Date of Discovery: 2023/05/10

Date of Report: 2023/05/12

2. Vulnerability Description.

If a BREACH attack is successful, the attacker can potentially access sensitive information transmitted over an encrypted connection. I have identified a possibility of BREACH attack within the affected system. If this attack succeeds in the system it will lead to data leakage, session hijacking, confidentiality breach, and data manipulation.

3. Impact assessment.

Under this OWASP category I have identified many vulnerabilities. One detected a BREACH attack is possible in this web application. It is under the domain of

<https://www.malwarebytes.com/business/pricing?quantity=%2527>.

The result of this vulnerability is that an attacker may be able to use this online application to undertake a BREACH (Browser Reconnaissance & Exfiltration through Adaptive Compression of Hypertext) attack.

It is still possible to carry out this type of attack and to breach the data from the online application due to factors that make the BREACH attack conceivable on this web site, even if it is secured with SSL/TLS protected communication.

1. Information leakage: If a BREACH attack is successful, encrypted HTTPS answers may reveal private data. This includes any private data transmitted within the compressed response body, such as CSRF tokens, session IDs, authentication credentials, and others.
2. Compression feature of HTTP exploited: BREACH makes use of the compression capability of HTTP to lower the size of the response body. An attacker can determine whether certain information is present in the answer by altering the content and tracking changes in the compressed response size.
3. Side-channel attack vector: The BREACH attack uses numerous queries to track changes in the size of compressed answers as a side-channel attack vector. An attacker can obtain sensitive information by monitoring the variations in the compressed response sizes and conducting controlled queries while evaluating these changes.

Because of this vulnerability marked as **MEDIUM**. It's a must to address it with a good solution.

4. Steps to Reproduce.

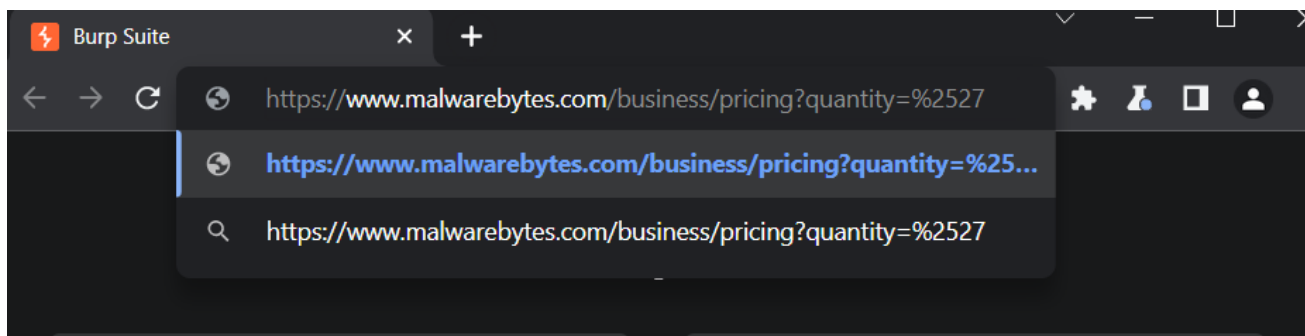
Reproduction of this vulnerability must be done in a controlled environment and at your own risk of anything happens.





1. First need to Setup network traffic intercepting tool like burp suite to capture the request and response to such an attack successfully.
2. Determine which pages that send sensitive data, such as CSRF tokens, session IDs, or other private information. In this case I have identified the page **pricing** sends sensitive data in unencrypted manner. (<https://www.malwarebytes.com/business/pricing>)
3. Use a tool or write a script to send carefully constructed requests to the target pages. Change the requests to include patterns or secrets you think might be hidden in the compressed answers.
4. Take note of the compressed response size for each well-constructed request. Keep an eye out for any variations in response size when the request's content or secrets are changed.
5. Compare the response times for various queries with different secret or content variants. Observe any patterns or noteworthy modifications in the compressed answer sizes.

6. Compare the response times for various queries with different secret or content variants. Observe any patterns or noteworthy modifications in the compressed answer sizes.
7. Repeat the procedure to further validate the vulnerability by adjusting the created requests or adding new variations. Analyze the variations in response size and check for consistency in your observations.
8. Verify the correlation between changes in response size and the presence or absence of sensitive information. Check to see if the changes in compressed response size can be reliably seen.

5. Proof of concept.

Using burp suite



Name	Value	
quantity	%27	>
		   

Using net sparker

Method	Parameter	Value
GET	quantity	%27

Reflected Parameter(s)

- quantity

Sensitive Keyword(s)

- token,nonce

Request

```
GET /business/pricing?quantity=%2527 HTTP/1.1
Host: try.malwarebytes.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-us,en;q=0.5
Cache-Control: no-cache
Connection: Keep-Alive
Cookie: global_variables.user.type=eyJpc0J1c2luZXNzU21hbGwiOnRydWUsImVzaW5lc3NMYXJnZSI6ZmFsc2UsImVzaW5lc3MiOnRydWUsImVzaW5lc3VtZXIiOmZhbHNlfQ%3D%3D; global_variables.user.type=eyJpc0J1c2luZXNzU21hbGwiOnRydWUsImVzaW5lc3NMYXJnZSI6ZmFsc2UsImVzaW5lc3MiOnRydWUsImVzaW5lc3VtZXIiOmZhbHNlfQ%3D%3D; over100=false; over100=false; visited=true
User-Agent: Mozilla/5.0 (Windows NT 10.0; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36
X-Scanner: Netsparker
```

Response

Response Time (ms) : 2148.3101 Total Bytes Received : 131583 Body Length : 130663 Is Compressed : No

```
HTTP/1.1 200 OK
set-cookie: ubvs=296d5623-e43a-4a0b-ab9e-6d28a5104fa1; Max-Age=15552000; Path=/; SameSite=Lax
set-cookie: ubvt=v2%7C296d5623-e43a-4a0b-ab9e-6d28a5104fa1%7Cfcbe530a-2a05-4c25-8124-080cc3626596%3Ab%3A
single; Max-Age=259200; Domain=malwarebytes.com; Path=/; SameSite=Lax
set-cookie: ubpv=b%2Cfcbe530a-2a05-4c25-8124-080cc3626596; Max-Age=15897600; Path=/business/pricing/10-99-devices/; SameSite=Lax
x-unbounce-visitorid: 296d5623-e43a-4a0b-ab9e-6d28a5104fa1
link: <https://try.malwarebytes.com/business/pricing/10-99-devices/>; rel="canonical"
x-unbounce-pageid: fcbe530a-2a05-4c25-8124-080cc3626596
x-proxy-backend: page-server
x-unbounce-variant: b
content-length: 17394
content-location: https://try.malwarebytes.com/business/pricing/10-99-devices/
content-type: text/html; charset=utf-8
content-encoding:
date: Mon, 15 May 2023 10:56:25 GMT
etag: "b:296d5623e43a4a0bab9e6d28a5104fa1"

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"><html xmlns="http://www.w3.org/1999/xhtml"><head><META http-equiv="Content-Type" content="text/html; charset=UTF-8" >
<!--fcbe530a-2a05-4c25-8124-080cc3626596 b-->

<title>Pricing: 10-99 Device Support Malwarebytes</title>
<meta name="keywords" content="">
<meta name="description" content="Find out pricing options for cyber security protection for businesses with 10-99 endpoints.">
```


6. Proposed mitigation/fix.

1. If possible, disable HTTP level compression.
2. Separate sensitive information from user input
3. Protect vulnerable pages with CSRF token. The Same Site Cookie attribute will mitigate this issue, because to exploit this issue an attacker forces the victim to visit a target website using invisible frames. With the Same Site cookie attribute added, cookies that belong to the target won't be sent with a request that does not include top level navigation.
4. Hide the length of the traffic by adding a random number of bytes to the responses.
5. Add in a rate limit, so that the page maximum is reached five times per minute.
6. Implement per-session, randomized CSRF tokens to stop attackers from successfully launching CSRF attacks. To defend against CSRF attacks, think about employing double-submit cookies or other secure techniques.
7. To defend against additional attack routes that could indirectly expose data, like SQL injection and Cross-Site Scripting (XSS), provide appropriate input validation and output encoding.
8. To identify any unusual behavior or anomalies relating to compression, response times, or data leakage, implement monitoring and intrusion detection systems.