AARDWOLF SECURITY

WEB APPLICATION PENETRATION TEST REPORT FOR TEAMFLECT LTD

Version 1.1

01908 733540

william.fieldhouse@aardwolfsecurity.com

Aardwolf Security Ltd
Midsummer Court
314 Midsummer Boulevard
Milton Keynes
Buckinghamshire
MK9 2UB

Author: William Fieldhouse

Date: 30th April 2024

CONTENTS

CONTACT DETAILS **2**
DOCUMENT VERSION **3**
SCOPE **4**
RETEST EXECUTIVE SUMMARY **5**
EXECUTIVE SUMMARY **5**
KEY RECOMMENDATIONS 6
RETEST VULNERABILITY COUNT **7**
VULNERABILITY COUNT **7**
VULNERABILITY LIST **8**
TECHNICAL FINDINGS **10**
Stored Cross Site Scripting (XSS) - Fixed 10
Insecure Upload Functionality – Partially Fixed 20
HTML Injection in Email Template - Fixed 24
Software Version Numbers Revealed - Accepted 28
Enabled CBC Mode Ciphers Detected - Accepted 30
Cross-Origin Resource Sharing: Arbitrary Origin Trusted - Not Fixed 33
Cookie Without HttpOnly Flag Set - Fixed 35
Missing HTTP Security Headers – Partially Fixed 37
Cross-Domain Script Include - Accepted 40
Concurrent Logins Permitted -Accepted 42
Environment Information Disclosure - Accepted 44
Unencrypted Network Service - Accepted 46
APPENDIX A – OWASP TOP 10 VULNERABILITIES **48**
APPENDIX B – NMAP TCP **50**
APPENDIX C – NMAP UDP **51**

CONTACT DETAILS

Customer Contact Information	
Contact Name:	Bora Unlu
Title:	Director
E-mail:	bora@teamflect.com

Consultant Information	
Contact Name:	William Fieldhouse
Title:	Director / Senior Penetration Tester
Email:	william.fieldhouse@aardwolfsecurity.com

Consultant Information	
Contact Name:	Dave Shanahan
Title:	Senior Penetration Tester
Email:	dave.shanahan@aardwolfsecurity.com

DOCUMENT VERSION

Version	Date	Consultant	Role	Comments
0.1	18.01.2024	William Fieldhouse	Senior Penetration Tester	Initial draft
0.2	19.01.2024	Dave Shanahan	Sally Phillips	Quality assurance
1.0	22.01.2024	William Fieldhouse	Senior Penetration Tester	First release
1.1	30.04.2024	William Fieldhouse	Senior Penetration Tester	Retest

SCOPE

Teamflect mandated Aardwolf Security Ltd (AS) to perform penetration testing on the following production items in scope:

- Web application –
  - app.teamflect.com
  - admin.teamflect.com

- API –
  - api.teamflect.com

- Network –
  - 20.50.153.39
  - 13.107.213.64

The purpose of the assessment was to improve the security posture of Teamflect's assets and identify weak spots within the external network, web application and associated API. For web applications AS use a custom developed testing methodology which is based upon the OWASP testing guidelines.

RETEST EXECUTIVE SUMMARY

The retest has revealed significant progress was made in addressing major vulnerabilities. Teamflect have successfully remediated high and medium risk issues such as Stored Cross-Site Scripting (XSS) and HTML Injection in email templates. Furthermore, the security of the web application has been enhanced by successfully implementing HttpOnly flags on cookies, which mitigates the risk of client-side scripts accessing protected cookies.

However, some areas, such as the Insecure Upload Functionality and the implementation of HTTP Security Headers, have only seen partial improvements. These components still require further attention to fully secure the application.

Further to the above one unresolved issue that remains is the application's Cross-Origin Resource Sharing (CORS) policy, which continues to trust arbitrary origins.

Several lower-priority risks, including the exposure of software version numbers and the use of CBC mode ciphers have been accepted after assessing operational needs against potential security threats. Likewise, risks related to environmental information disclosure and unencrypted network services are deemed acceptable under current operational policies.

EXECUTIVE SUMMARY

Aardwolf Security (AS) were approached by Teamflect Ltd to perform a penetration test in accordance with the scope outlined above (see Scope Section). Testing was conducted between the 15th and 18th of January 2024. The following report details the issues identified during this phase of the overall assessment.

At the conclusion of testing AS identified one high, three medium, six low and three informational risk issues across the network and web application.

The issues noted have been categorised according to the OWASP (Open Web Application Security Project) standards, aiding in understanding the nature and potential impact of each vulnerability. Below is a detailed overview of these findings, along with recommendations for remediation.

Injection Flaws: Injection flaws are a prevalent and hazardous type of vulnerability that can lead to significant security breaches. In this category, it was noted that several issues affect the web application. A high-impact Stored Cross-Site Scripting (XSS) vulnerability poses a substantial threat to the application's security. Additionally, medium-impact vulnerabilities like HTML Injection in Email Template and CSV/Formula Injection were discovered, each presenting its own risks to data integrity and application security.

Security Misconfiguration: Security misconfigurations can expose applications to various threats. In this category, it was observed that a range of issues exist, including a medium-impact Insecure Upload Functionality vulnerability and several low-impact vulnerabilities such as Revealed Software Version Numbers, Enabled CBC Mode Ciphers, and Arbitrary Origin Trusted in Cross-Origin Resource Sharing. These findings highlight the importance of proper configuration and regular security reviews to prevent potential exploits in the application.

Each category of vulnerabilities presents specific risks that need to be addressed to enhance the overall security of the digital assets. It is recommended to prioritise the remediation of high and medium impact issues, followed by the low impact ones.

KEY RECOMMENDATIONS

It is recommended that the following actions be taken to increase the security of the network and web application:

- Implement input validation and encoding to prevent XSS attacks.
- Enforce file type validation for uploads.
- Sanitise user input in email templates.
- Validate and escape data in CSV exports.
- Obfuscate software version numbers on public interfaces.
- Replace CBC mode ciphers with secure alternatives.
- Limit Cross-Origin Resource Sharing to trusted domains.
- Set HttpOnly flag on cookies to prevent XSS.
- Add essential HTTP security headers for enhanced protection.

RETEST VULNERABILITY COUNT

Critical	High	Medium	Low	Informational
0	0	1	5	3

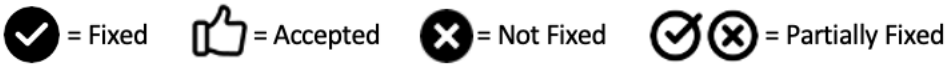
VULNERABILITY COUNT

Critical	High	Medium	Low	Informational
0	1	2	6	3

VULNERABILITY LIST

ID	Vulnerability	Risk	Status
ID-001	Stored Cross Site Scripting (XSS)	High	
ID-002	Insecure Upload Functionality	Medium	
ID-003	HTML Injection in Email Template	Medium	
ID-004	Software Version Numbers Revealed	Low	
ID-005	Enabled CBC Mode Ciphers Detected	Low	
ID-006	Cross-Origin Resource Sharing: Arbitrary Origin Trusted	Low	
ID-007	Cookie Without HttpOnly Flag Set	Low	
ID-008	Missing HTTP Security Headers	Low	
ID-009	Cross-Domain Script Include	Low	

ID-010	Concurrent Logins Permitted	 Informational	
ID-011	Environment Information Disclosure	 Informational	
ID-012	Unencrypted Network Service	 Informational	



TECHNICAL FINDINGS

Stored Cross Site Scripting (XSS) - Fixed

ID-001
Risk: High
CVSS Score: 7.3

CVSS Version 3.1 Metrics

Attack Vector: **Network**	Scope: **Unchanged**	Attack Complexity: **Low**
Confidentiality: **High**	Required Privileges: **Low**	Integrity: **High**
User Interaction: **Required**	Availability: **None**	

Description

A Stored Cross-Site Scripting (XSS) attack is a type of security vulnerability that occurs when an application allows an attacker to inject malicious scripts into web pages that are then stored and later served to other users. The term stored indicates that the malicious script is persisted on the target server, as opposed to being reflected in real-time.

When paired with a session cookie that does not have the HttpOnly flag enabled, the impact of a Stored XSS attack is magnified. The HttpOnly flag is a security attribute that can be set on a cookie to prevent it from being accessed through client-side scripts, including JavaScript. If a session cookie lacks the HttpOnly flag and a Stored XSS vulnerability is exploited then account hijacking could be possible.

Example

Multiple instances of stored XSS were discovered that appeared to have the same root cause. In each of the attack scenarios listed, the affected field was linked to rich text functionality. Whilst the name of the vulnerable parameter differed, its value always contained a configuration hash:

```
{
  "goalId":"65a543f27187888523abd7d7",
  "comment":
  "{\"blocks\":[{\"key\":\"63c2l\",\"text\":\"This is a comment\",\"type\":\"
  unstyled\",\"depth\":0,\"inlineStyleRanges\":[],\"entityRanges\":[],\"data\
  ":{}}],\"entityMap\":{}}",
  "textComment":"This is a comment",
  "mentionedUsers":[
  ],
  "attachments":[
  ]
}
```


Figure 1 - Example of configuration hash.

Feedback Functionality

In this attack, an end user (*Molly Dennis*) gave feedback to an administrator (*Henry Bobinny*).

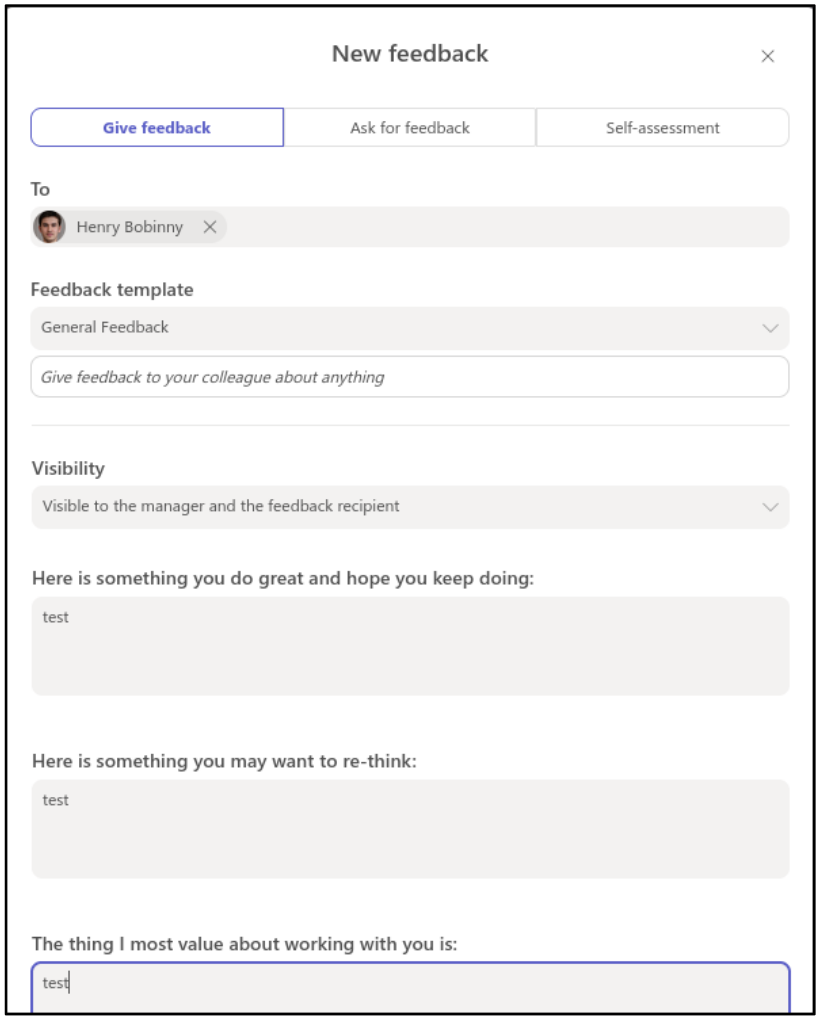


Figure 2 - The target administrator and the General Feedback template were selected.

When submitting this form, an HTTP request was generated against the API's /api/*feedback/sendInstantFeedback* endpoint. Within this request, the *description* parameter was vulnerable to XSS.



Figure 3 - An XSS payload inserted into the description parameter

After this feedback was submitted, the administrator received a notification stating Molly Dennis had sent them feedback.



Figure 4 - Notification alerting the administrator of new feedback

The DOM of the adminstrator's browser contained the XSS payload within the feedback modal.

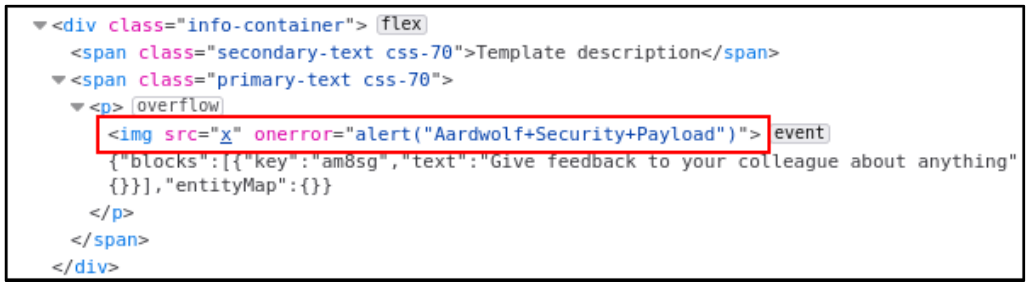


Figure 5 - Browser's DOM processing the XSS payload

Upon opening the feedback modal, the XSS payload would trigger, attacking the administrator.

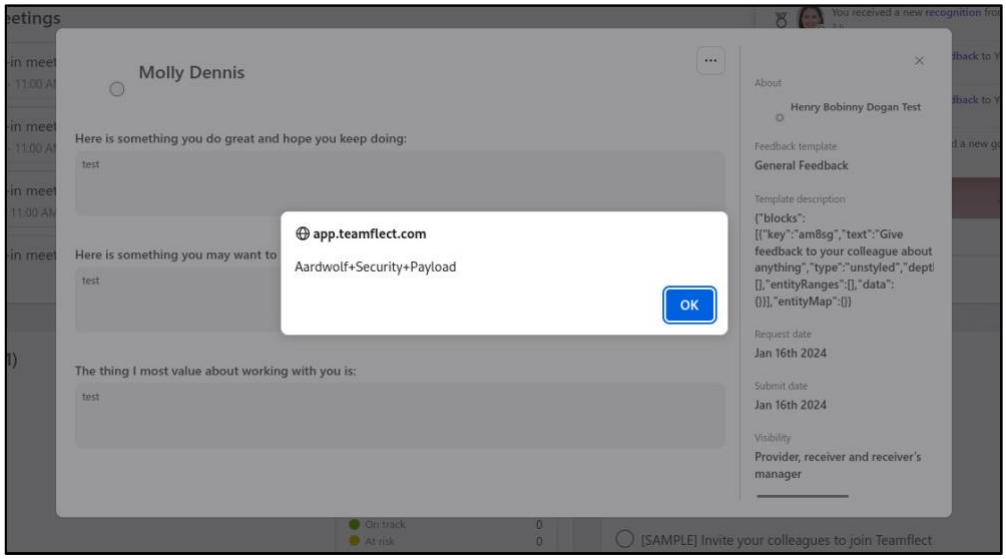


Figure 6 - Benign XSS payload exploiting the administrator's browser

The impact of this instance was magnified as it was possible to send feedback to users outside the attacker's organisation. Therefore, permitting account hijacking between organisations.

Goal Functionality

In this demonstration, the *Goal* functionality could be exploited to perform an XSS attack.

After creating a *Goal,* an attacker could opt to edit the *Goal*. This would produce a call to the */api/goal/editGoal* API endpoint to update the *Goal*. The *description* field was vulnerable to XSS so a benign XSS payload was inserted that simply displayed a JavaScript alert box.



Figure 7 - Edit Goal HTTP Request

The response of this request displayed the XSS payload:



Figure 8 - Edit Goal HTTP Response

Upon selecting the *Goal* in the application's UI, the user's browser processed the XSS payload inserted it into the DOM:



Figure 9 - Browser DOM contains XSS Payload

When a user visited this *Goal*, the XSS payload would be executed in their browser:

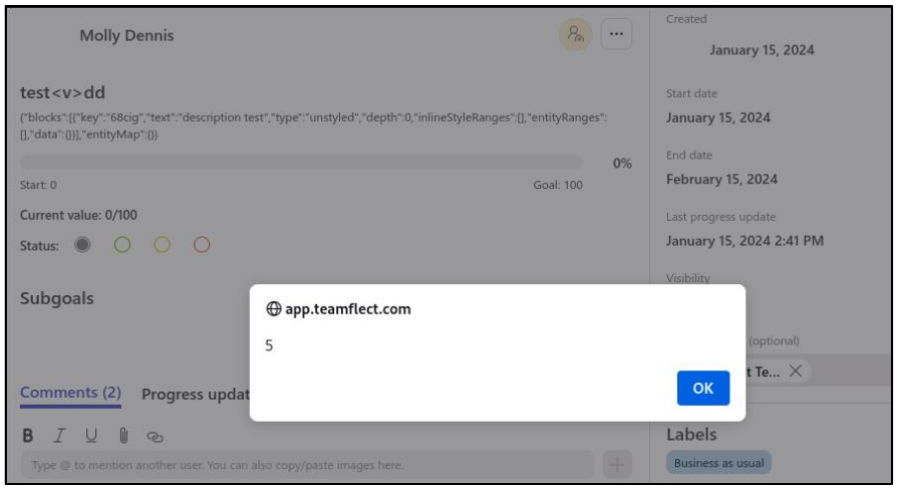


Figure 10 - The XSS Payload being rendered in the User's browser

Goal Comment

The *comment* section within a *Goal* was also found to suffer from the same vulnerability. When submitting a *comment*, a call to the API method */api/goal/addComment* was performed and contained the vulnerable parameter named *comment*:



Figure 11 - Add comment HTTP request

The browser's DOM for this comment would then process the XSS payload that was introduced to the application:

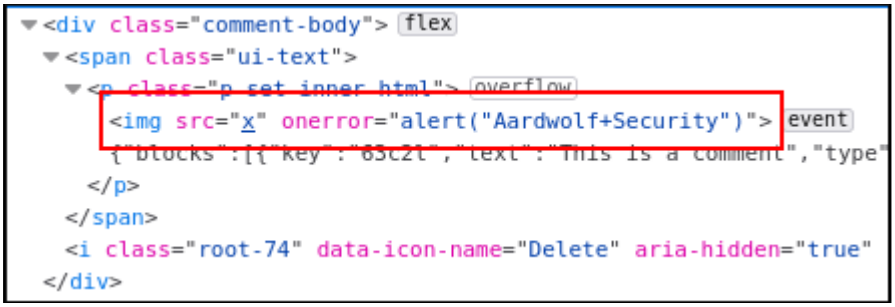


Figure 12 - XSS payload processed by the browser's DOM

When the user visited this comment, the payload would execute and render in their browser.

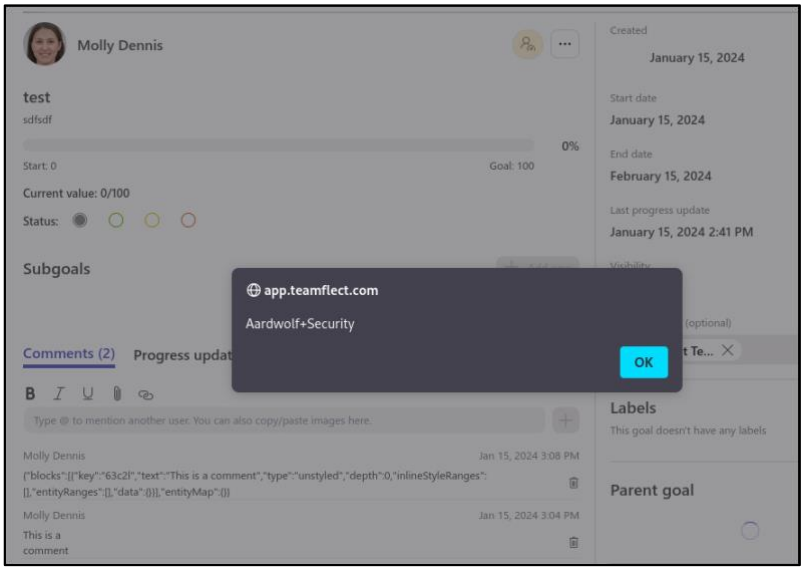


Figure 13 - Victim's browser executing the XSS payload

Admin Section

The same issues were present within the rich text editor within the admin section as shown below:

Request -

Request

Pretty Raw Hex \n

```
1  POST /api/admin/updateCompetency HTTP/2
2  Host: api.teamflect.com
3  Content-Length: 2257
4  Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
5  Accept: application/json, text/plain, */*
6  Content-Type: application/json
7  Sec-Ch-Ua-Mobile: ?0
8  Authorization: Bearer
   eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NWE1NTcxZDcxODc4ODg1MjNhZjI5YTYiLCJpYXQi
   OjE3MDU0MjIwOTAsImV4cCI6MTcwNTUwODQ5MCwiaXNzIjoiaHR0cHM6Ly90ZWFtZmxlY3QuY29tIn0.5AhEh9FcS
   zqYFYq-lsgXp_t-EZWgU9B8SAq-pe_VE7I
9  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
   Gecko) Chrome/120.0.6099.199 Safari/537.36
10 Sec-Ch-Ua-Platform: "macOS"
11 Origin: https://admin.teamflect.com
12 Sec-Fetch-Site: same-site
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
17 Priority: u=1, i
18
19 {
     "name":"Problem-Solving",
     "description":
     "<img src=x onerror=alert(1)>{\"blocks\":[{\"key\":\"9qah1\",\"text\":\"{\\\"blocks\\\"
     :[{\\\"key\\\":\\\"bigk4\\\",\\\"text\\\":\\\"Employees should possess basic problem-so
     lving skills, identifying and resolving straightforward issues within their roles.aaaa\
     \",\\\"type\\\":\\\"unstyled\\\",\\\"depth\\\":0,\\\"inlineStyleRanges\\\":[],\\\"enti
     tyRanges\\\":[],\\\"data\\\":{}}],\\\"entityMap\\\":{}}\",\"type\":\"unstyled\",\"depth
     \":0,\"inlineStyleRanges\":[],\"entityRanges\":[],\"data\":{}}],\"entityMap\":{}}",
     "level":0,
     "expectation":
     "<img src=x onerror=alert(2)>{\"blocks\":[{\"key\":\"95d91\",\"text\":\"{\\\"blocks\\\"
     :[{\\\"key\\\":\\\"92ed\\\",\\\"text\\\":\\\"Seek guidance when needed, follow establis
     hed processes, and contribute ideas for improvement.aaa\\\",\\\"type\\\":\\\"unstyled\\
     \",\\\"depth\\\":0,\\\"inlineStyleRanges\\\":[],\\\"entityRanges\\\":[],\\\"data\\\":{}
     }],\\\"entityMap\\\":{}}\",\"type\":\"unstyled\",\"depth\":0,\"inlineStyleRanges\":[],\
     "entityRanges\":[],\"data\":{}}],\"entityMap\":{}}",
     "_id":"65a5571d7187888523af290b",
     "tenantId":"f5755362-786a-4827-8212-9c6c520c8903",
     "createdAt":"2024-01-15T16:02:37.153Z",
```

Figure 14 - XSS input into the competency and expectations fields

Response -
https://admin.teamflect.com/#/career/competencylibrary

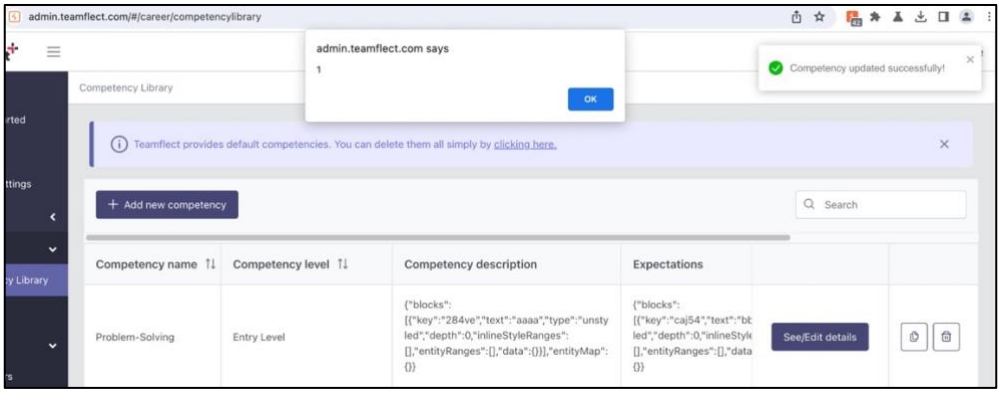


Figure 15 - XSS executes once the details above are saved

Remediation

The following is recommended to remediate XSS vulnerabilities:

- Never trust user input
- Never insert untrusted data except in allowed locations
- HTML escape before inserting untrusted data into HTML element content
- Use whitelists in place for Black lists for input filtering

Affected Hosts

URL/IP Address	Port
app.teamflect.com	443
admin.teamflect.com	443

References

https://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29

Retest Notes

The issue has been remedied with the input showing as being stripped from the request:

Request –

```
POST /api/request/addNewRequest HTTP/2
Host: api.teamflect.com
Content-Length: 1216
Sec-Ch-Ua: "Not-A.Brand";v="99", "Chromium";v="124"
Language: en
Sec-Ch-Ua-Mobile: ?0
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NWVlY2M0YTViZjYyMDVkOTMwYT
```

```
g3OTAiLCJ0ZW5hbnRJZCI6IjFlMWYyMzRhLTE0MzQtNGVlNy04M2U1LTZiOTBkNmM2MDBhMCIsI
mlhdCI6MTcxNDQxNTQ4NSwiZXhwIjoxNzE0NTAxODg1LCJpc3MiOiJodHRwczovL3RlYW1mbGVj
dC5jb20ifQ.RsQbxLhpGWP6OIQg42F2TaIq_5W6iiUPnH1rAfrf6VY
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/124.0.6367.60 Safari/537.36
Content-Type: application/json
Accept: application/json, text/plain, */*
Sec-Ch-Ua-Platform: "macOS"
Origin: https://app.teamflect.com
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Priority: u=1, i

{"title":"aaaa","description":"<img src =q
onerror=prompt(8)>{\"blocks\":[{\"key\":\"67ovd\",\"text\":\"bbbbb\",\"type
\":\"unstyled\",\"depth\":0,\"inlineStyleRanges\":[],\"entityRanges\":[],\"
data\":{}}],\"entityMap\":{}}","dueDate":null,"selectedUsers":[{"id":"a48bc
592-3ec9-4f75-87ea-
54d8fdd4d53f","key":"AdeleV@teamblee.onmicrosoft.com","profilePic":"blob:ht
tps://app.teamflect.com/54b626ea-059e-4baa-88ac-b93d1f905f7e","text":"Adele
Vance
(AdeleV@teamblee.onmicrosoft.com)","department":"Retail","country":"United
States","birthday":"1990-08-17","employeeHireDate":"2023-02-
28","officeLocation":"Denver Office","secondaryText":"Retail
Manager","userPrincipalName":"AdeleV@teamblee.onmicrosoft.com","userRelatio
n":{"hasNotSignedUp":true},"mail":"AdeleV@teamblee.onmicrosoft.com","displa
yName":"Adele
Vance","isNotAllowed":false,"imageUrl":"blob:https://app.teamflect.com/54b6
26ea-059e-4baa-88ac-
b93d1f905f7e"}],"relatedEmail":null,"isIndividual":true,"addedFrom":"taskPa
ge","relatedGoalId":"","isRecurring":false,"createFirstTask":true,"descript
ionAttachments":[],"notificationDays":[],"weekFrequency":1,"notificationHou
r":19,"notificationTimeZone":1,"recurringDueDay":1,"isLockForEditing":false
,"checkLists":[]}
```

Response –

```
HTTP/2 200 OK
Date: Mon, 29 Apr 2024 18:36:45 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 1216
Access-Control-Allow-Origin: *
Etag: W/"4c0-+od8UIf3g8xQEx8Os3bokTDrBPA"
Request-Context: appId=cid-v1:
X-Azure-Ref: 20240429T183645Z-
17fd566d545lwwlp72kd9xathn000000072000000000fcb3
X-Cache: CONFIG_NOCACHE
Accept-Ranges: bytes

{"tenantId":"1e1f234a-1434-4ee7-83e5-6b90d6c600a0","createdAt":"2024-04-
29T18:36:45.181Z","dueDate":null,"status":1,"title":"aaaa","description":"{
\"blocks\":[{\"key\":\"67ovd\",\"text\":\"bbbbb\",\"type\":\"unstyled\",\"d
epth\":0,\"inlineStyleRanges\":[],\"entityRanges\":[],\"data\":{}}],\"entit
yMap\":{}}","descriptionText":"bbbbb","checkList":[],"assignedBy":{"id":"9f
d850b8-ad24-4a53-a9e7-8e3b3c1d15ff","displayName":"Miriam
Graham","userPrincipalName":"MiriamG@teamblee.onmicrosoft.com","department"
:"Sales &
```

```
Marketing","jobTitle":"Director"},"assignedTo":[{"createdAt":"2024-04-
29T18:36:45.181Z","status":1,"user":{"oid":"a48bc592-3ec9-4f75-87ea-
54d8fdd4d53f","displayName":"Adele
Vance","userPrincipalName":"AdeleV@teamblee.onmicrosoft.com","mail":"AdeleV
@teamblee.onmicrosoft.com","jobTitle":"Retail
Manager","department":"Retail"},"isDeleted":false,"checkList":[],"_id":"662
fe8bde01f5619261e87bd","individualComments":[]}],"sharedComments":[],"relat
edEmail":null,"isIndividual":true,"isChecked":false,"addedFrom":"taskPage",
"attachments":[],"descriptionAttachments":[],"isRecurring":false,"labels":[
],"isLockForEditing":false,"_id":"662fe8bde01f5619261e87bc","updatedAt":"20
24-04-29T18:36:45.181Z","__v":0}
```

Insecure Upload Functionality – Partially Fixed

ID-002
Risk: Medium
CVSS Score: 6.9

CVSS Version 3.1 Metrics

Attack Vector: **Network**	Scope: **Changed**	Attack Complexity: **Low**
Confidentiality: **None**	Required Privileges: **High**	Integrity: **High**
User Interaction: **Required**	Availability: **Low**	

Description

Insecure file upload functionality refers to a vulnerability in web applications where an attacker can upload malicious files to the server. This can lead to a variety of attacks, including code execution, denial of service, and data breaches. The vulnerability arises when the application does not properly validate or restrict the types, content, and destination of uploaded files.

Potential Risks include:

- **Remote Code Execution (RCE):** Attackers can upload scripts or binaries that, when executed, grant them control over the server.
- **System Compromise:** Malicious files can be used to introduce backdoors, malware, or other malicious tools.
- **Data Breach:** Uploaded files might be used to extract sensitive data from the server.
- **Denial of Service (DoS):** Attackers can upload large or specially crafted files to exhaust server resources.

Example

The following vulnerability could potentially be exploited by uploading an executable file, which an admin-level user could later download and be tricked into running on their device. This action potentially allows an attacker to gain control over the user's device.

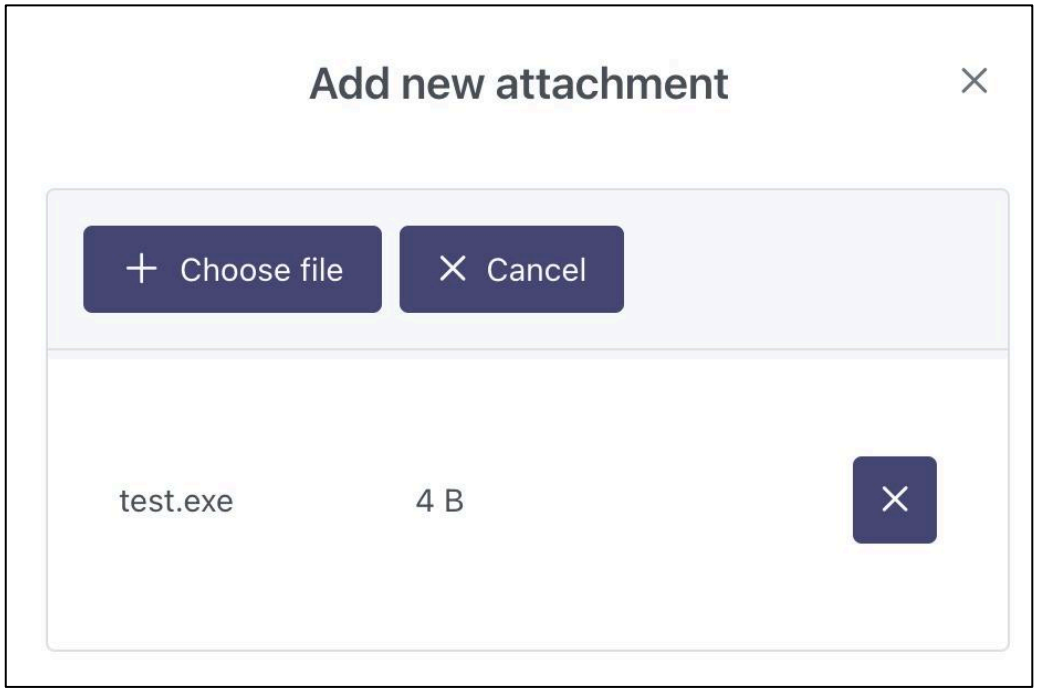


Figure 16 - Arbitrary file types allowed to be uploaded

Remediation

- **Whitelist File Types:** Only allow specific file types to be uploaded. Rely on server-side validation, not just client-side checks.
- **Limit File Size:** Set a maximum file size for uploads to prevent resource exhaustion attacks.
- **Use Secure File Upload Libraries:** Utilise well-known libraries or solutions that handle file uploads securely.

By implementing these recommendations, the risk associated with insecure file upload vulnerabilities can be significantly reduced.

Affected Hosts

URL/IP Address	Port
admin.teamflect.com	443

References

https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

Retest Notes

The issue has been partially remedied by front end checks, these however could be bypassed using a proxy to alter the initial request as shown below:

Original Request -

```
POST /api/admin/addBulkAttachment?attachmentVisibility=0&oid=38f40519-33a6-
4533-80f6-37c5f6708585 HTTP/2
Host: api.teamflect.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:125.0)
Gecko/20100101 Firefox/125.0
Accept: application/json, text/plain, */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NWVlY2M0YTViZjYyMDVkOTMwYT
g3OTAiLCJ0ZW5hbnRJZCI6IjFlMWYyMzRhLTE0MzQtNGVlNy04M2U1LTZiOTBkNmM2MDBhMCIsI
mlhdCI6MTcxNDQ2NzQ1MywiZXhwIjoxNzE0NTUzODUzLCJpc3MiOiJodHRwczovL3RlYW1mbGVj
dC5jb20ifQ.eOeuwe8FoQySt2ibCoggUqIWIiF-zJmnamE0zJkNKcs
Content-Type: multipart/form-data; boundary=--------------------------
224301185332561056441840515544
Content-Length: 227
Origin: https://admin.teamflect.com
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Te: trailers

----------------------------224301185332561056441840515544
Content-Disposition: form-data; name="files"; filename="test.exe.jpg"
Content-Type: image/jpeg

Ss
----------------------------224301185332561056441840515544—
```

Edited Request -

```
POST /api/admin/addBulkAttachment?attachmentVisibility=0&oid=38f40519-33a6-
4533-80f6-37c5f6708585 HTTP/2
Host: api.teamflect.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:125.0)
Gecko/20100101 Firefox/125.0
Accept: application/json, text/plain, */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NWVlY2M0YTViZjYyMDVkOTMwYT
g3OTAiLCJ0ZW5hbnRJZCI6IjFlMWYyMzRhLTE0MzQtNGVlNy04M2U1LTZiOTBkNmM2MDBhMCIsI
mlhdCI6MTcxNDQ2NzQ1MywiZXhwIjoxNzE0NTUzODUzLCJpc3MiOiJodHRwczovL3RlYW1mbGVj
dC5jb20ifQ.eOeuwe8FoQySt2ibCoggUqIWIiF-zJmnamE0zJkNKcs
Content-Type: multipart/form-data; boundary=--------------------------
224301185332561056441840515544
Content-Length: 228
Origin: https://admin.teamflect.com
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Te: trailers

----------------------------224301185332561056441840515544
Content-Disposition: form-data; name="files"; filename="test.exe"
Content-Type: application/exe

Ss
```

```
----------------------------224301185332561056441840515544--
```

Response -

```
HTTP/2 200 OK
Date: Tue, 30 Apr 2024 09:17:40 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 701
Access-Control-Allow-Origin: *
Etag: W/"2bd-Pg+ePauoYjbT0z5ycsW0CJ0WmVM"
Request-Context: appId=cid-v1:
X-Azure-Ref:                                                    20240430T091740Z-
er1bd8974cf2hkwn6hqfyqx0q40000000g80000000012ydh
X-Cache: CONFIG_NOCACHE
Accept-Ranges: bytes

[{"url":"https://teamblee.blob.core.windows.net/attachments/test17144686604
09.exe","fileName":"test.exe","title":"test.exe",
```

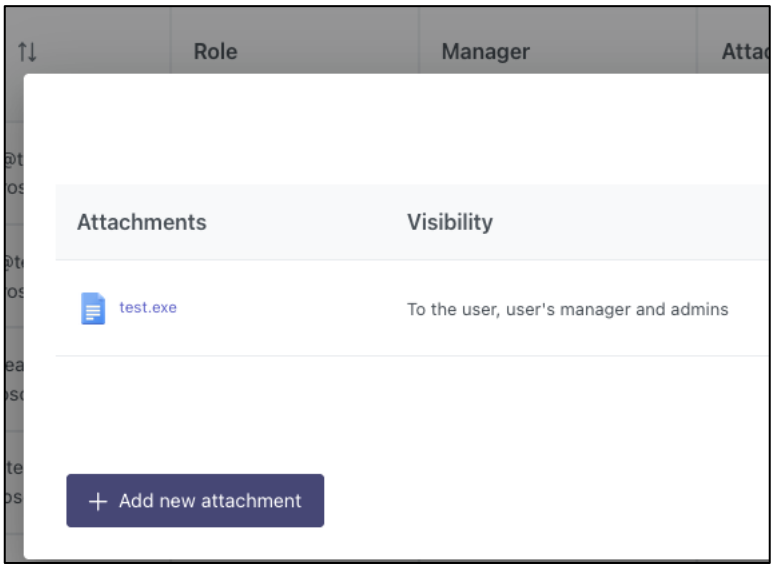


Figure 17 - Arbitrary file types allowed to be uploaded

HTML Injection in Email Template - Fixed

ID-003
Risk: Medium
CVSS Score: 6.3

CVSS Version 3.1 Metrics

Attack Vector: **Network**	Scope: **Unchanged**	Attack Complexity: **Low**
Confidentiality: **High**	Required Privileges: **Low**	Integrity: **Low**
User Interaction: **Required**	Availability: **None**	

Description

HTML injection is a type of injection vulnerability that occurs when a user is able to control an input point and is able to inject arbitrary HTML code into a vulnerable web page. This vulnerability can have many consequences, like disclosure of a user's session cookies that could be used to impersonate the victim, or, more generally, it can allow the attacker to modify the page content seen by the victims.

This vulnerability occurs when user input is not correctly sanitized, and the output is not encoded. An injection allows the attacker to send a malicious HTML page to a victim. The targeted browser will not be able to distinguish (trust) legitimate parts from malicious parts of the page, and consequently will parse and execute the whole page in the victim's context.

There is a wide range of methods and attributes that could be used to render HTML content. If these methods are provided with an untrusted input, then there is an high risk of HTML injection vulnerability. For example, malicious HTML code can be injected via the innerHTML JavaScript method, usually used to render user-inserted HTML code. If strings are not correctly sanitized, the method can enable HTML injection. A JavaScript function that can be used for this purpose is document.write().

Example

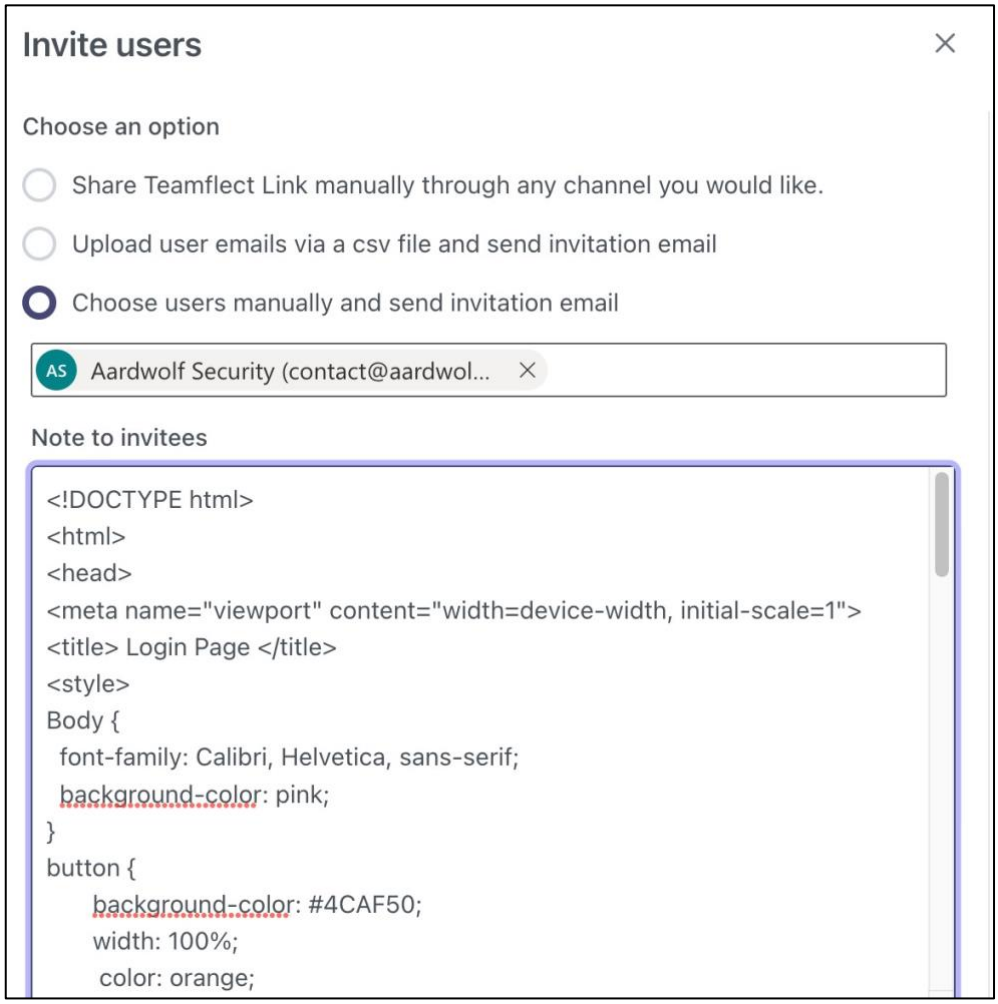


Figure 18 - Note to invitees allows HTML input

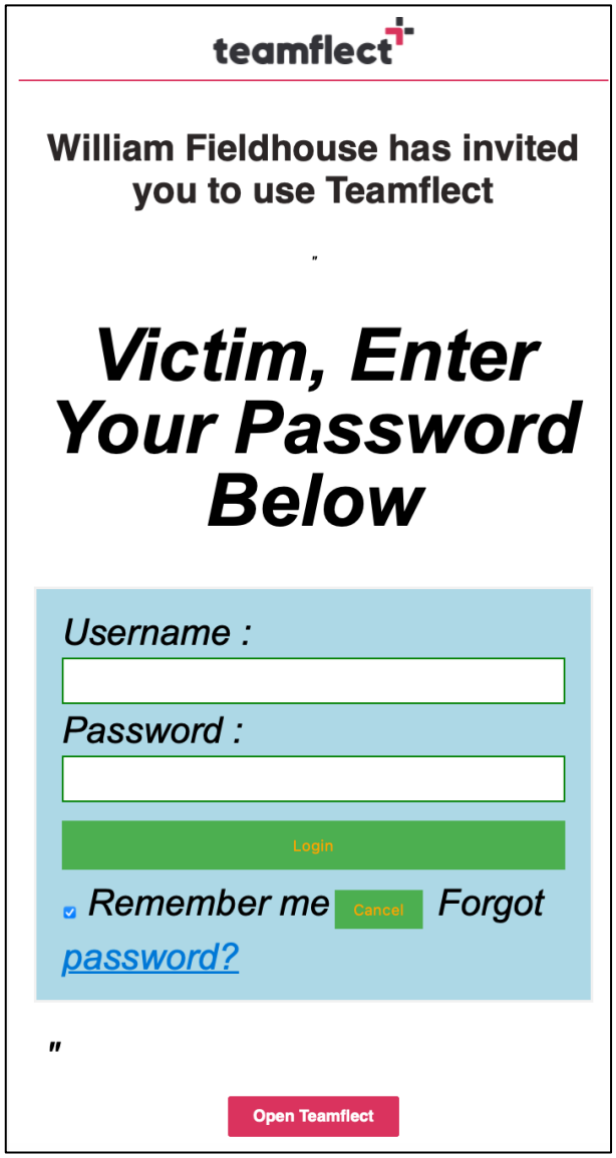


Figure 19 - HTML input renders in the victims email

Remediation

The following is recommended to remediate HTML injection vulnerabilities:

- Never trust user input
- Never insert untrusted data except in allowed locations
- HTML escape before inserting untrusted data into HTML element content
- Use whitelists in place for Blacklists for input filtering

Affected Hosts

URL/IP Address	Port
admin.teamflect.com	443

References

https://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29

Retest Notes

Special characters have now been stripped from forms sent to other users:

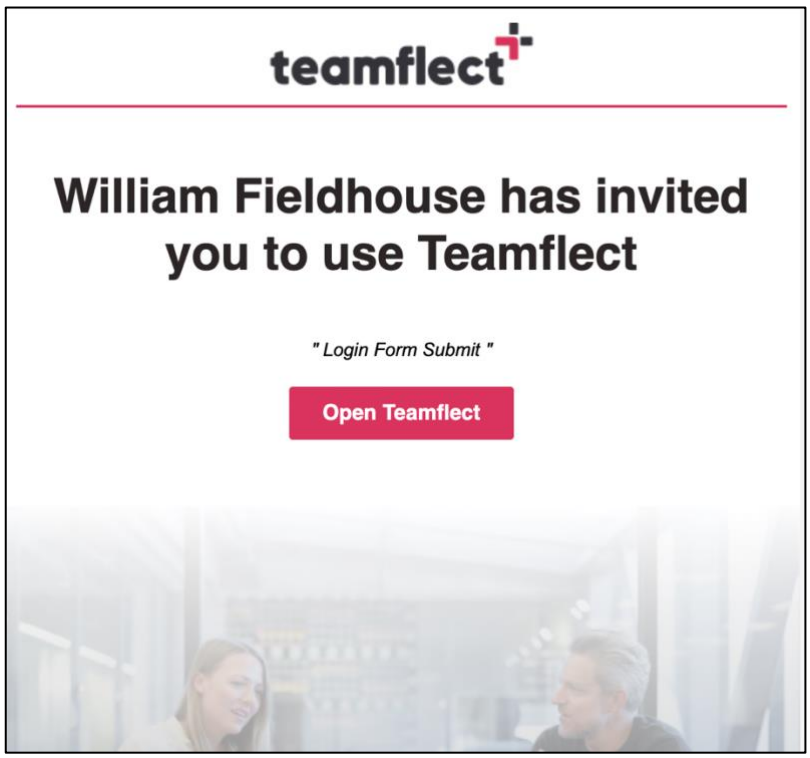


Figure 20 - HTML input no longer renders in the victims email.

Software Version Numbers Revealed - Accepted

ID-004
Risk: Low
CVSS Score: 3.7

CVSS Version 3.1 Metrics

Attack Vector: **Network**	Scope: **Unchanged**	Attack Complexity: **High**
Confidentiality: **Low**	Required Privileges: **None**	Integrity: **None**
User Interaction: **None**	Availability: **None**	

Description

Web server fingerprinting is the task of identifying the type and version of web server that a target is running on. While web server fingerprinting is often encapsulated in automated testing tools, it is important for researchers to understand the fundamentals of how these tools attempt to identify software, and why this is useful.

Accurately discovering the type of web server that an application runs on can enable security testers to determine if the application is vulnerable to attack. In particular, servers running older versions of software without up-to-date security patches can be susceptible to known version-specific exploits.

Example

Request -
https://admin.teamflect.com/static/js/3933.992e63ef.chunk.js.map

Response -

```
HTTP/2 200 OK
Cache-Control: public, must-revalidate, max-age=15, proxy-revalidate, s-maxage=30
Content-Type: application/octet-stream
Last-Modified: Mon, 15 Jan 2024 13:08:57 GMT
Etag: "04427147"
X-Cache: TCP_MISS
Strict-Transport-Security: max-age=10886400; includeSubDomains; preload
```

```
Referrer-Policy: same-origin
[...]
Released under MIT license <https://lodash.com/license>\n * Based on
Underscore.js 1.8.3 <http://underscorejs.org/LICENSE>\n *
```

Remediation

While exposed server information is not necessarily in itself a vulnerability, it is information that can assist attackers in exploiting other vulnerabilities that may exist. Exposed server information can also lead attackers to find version-specific server vulnerabilities that can be used to exploit unpatched servers. For this reason it is recommended that some precautions be taken. These actions include:

Obscuring web server information in headers, such as with Apache's mod_headers module.

Using a hardened reverse proxy server to create an additional layer of security between the web server and the Internet.

Ensuring that web servers are kept up-to-date with the latest software and security patches.

Affected Hosts

URL/IP Address	Port
admin.teamflect.com	443

References

https://en.wikipedia.org/wiki/Proxy_server#Reverse_proxies
https://httpd.apache.org/docs/current/mod/mod_headers.html

Retest Notes

The issue has been accepted.

Enabled CBC Mode Ciphers Detected - Accepted

ID-005
Risk: Low
CVSS Score: 3.7

CVSS Version 3.1 Metrics

Attack Vector: **Network**	Scope: **Unchanged**	Attack Complexity: **High**
Confidentiality: **None**	Required Privileges: **None**	Integrity: **Low**
User Interaction: **None**	Availability: **None**	

Description

The affected server is configured to support Cipher Block Chaining (CBC) encryption mode. This mode of operation is known to have weaknesses and might allow an attacker located within the same network as the user to decrypt the encrypted traffic.

Example

See the following output for Test SSL:



Figure 21 - CBC Ciphers in Use

```
Running client simulations (HTTP) via sockets

Android 6.0                    TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256, 256 bit ECDH (P-256)
Android 7.0 (native)           TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
Android 8.1 (native)           TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
Android 9.0 (native)           TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 384 bit ECDH (P-384)
Android 10.0 (native)          TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
Android 11 (native)            TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
Android 12 (native)            TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
Chrome 79 (Win 10)             TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 384 bit ECDH (P-384)
Chrome 101 (Win 10)            TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
Firefox 66 (Win 8.1/10)        TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
Firefox 100 (Win 10)           TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
IE 6 XP                        No connection
IE 8 Win 7                     No connection
IE 8 XP                        No connection
IE 11 Win 7                    TLSv1.2 ECDHE-RSA-AES256-SHA384, 384 bit ECDH (P-384)
IE 11 Win 8.1                  TLSv1.2 ECDHE-RSA-AES256-SHA384, 384 bit ECDH (P-384)
IE 11 Win Phone 8.1            TLSv1.2 ECDHE-RSA-AES128-SHA256, 256 bit ECDH (P-256)
IE 11 Win 10                   TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
Edge 15 Win 10                 TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 384 bit ECDH (P-384)
Edge 101 Win 10 21H2           TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
Safari 12.1 (iOS 12.2)         TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 384 bit ECDH (P-384)
Safari 13.0 (macOS 10.14.6)    TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 384 bit ECDH (P-384)
Safari 15.4 (macOS 12.3.1)     TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
Java 7u25                      No connection
Java 8u161                     TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 384 bit ECDH (P-384)
Java 11.0.2 (OpenJDK)          TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 384 bit ECDH (P-384)
Java 17.0.3 (OpenJDK)          TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
go 1.17.8                      TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
LibreSSL 2.8.3 (Apple)         TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
OpenSSL 1.0.2e                 TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 384 bit ECDH (P-384)
OpenSSL 1.1.0l (Debian)        TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
OpenSSL 1.1.1d (Debian)        TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
OpenSSL 3.0.3 (git)            TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 256 bit ECDH (P-256)
Apple Mail (16.0)              TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 384 bit ECDH (P-384)
Thunderbird (91.9)             TLSv1.2 ECDHE-RSA-AES256-GCM-SHA384, 384 bit ECDH (P-384)
```

Figure 22 - Older ciphers supported by Internet Explorer 11

Remediation

It is recommended that CBC mode cipher encryption is disabled on HTTPS services.

Affected Hosts

URL/IP Address	Port
admin.teamflect.com	443
app.teamflect.com	443
api.teamflect.com	443

References

https://learn.microsoft.com/en-us/dotnet/standard/security/vulnerabilities-cbc-mode

Retest Notes

The issue has been accepted.

Cross-Origin Resource Sharing: Arbitrary Origin Trusted - Not Fixed

ID-006
Risk: Low
CVSS Score: 3.5

CVSS Version 3.1 Metrics

Attack Vector: **Network**	Scope: **Unchanged**	Attack Complexity: **Low**
Confidentiality: **Low**	Required Privileges: **Low**	Integrity: **None**
User Interaction: **Required**	Availability: **None**	

Description

A HTML5 Cross-Origin Resource Sharing (CORS) policy controls whether and how content running on other domains can perform a two-way interaction with the domain that publishes the policy. The policy is fine-grained and can apply access controls per-request based on the URL and other features of the request.

Trusting arbitrary origins effectively disables the Same-Origin Policy (SOP), allowing two-way interaction by third-party web sites. Unless the response consists only of unprotected public content, this policy is likely to present a security risk.

If the site specifies the header *Access-Control-Allow-Credentials: true*, third-party sites may be able to carry out privileged actions and retrieve sensitive information. Even if it does not, attackers may be able to bypass any IP-based access controls by proxying through users' browsers.

Example

The target API supported the wildcard option for its CORS policy. This wildcard option prevents authenticated requests by default, as it sets the *Access-Control-Allow-Credentials* header to false. However, whilst this does reduce the impact, it still permits arbitrary origins from interacting with the API.

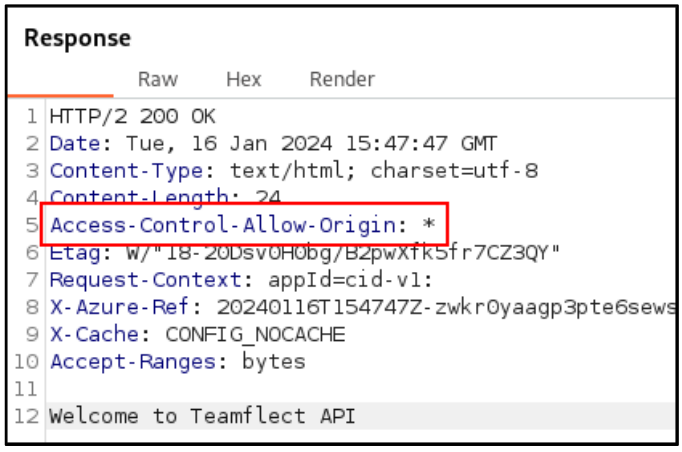


Figure 23 - API Response Permits Arbitrary Origins

Remediation

Rather than using a wildcard or programmatically verifying supplied origins, use a whitelist of trusted domains.

Affected Hosts

URL/IP Address	Port
app.teamflect.com	443

References

https://cwe.mitre.org/data/definitions/942.html

Retest Notes

The affected header still is unchanged from the original assessment:

Request –
https://api.teamflect.com/

Response –

```
HTTP/2 200 OK
Date: Mon, 29 Apr 2024 18:19:26 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 24
Access-Control-Allow-Origin: *
Etag: W/"18-20Dsv0H0bg/B2pwXfk5fr7CZ3QY"
Request-Context: appId=cid-v1:
X-Azure-Ref:                                                    20240429T181926Z-
17fd566d545sjgll6s75ua1z18000000039g000000015sby
X-Cache: CONFIG_NOCACHE
Accept-Ranges: bytes

Welcome to Teamflect API
```

Cookie Without HttpOnly Flag Set - Fixed

ID-007
Risk: Low
CVSS Score: 3.5

CVSS Version 3.1 Metrics

Attack Vector: **Network**	Scope: **Unchanged**	Attack Complexity: **Low**
Confidentiality: **Low**	Required Privileges: **Low**	Integrity: **None**
User Interaction: **Required**	Availability: **None**	

Description

The target application was found to set the session cookie insecurely. Cookies which do not have the HttpOnly attribute set may be accessed by scripts running within the browser, this makes them vulnerable to being exposed by issues such as Cross-Site Scripting (XSS), allowing the attacker to hijack the user's session and gain unauthorised access to the application as a result.

This flag prevents JavaScript from accessing the browser's cookies, which can prevent session hijacking when an XSS is found.

Example

The *takdirUser* session token did not enforce the HttpOnly flag:



Figure 24 - The HttpOnly option was set to false

Remediation

Ensure that all cookies set by the application are set with the 'HttpOnly' attribute.

Affected Hosts

URL/IP Address	Port
app.teamflect.com	443

References

https://owasp.org/www-community/HttpOnly

Retest Notes

The issue has been remedied as the cookie is no longer present.

Missing HTTP Security Headers – Partially Fixed

ID-008
Risk: Low
CVSS Score: 3.1

CVSS Version 3.1 Metrics

Attack Vector: **Network**	Scope: **Unchanged**	Attack Complexity: **High**
Confidentiality: **Low**	Required Privileges: **Low**	Integrity: **None**
User Interaction: **None**	Availability: **None**	

Description

Although not a direct vulnerability with the web application, the following HTTP headers can be added to contribute to increasing overall security posture.

Strict-Transport-Security

The HTTP Strict-Transport-Security response header (often abbreviated as HSTS) informs browsers that the site should only be accessed using HTTPS, and that any future attempts to access it using HTTP should automatically be converted to HTTPS.

Content Sniffing Not Disabled header

There was no "X-Content-Type-Options" HTTP header with the value nosniff set in the response. The lack of this header means that certain browsers try to determine the content type and encoding of the response even when these properties are defined correctly. This can make the web application vulnerable against XSS attacks, e.g. Internet Explorer and Safari treat responses with the content type text/plain as HTML, if they contain HTML tags.

Content Security Policy (CSP)

It was discovered that no CSP was implemented by the web application in scope. The CSP header is a feature that allows web developers to specify what resources are allowed to be included in a particular page.

X-Permitted-Cross-Domain-Policies

The X-Permitted-Cross-Domain-Policies header is used to permit cross-domain requests from Flash and PDF documents.

Example

Missing headers:

- strict-transport-security
- x-content-type-options
- content-security-polic
- x-permitted-cross-domain-policies

Request -
https://api.teamflect.com/api/admin/updateCompetency

Response-

```
HTTP/2 200 OK
Date: Thu, 18 Jan 2024 10:28:34 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 1986
Access-Control-Allow-Origin: *
Etag: W/"7c2-RDRdMS/0ZoZGvRWDBmnrrNqQNoM"
Request-Context: appId=cid-v1:
X-Azure-Ref:                                                    20240118T102833Z-
t3vty2baa50h1d25vdv2gwd4mc00000001dg000000013npr
X-Cache: CONFIG_NOCACHE
Accept-Ranges: bytes
```

Remediation

Ensure that HTTP security headers are applied where necessary

Affected Hosts

URL/IP Address	Port
admin.teamflect.com	443
app.teamflect.com	443
api.teamflect.com	443

References

https://owasp.org/www-project-secure-headers/#div-headers

Retest Notes

The API has HTTP headers missing:

Request –
https://api.teamflect.com/

Response –

```
HTTP/2 200 OK
Date: Mon, 29 Apr 2024 18:19:26 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 24
Access-Control-Allow-Origin: *
Etag: W/"18-20Dsv0H0bg/B2pwXfk5fr7CZ3QY"
Request-Context: appId=cid-v1:
X-Azure-Ref:                                                    20240429T181926Z-
17fd566d545sjgll6s75ua1z18000000039g000000015sby
X-Cache: CONFIG_NOCACHE
Accept-Ranges: bytes

Welcome to Teamflect API
```

Cross-Domain Script Include - Accepted

ID-009
Risk: Low
CVSS Score: 3.1

CVSS Version 3.1 Metrics

Attack Vector: **Network**	Scope: **Unchanged**	Attack Complexity: **High**
Confidentiality: **Low**	Required Privileges: **Low**	Integrity: **None**
User Interaction: **None**	Availability: **None**	

Description

When an application includes a script from an external domain, this script is executed by the browser within the security context of the invoking application. The script can therefore do anything that the application's own scripts can do, such as accessing application data and performing actions within the context of the current user.

If you include a script from an external domain, then you are trusting that domain with the data and functionality of your application, and you are trusting the domain's own security to prevent an attacker from modifying the script to perform malicious actions within your application.

Example

Request -
https://admin.teamflect.com/

Response -

```
HTTP/2 200 OK
Cache-Control: public, must-revalidate, max-age=15, proxy-revalidate, s-maxage=30
Content-Type: text/html
Last-Modified: Mon, 15 Jan 2024 13:08:57 GMT
Etag: "04427147"
Vary: Accept-Encoding
X-Cache: TCP_MISS
Strict-Transport-Security: max-age=10886400; includeSubDomains; preload
Referrer-Policy: same-origin
X-Content-Type-Options: nosniff
```

```
X-Xss-Protection: 1; mode=block
X-Dns-Prefetch-Control: off
[...]
}</script><script        src="//fast.appcues.com/144868.js"></script><script
type="text/javascript">
```

Remediation

Scripts should ideally not be included from untrusted domains. Applications that rely on static third-party scripts should consider using Subresource Integrity to make browsers verify them, or copying the contents of these scripts onto their own domain and including them from there. If that is not possible (e.g. for licensing reasons) then consider reimplementing the script's functionality within application code.

Affected Hosts

URL/IP Address	Port
admin.teamflect.com	443

References

https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/11-Client_Side_Testing/13-Testing_for_Cross_Site_Script_Inclusion

Retest Notes

The issue has been accepted

Concurrent Logins Permitted - Accepted

ID-010
Risk: Informational
CVSS Score: 0.0

CVSS Version 3.1 Metrics

Attack Vector: **Network**	Scope: **Unchanged**	Attack Complexity: **Low**
Confidentiality: **None**	Required Privileges: **High**	Integrity: **None**
User Interaction: **None**	Availability: **None**	

Description

The web application did not prevent a particular user from logging in multiple times and creating multiple simultaneous sessions.

Failure to prevent concurrent logins makes it harder for a user to identify that their account has been compromised as illegitimate and legitimate use could occur at the same time.

In addition, permitting a user to log in multiple times may create concurrency faults. These are errors created when data is updated (almost) simultaneously by separate requests from alternative sessions. This can lead to inconsistencies or exceptions (depending upon the nature of the data being modified) and at the very least could use extra resources, cause user confusion and create inconsistent log entries.

Remediation

User accounts within a web application should only be permitted to use one session at a time. If the user authenticates again then any previously valid sessions should be immediately terminated, with an appropriate message displayed within both sessions and the event logged in back-end systems.

Affected Hosts

URL/IP Address	Port

admin.teamflect.com	443
app.teamflect.com	443

References

https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

Environment Information Disclosure - Accepted

ID-011
Risk: Informational
CVSS Score: 0.0

CVSS Version 3.1 Metrics

Attack Vector: **Network**	Scope: **Unchanged**	Attack Complexity: **Low**
Confidentiality: **None**	Required Privileges: **None**	Integrity: **None**
User Interaction: **None**	Availability: **None**	

Description

The target application revealed environment information that could allow an attacker to fingerprint and profile the application. Information such as software and version numbers can be leveraged by an attacker to perform targetted attacks.

Example

An HTTP request to the API would result in the HTTP response revealing its web framework through the *X-Powered-By* header:

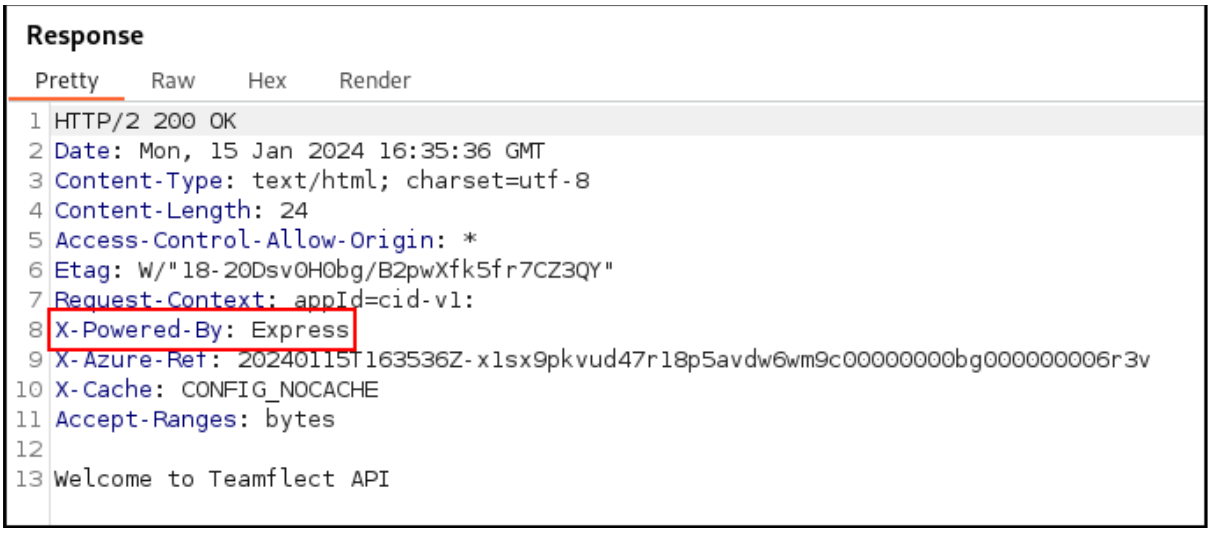


Figure 25 - X-Powered-By header disclosed the Express web framework

Remediation

Where possible, all areas of the application that disclose environment information should be disabled.

In Teamflect's situation, the X-Powered-By HTTP response header can be disabled by using the disable method:

app.disable('x-powered-by')

Affected Hosts

URL/IP Address	Port
app.teamflect.com	443

References

https://expressjs.com/en/advanced/best-practice-security.html#reduce-fingerprinting

Unencrypted Network Service - Accepted

ID-012
Risk: Informational
CVSS Score: 0.0

CVSS Version 3.1 Metrics

Attack Vector: **Network**	Scope: **Unchanged**	Attack Complexity: **Low**
Confidentiality: **None**	Required Privileges: **None**	Integrity: **None**
User Interaction: **None**	Availability: **None**	

Description

A single instance of an unencrypted network services was identified on the server. Unencrypted services are vulnerable to Man-in-the-Middle (network sniffing) attacks. A malicious user, who has gained access to the network, can read the traffic traversing it, modify the existing traffic or generate their own traffic.

In a modern wired and switched network, data traffic is normally confined between the two communicating devices. Therefore, a malicious user would need to be in a position where they could deceive the network and intercept the traffic either through ARP spoofing, DNS poisoning or listening to broadcast traffic.

Any malicious user who has placed themselves in this position would then be able to view any of the unencrypted communication. In this instance it was discovered that the unencrypted HTTP was found to be open, however, when attempting to access this port it would redirect the communication to the secure HTTPS port, therefore, this issue has been downgraded to an informational.

Example

The following example shows port 80 being open:

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-19 10:56 EST
Nmap scan report for 20.50.153.39
Host is up (0.027s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT    STATE SERVICE
```

```
80/tcp  open  http
```

Remediation

It is recommended that all services running on plaintext protocols should be disabled if not required. If the services are required, then they should be replaced with a suitable and secure alternative where possible.

In this instance, HTTP services should be disabled and replaced with HTTPS.

Affected Host

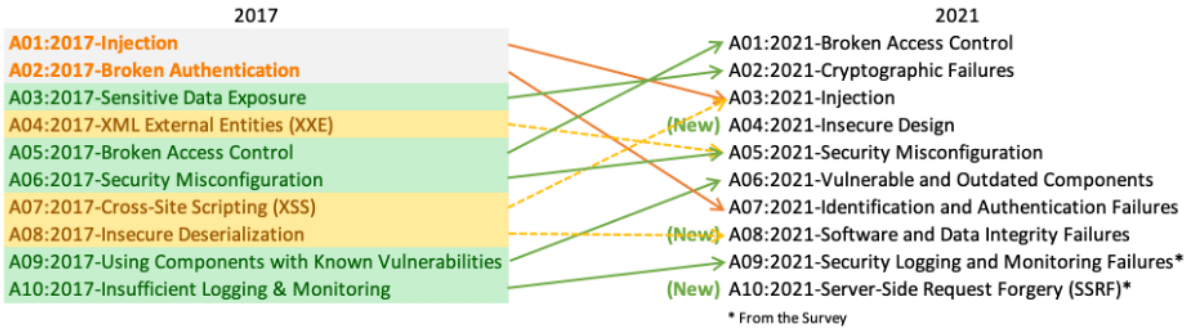
URL	Port
20.50.153.39	80/tcp

References

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/09-Testing_for_Weak_Cryptography/03-Testing_for_Sensitive_Information_Sent_via_Unencrypted_Channels

APPENDIX A – OWASP TOP 10 VULNERABILITIES

The latest top 10 vulnerability categories and how they have changed since 2017



A01:2021-Broken Access Control moves up from the fifth position; 94% of applications were tested for some form of broken access control. The 34 Common Weakness Enumerations (CWEs) mapped to Broken Access Control had more occurrences in applications than any other category.

A02:2021-Cryptographic Failures shifts up one position to #2, previously known as Sensitive Data Exposure, which was broad symptom rather than a root cause. The renewed focus here is on failures related to cryptography which often leads to sensitive data exposure or system compromise.

A03:2021-Injection slides down to the third position. 94% of the applications were tested for some form of injection, and the 33 CWEs mapped into this category have the second most occurrences in applications. Cross-site Scripting is now part of this category in this edition.

A04:2021-Insecure Design is a new category for 2021, with a focus on risks related to design flaws. If we genuinely want to "move left" as an industry, it calls for more use of threat modeling, secure design patterns and principles, and reference architectures.

A05:2021-Security Misconfiguration moves up from #6 in the previous edition; 90% of applications were tested for some form of misconfiguration. With more shifts into highly configurable software, it's not surprising to see this category move up. The former category for XML External Entities (XXE) is now part of this category.

A06:2021-Vulnerable and Outdated Components was previously titled Using Components with Known Vulnerabilities and is #2 in the Top 10 community survey, but also had enough data to make the Top 10 via data analysis. This category moves up from #9 in 2017 and is a known issue that we struggle to test and assess risk. It is the

only category not to have any Common Vulnerability and Exposures (CVEs) mapped to the included CWEs, so a default exploit and impact weights of 5.0 are factored into their scores.

A07:2021-Identification and Authentication Failures was previously Broken Authentication and is sliding down from the second position, and now includes CWEs that are more related to identification failures. This category is still an integral part of the Top 10, but the increased availability of standardized frameworks seems to be helping.

A08:2021-Software and Data Integrity Failures is a new category for 2021, focusing on making assumptions related to software updates, critical data, and CI/CD pipelines without verifying integrity. One of the highest weighted impacts from Common Vulnerability and Exposures/Common Vulnerability Scoring System (CVE/CVSS) data mapped to the 10 CWEs in this category. Insecure Deserialization from 2017 is now a part of this larger category.

A09:2021-Security Logging and Monitoring Failures was previously Insufficient Logging & Monitoring and is added from the industry survey (#3), moving up from #10 previously. This category is expanded to include more types of failures, is challenging to test for, and isn't well represented in the CVE/CVSS data. However, failures in this category can directly impact visibility, incident alerting, and forensics.

A10:2021-Server-Side Request Forgery is added from the Top 10 community survey (#1). The data shows a relatively low incidence rate with above average testing coverage, along with above-average ratings for Exploit and Impact potential. This category represents the scenario where the security community members are telling us this is important, even though it's not illustrated in the data at this time.

APPENDIX B – NMAP TCP

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-19 10:56 EST
Nmap scan report for 20.50.153.39
Host is up (0.027s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT    STATE SERVICE
80/tcp  open  http
|_http-title: Microsoft Azure Web App - Error 404
443/tcp open  https
|         ssl-cert:         Subject:         commonName=*.msha-slice-1-am2-0-
ase.p.azurewebsites.net/organizationName=Microsoft
Corporation/stateOrProvinceName=Washington/countryName=US
|       Subject       Alternative       Name:       DNS:*.msha-slice-1-am2-0-
ase.p.azurewebsites.net,                        DNS:*.scm.msha-slice-1-am2-0-
ase.p.azurewebsites.net,   DNS:*.msha-slice-1-am2-0-ase.p.azure-mobile.net,
DNS:*.scm.msha-slice-1-am2-0-ase.p.azure-mobile.net,  DNS:*.azure-apim.net,
DNS:*.consent.azure-apim.net, DNS:*.admin.azure-apim.net, DNS:msha-slice-1-
am2-0-ase.scm.p.azurewebsites.net
| Not valid before: 2023-08-13T00:00:00
|_Not valid after:  2024-08-13T23:59:59
|_ssl-date: 2024-01-19T15:56:18+00:00; 0s from scanner time.
| tls-alpn:
|_  http/1.1
|_http-title: Microsoft Azure Web App - Error 404

Nmap scan report for 13.107.213.64
Host is up (0.012s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT    STATE  SERVICE
53/tcp  closed domain
80/tcp  open   http
|_http-title: Site doesn't have a title.
443/tcp open   https
| ssl-cert: Subject: commonName=*.azureedge.net/organizationName=Microsoft
Corporation/stateOrProvinceName=WA/countryName=US
|        Subject        Alternative        Name:        DNS:*.azureedge.net,
DNS:*.media.microsoftstream.com,    DNS:*.origin.mediaservices.windows.net,
DNS:*.streaming.mediaservices.windows.net
| Not valid before: 2024-01-12T18:31:38
|_Not valid after:  2025-01-06T18:31:38
|_http-title: Site doesn't have a title.
```

APPENDIX C – NMAP UDP

```
└─$ sudo nmap -sU --top-ports 100 -iL iplist.txt -oA teamflect_udp
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-19 10:57 EST
Nmap scan report for 20.50.153.39
Host is up (0.015s latency).
All 100 scanned ports on 20.50.153.39 are in ignored states.
Not shown: 100 open|filtered udp ports (no-response)

Nmap scan report for 13.107.213.64
Host is up (0.011s latency).
Not shown: 98 open|filtered udp ports (no-response)
PORT    STATE    SERVICE
53/udp  filtered domain
443/udp filtered https
```