

SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY



Web Security- IE2062

Insecure Design

A04:2021

Danuka Nuwan

IT22349842

Vulnerability Title: Insecure Design

Description of Vulnerability: The application or website is considered unsafe due to several programming design flaws, including the system's vulnerability to network isolation, insufficient security measures, and failure to adhere to the principle of least privilege. Hackers can exploit this type of performance vulnerability in several ways to gain unauthorized access, disclose data, manipulate information, and disrupt services, thereby compromising the system's confidentiality, integrity, and availability.

Website - www.kindredgroup.com

Components affected:

- Application architecture
- Data flow
- Access control mechanisms
- Encryption protocols
- Input validation and sanitization
- Error handling and logging

Evaluation of Consequences:

Critical: An insecure application/website is susceptible to a multitude of security vulnerabilities and issues, which can have severe consequences such as financial loss, non-compliance with regulations, loss of confidence, and public exposure.

Instructions to Replicate:

Do a full appraisal of the approach of the application's architecture with an emphasis on the path of sensitive data, how access to such data will be restricted, and identity and trust of components.

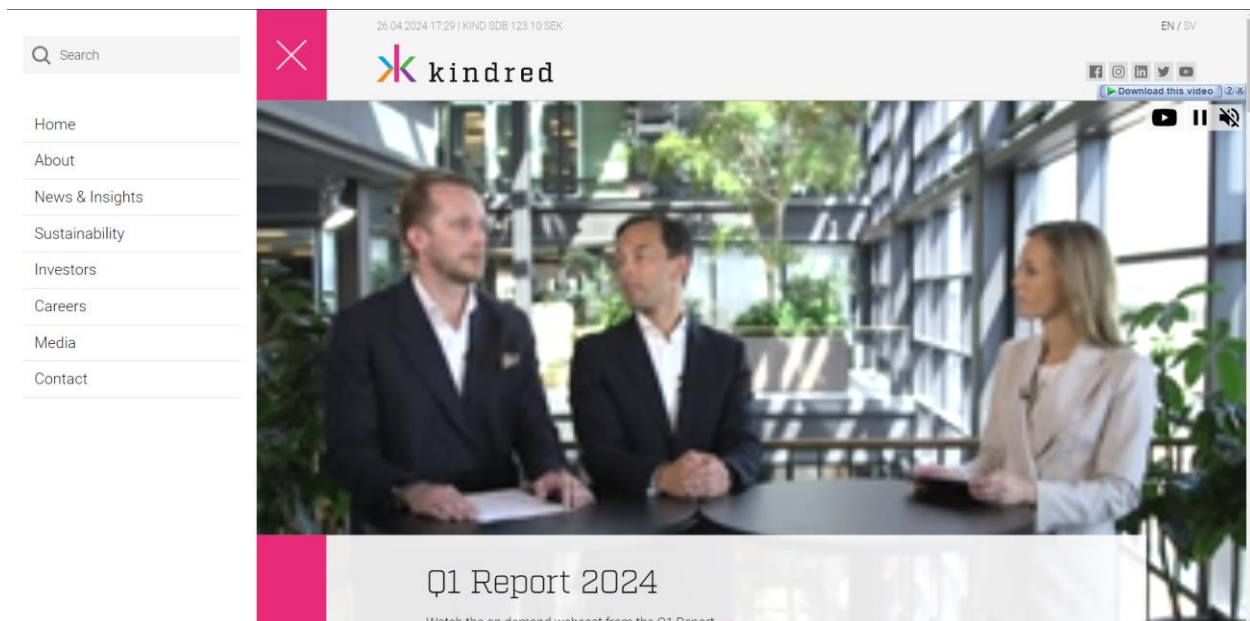
Find out possible security holes originating from design choices that are not secure including leaving input validation, inappropriate management of sensitive data, and shortage of secret encryption algorithms.

Check both the access control techniques implemented and their efficiency, by looking at user authentication, authorization, and session handling.

Assess that of the application which is to check on its error handling and logging methods so as to confirm whether they offer sufficient visibility to security-related events and anomalies.

Carry out tests to assess whether encryptions are being employed correctly through the application in order to guarantee that the communication is seriously protected whether in transit or at rest.

Proof of Concept:



6. Insecure Frame (External)

LOW



1

CONFIRMED



1

Netsparker identified an external insecure or misconfigured iframe.

Impact

IFrame sandboxing enables a set of additional restrictions for the content within a frame in order to restrict its potentially malicious code from causing harm to the web page that embeds it.

The Same Origin Policy (SOP) will prevent JavaScript code from one origin from accessing properties and functions - as well as HTTP responses - of different origins. The access is only allowed if the protocol, port and also the domain match exactly.

Here is an example, the URLs below all belong to the same origin as *http://site.com*:

http://site.com
http://site.com/
http://site.com/my/page.html

Whereas the URLs mentioned below aren't from the same origin as *http://site.com*:

http://www.site.com (a sub domain)
http://site.org (different top level domain)
https://site.com (different protocol)
http://site.com:8080 (different port)

When the `sandbox` attribute is set, the iframe content is treated as being from a unique origin, even if its hostname, port and protocol match exactly. Additionally, sandboxed content is re-hosted in the browser with the following restrictions:

- Any kind of plugin, such as ActiveX, Flash, or Silverlight will be disabled for the iframe.
- Forms are disabled. The hosted content is not allowed to make forms post back to any target.
- Scripts are disabled. JavaScript is disabled and will not execute.
- Links to other browsing contexts are disabled. An anchor tag targeting different browser levels will not execute.
- Unique origin treatment. All content is treated under a unique origin. The content is not able to traverse the DOM or read cookie information.

When the `sandbox` attribute is not set or not configured correctly, your application might be at risk.

A compromised website that is loaded in such an insecure iframe might affect the parent web application. These are just a few examples of how such an insecure frame might affect its parent:

- It might trick the user into supplying a username and password to the site loaded inside the iframe.
- It might navigate the parent window to a phishing page.
- It might execute untrusted code.
- It could show a popup, appearing to come from the parent site.

Vulnerabilities

6.1. <https://www.kindredgroup.com/>

CONFIRMED

Frame Source(s)

- https://www.youtube.com/embed/3hcGibYHmjM?version=3&loop=1&mute=1&autoplay=1&showinfo=0&rel=0&playsinline=0&modestbranding=1&controls=1&cc_load_policy=0
- <https://www.youtube.com/embed/KklWksJKRT4?mute=1&autoplay=1&showinfo=0&rel=0&playsinline=0&modestbranding=1>

Parsing Source

- DOM Parser

Request

```
GET / HTTP/1.1
Host: www.kindredgroup.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
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) : 515.5052 Total Bytes Received : 92412 Body Length : 91711 Is Compressed : No

```
HTTP/1.1 200 OK
Set-Cookie: EpiStateMarker=true; path=/
Set-Cookie: ARRAffinity=f62dc48792b6e16789f38b9331562ab71aac9ec805fac06e15282a091076b114;Path=/;HttpOnly;Secure;Domain=www.kindredgroup.com
Set-Cookie: ARRAffinitySameSite=f62dc48792b6e16789f38b9331562ab71aac9ec805fac06e15282a091076b114;Path=/;HttpOnly;SameSite=None;Secure;Domain=www.kindredgroup.com
Request-Context: appId=cid-v1:f6f0367f-9d11-44c1-bd87-a01942e8c453
CF-RAY: 87b1cb230fc9513a-CMB
Server: cloudflare
CF-Cache-Status: DYNAMIC
Connection: keep-alive
Content-Encoding:
Strict-Transport-Security: max-age=2592000
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Date: Sat, 27 Apr 2024 21:09:25 GMT
```


Remedy

- Apply sandboxing in inline frame

```
<iframe sandbox src="framed-page-url"></iframe>
```


- For untrusted content, avoid the usage of seamless attribute and allow-top-navigation, allow-popups and allow-scripts in sandbox attribute.

External References

- [HTML5 Security Cheat Sheet](#)

Remedy References

- [How to Safeguard your Site with HTML5 Sandbox](#)
- [Play safely in sandboxed IFrames](#)

 CLASSIFICATION	
OWASP 2017	A6
SANS Top 25	16
WASC	15
ISO27001	A.14.1.2

8. Missing Content-Type Header

LOW



1

Netsparker detected a missing Content-Type header which means that this website could be at risk of a MIME-sniffing attack.

Impact

MIME type sniffing is a standard functionality in browsers to find an appropriate way to render data where the HTTP headers sent by the server are either inconclusive or missing.

This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the intended content type.

The problem arises once a website allows users to upload content which is then published on the web server. If an attacker can carry out XSS (Cross-site Scripting) attack by manipulating the content in a way to be accepted by the web application and rendered as HTML by the browser, it is possible to inject code in e.g. an image file and make the victim execute it by viewing the image.

Vulnerabilities

8.1. <https://www.kindredgroup.com/globalassets/images/>

Certainty



Request

```
GET /globalassets/images/ HTTP/1.1
Host: www.kindredgroup.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
Cookie: ARRAffinity=f62dc48792b6e16789f38b9331562ab71aac9ec805fac06e15282a091076b114; EpiStateMarker=true; ARRAffinitySameSite=f62dc48792b6e16789f38b9331562ab71aac9ec805fac06e15282a091076b114; ShowMenu=true; cookie-consent-date=1714252212312; cookie-consent-settings={%22ad_storage%22:%22granted%22%2C%22analytics_storage%22:%22granted%22%2C%22functionality_storage%22:%22granted%22%2C%22personalization_storage%22:%22granted%22%2C%22security_storage%22:%22granted%22}
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) : 566.9974 Total Bytes Received : 194 Body Length : 0 Is Compressed : No

```
HTTP/1.1 500 Internal Server Error
CF-Cache-Status: DYNAMIC
CF-RAY: 87b1cd4ebecd513a-CMB
Server: cloudflare
Connection: keep-alive
Content-Length: 0
Date: Sat, 27 Apr 2024 21:10:54 GMT
```

Remedy

1. When serving resources, make sure you send the content-type header to appropriately match the type of the resource being served. For example, if you are serving an HTML page, you should send the HTTP header:

```
Content-Type: text/html
```

2. Add the X-Content-Type-Options header with a value of "nosniff" to inform the browser to trust what the site has sent is the appropriate content-type, and to not attempt "sniffing" the real content-type.

```
X-Content-Type-Options: nosniff
```

External References

- [MIME Sniffing: feature or vulnerability?](#)
- [X-Content-Type-Options HTTP Header](#)

9. Missing X-Frame-Options Header

LOW  1

Netsparker detected a missing X-Frame-Options header which means that this website could be at risk of a clickjacking attack.

The X-Frame-Options HTTP header field indicates a policy that specifies whether the browser should render the transmitted resource within a frame or an iframe. Servers can declare this policy in the header of their HTTP responses to prevent clickjacking attacks, which ensures that their content is not embedded into other pages or frames.

Impact

Clickjacking is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on a framed page when they were intending to click on the top level page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to other another page, most likely owned by another application, domain, or both.

Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker.

Vulnerabilities

9.1. <https://www.kindredgroup.com/>

Certainty



Request

```
GET / HTTP/1.1
Host: www.kindredgroup.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
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) : 515.5052 Total Bytes Received : 92412 Body Length : 91711 Is Compressed : No

```
HTTP/1.1 200 OK
Set-Cookie: EPiStateMarker=true; path=/
Set-Cookie: ARRAffinity=f62dc48792b6e16789f38b9331562ab71aac9ec805fac06e15282a091076b114;Path=/;HttpOnly;Secure;Domain=www.kindredgroup.com
Set-Cookie: ARRAffinitySameSite=f62dc48792b6e16789f38b9331562ab71aac9ec805fac06e15282a091076b114;Path=/;HttpOnly;SameSite=None;Secure;Domain=www.kindredgroup.com
Request-Context: appId=cid-v1:f6f0367f-9d11-44c1-bd87-a01942e8c453
CF-RAY: 87b1cb230fc9513a-CMB
Server: cloudflare
CF-Cache-Status: DYNAMIC
Connection: keep-alive
Content-Encoding:
Strict-Transport-Security: max-age=2592000
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Date: Sat, 27 Apr 2024 21:09:25 GMT
```

```
<!DOCTYPE html>
<html lang="en" class="no-js" data-categories="Regulatory">
<head>
<meta charset="utf-8">

<title>Kindred Group plc &#x2013; We continue to transform gambling</title>

<script>document.documentElement.classList.remove("no-js")</script>

<script type="text/javascript">
!function (T, l, y) { var S = T.location, k = "script", D = "instrumentationKey", C = "ingestionendpoint", I = "disableExceptionTracking", E = "ai.device.", b = "toLowerCase", w = "crossOrigin", N = "POST", e = "appInsightsSDK", t = y.name || "appInsights"; (y.name || T[e]) && (T[e] = t); var n = T[t] || function (d) { var g = !1, f = !1, m = { initialize: !0, queue: [], sv: "5", version: 2, config: d }; function v(e, t) { var n = {}, a = "Browser"; return n[E + "id"] = a[b](), n[E + "type"] = a, n["ai.operation.name"] = S && S.pathname || "_unknown_", n["ai.internal.sdkVersion"] = "javascript:snippet_" + (m.sv || m.version), { time: function () { var e = new Date; function t(e) { var t = "" + e; return 1 === t.length && (t = "0" + t), t } return e.getUTCFullYear() + "-" + t(1 + e.getUTCMonth()) + "-" + t(e.getUTCDate()) + "T" + t(e.getUTCHours()) + ":" + t(e.getUTCMinutes()) + ":" + t(e.getUTCSeconds()) + "." + ((e.getUTCMilliseconds() / 1e3).toFixed
```

Remedy

- Sending the proper X-Frame-Options in HTTP response headers that instruct the browser to not allow framing from other domains.
 - X-Frame-Options: DENYIt completely denies to be loaded in frame/iframe.
 - X-Frame-Options: SAMEORIGINIt allows only if the site which wants to load has a same origin.
 - X-Frame-Options: ALLOW-FROM URLIt grants a specific URL to load itself in a iframe. However please pay attention to that, not all browsers support this.
- Employing defensive code in the UI to ensure that the current frame is the most top level window.

External References

- [Clickjacking](#)
- [Can I Use X-Frame-Options](#)
- [X-Frame-Options HTTP Header](#)

Remedy References

- [Clickjacking Defense Cheat Sheet](#)



CLASSIFICATION

OWASP 2013	A5
OWASP 2017	A6
SANS Top 25	693
CAPEC	103
ISO27001	A.14.2.5

Suggested Resolution:

Conduct a full security assessment of the application's design, identifying and prioritizing any security risks and vulnerabilities.

Implementing secure incision procedures, like least privilege principle, defense-in-depth and fail-safe by design policies, throughout application architecture.

Implement secure coding practices that connectors avoid common security mistakes such as Invalid input, output encoding, unparameterized queries, and appropriate error handling.

Upgrade access control mechanisms to implement least privilege principle and RBAC (RBAC is role-based access control, let's come up with a better acronym) tenets, limiting the users to only the functions and resources they are provided.

Utilize robust encryption algorithms and protocols to safeguard mobile information in motion and while ceasing, implementing good key management techniques.