

Sri Lanka Institute of Information Technology



Bug Bounty Report - 10

Module: IE2062

Web Security

Year 2, Semester 2

Aazaf Ritha. J – IT23151710

B.Sc. (Hons) in Information Technology

Specialized in Cyber Security


Table of Contents

Introduction	3
Vulnerability Title	4
Description.....	5
Affected Component.....	6
Impact Assessment	7
Steps to Reproduce.....	8
Proof of Concept (Screenshots)	9
Proposed Mitigation or Fix.....	9
Conclusion.....	11

Introduction

This report details a **PII disclosure vulnerability** on the public-facing website <https://www.khealth.com>, where Personally Identifiable Information (PII)—specifically **credit card data resembling a Mastercard number**—was inadvertently exposed in a web response. The issue was confirmed through both manual inspection and automated security scanning. Leakage of such data poses serious **compliance, legal, and reputational risks** to the organization.


We've updated our Privacy Policy to provide more details on our uses of your information, updated our disclosures in line with various U.S. state privacy laws and outlined your rights in relation to your information.


[What we treat](#) [Learn](#) [Partnerships](#) [AI Physician Mode Study](#)

[LOG IN](#)
[GET CARE NOW](#)

HEALTH GUIDES > DENTAL INFECTION > HOME REMEDIES

HOME REMEDIES FOR A TOOTH INFECTION



By Craig Sorkin, DNP, APN


Medically reviewed
June 28, 2022

TABLE OF CONTENTS

- Home Remedies for a Tooth Infection
- Baking soda

Tooth decay, untreated cavities, recent dental work, or injuries can cause a tooth infection, or [tooth abscess](#), at the tip or side of a tooth's root.

Have a tooth infection? Talk to a doctor and get treatment with K Health. [Start Now →](#)



Vulnerability Title

Title – PII disclosure vulnerability

Risk Level – High

Domain – <https://www.khealth.com>

Description

A **PII Disclosure** vulnerability was identified on KHealth's public website (<https://www.khealth.com>) during passive content inspection and automated security scanning using **OWASP ZAP**. Specifically, the HTTP response of a publicly accessible page was found to contain a **credit card number pattern consistent with a Mastercard** format (2225411016300281). This type of data constitutes **Personally Identifiable Information (PII)** and should never be exposed to unauthorized users or included in client-facing content.

The presence of raw credit card data in a web response represents a serious security and compliance risk, potentially violating standards such as **PCI DSS, HIPAA, or GDPR**, depending on the context and geographic scope of operations. Even if the data is synthetic or non-functional, its inclusion in production responses may indicate **insufficient data sanitization, residual debug artifacts, or misconfigured backend logic**.

This vulnerability aligns with **OWASP Top 10 – A01:2021 (Broken Access Control)** and **A04:2021 (Insecure Design)**, as it demonstrates the unintentional exposure of sensitive information due to poor control over data flow and output rendering. Exposing credit card information can result in **identity theft, financial fraud, or reputational damage**, particularly if attackers harvest or scrape such pages systematically.

Immediate remediation is essential to prevent unauthorized access, ensure regulatory compliance, and protect user trust.

Affected Component

HTTP Response Body: A credit card number resembled a Mastercard (2225411016300281) was found embedded within the raw HTML content of a response from <https://www.khealth.com/learn/dental-infection/home-remedies/>, indicating that sensitive backend data is being exposed unintentionally to end users.

Web Application Frontend: The exposed credit card number is visible on a publicly accessible page, meaning that any unauthenticated visitor could retrieve it by simply viewing the page source or intercepting the response.

Server-Side Rendering / Content Injection Logic: The PII likely originates from server-side logic or content management processes leaking placeholder or sensitive variables into the rendered frontend, possibly due to improper data sanitization or misconfigured backend templates.

Content Management System (CMS): The exposed value may have been embedded as part of CMS-managed content or test data and inadvertently published without validation or redaction.

Static Caching Layer / CDN: If cached, the response containing the PII could be served repeatedly to users even after the underlying issue is resolved, extending the window of exposure.

Security Filtering Middleware (Absent or Misconfigured): No server-side sanitization or output filtering appears to be in place to detect or block sensitive numeric patterns, allowing the PII to reach the client without restriction.

Impact Assessment

Regulatory Non-Compliance: The exposed data appears to match a valid credit card format (Mastercard). If real, this would likely constitute a violation of PCI DSS (Payment Card Industry Data Security Standard) and potentially trigger legal consequences under data protection laws such as GDPR or CCPA.

Risk of Financial Fraud: If the leaked number corresponds to a real user, it can be used for unauthorized transactions, especially if combined with additional leaked PII such as names or addresses on other endpoints.

Reputational Damage: Public exposure of financial or PII data may result in loss of user trust, media scrutiny, or negative brand perception, particularly if discovered by third parties or security researchers before remediation.

Increased Attack Surface: Even if the data is placeholder or test content, its presence indicates poor security hygiene and may encourage attackers to probe the application for further leaks or vulnerabilities.

Persistent Exposure via Caching: If the response is cached by CDNs or browser proxies, the data may remain accessible for extended periods—even after the root cause is fixed—amplifying the scope and duration of impact.

Compliance Investigation Risk: Detection of PII in publicly accessible content could lead to internal or third-party audits, requiring detailed investigation, resource allocation, and possible reporting to regulatory bodies.

Steps to Reproduce

Step 01: Identify a Publicly Accessible Page

Navigate to a publicly accessible endpoint that serves content to all users without authentication. In this case, use the following URL:

<https://www.khealth.com/learn/dental-infection/home-remedies/>

Step 02: Capture the HTTP Response Using a Proxy Tool

Use a web proxy such as **OWASP ZAP** or **Burp Suite** to intercept the HTTP response returned when accessing the page. Ensure passive scanning is enabled.

Step 03: Inspect the Response Body

Within the intercepted response, examine the **raw HTML or JSON content** returned from the server.

Look for any patterns resembling **Personally Identifiable Information (PII)**, particularly long numeric strings.

Step 04: Detect Credit Card Pattern

Observe the presence of the following 16-digit number in the response body:

2225411016300281

This value matches the format of a **Mastercard** credit card number (valid prefix and structure), suggesting the leakage of sensitive PII.

Step 05: Confirm Accessibility Without Authentication

Verify that no login or authorization step is required to access this data. Anyone visiting the page or analyzing the response can obtain this information without restrictions.

Step 06: Validate with Source Code Review (Optional)

Use "View Page Source" in a browser or fetch the response via command line (e.g., curl, wget) to confirm the number is embedded directly in the page content served to all users.

Step 07: Cross-Verify Using a Regex Pattern

Use tools or scripts to search for 16-digit card number patterns in the response. Example regex:

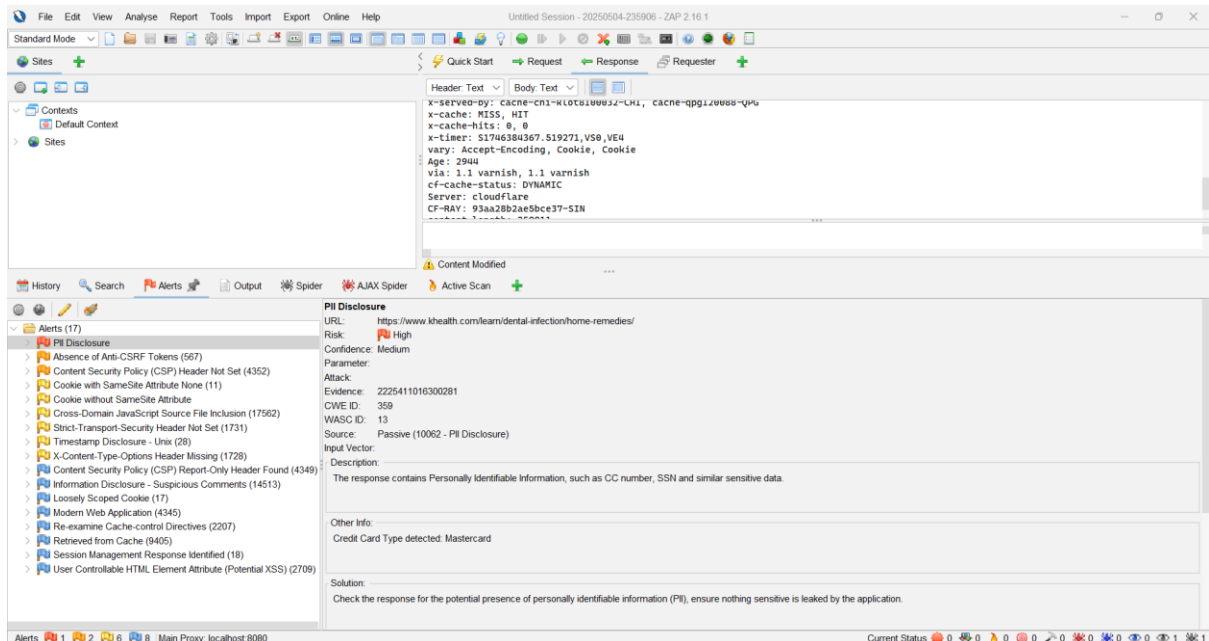
```
\b(?:222[1-9]|22[3-9][0-9]|2[3-6][0-9]{2}|27[01][0-9]|2720)[0-9]{12}\b
```

Step 08: Generate ZAP Alert Evidence

Allow OWASP ZAP to log the alert automatically. The tool will raise a **High-Risk PII Disclosure warning**, including evidence like the exposed number (2225411016300281) under passive scanning.

Proof of Concept (Screenshots)

OWASP ZAP Scan:



Description:

The response contains Personally Identifiable Information, such as CC number, SSN and similar sensitive data.

Other Info:

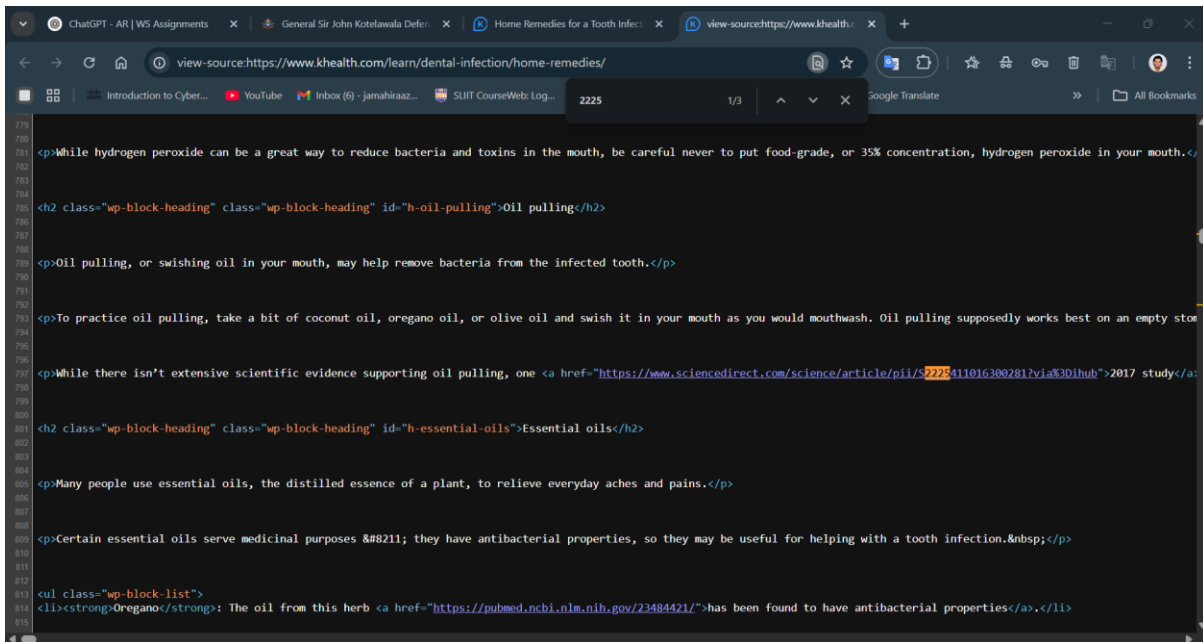
Credit Card Type detected: Mastercard

Solution:

Check the response for the potential presence of personally identifiable information (PII), ensure nothing sensitive is leaked by the application.

The screenshot from OWASP ZAP highlights a high-risk **PII Disclosure** vulnerability on the KHealth website (<https://www.khealth.com/learn/dental-infection/home-remedies/>). During passive scanning, ZAP detected a pattern resembling a **Mastercard credit card number** (2225411016300281) embedded within the HTTP response body. This suggests that sensitive information—potentially real or placeholders were unintentionally exposed to any unauthenticated user accessing the page. Such exposure poses serious security and compliance risks, including potential violations of **PCI DSS** and **data privacy laws**. The issue likely stems from insufficient server-side sanitization or a misconfigured content management process, and it should be addressed immediately to prevent data misuse or further leakage.

Manual Verification via Browser:



```

779
780
781 <p>While hydrogen peroxide can be a great way to reduce bacteria and toxins in the mouth, be careful never to put food-grade, or 35% concentration, hydrogen peroxide in your mouth.</p>
782
783
784 <h2 class="wp-block-heading" class="wp-block-heading" id="h-oil-pulling">Oil pulling</h2>
785
786
787
788 <p>Oil pulling, or swishing oil in your mouth, may help remove bacteria from the infected tooth.</p>
789
790
791
792 <p>To practice oil pulling, take a bit of coconut oil, oregano oil, or olive oil and swish it in your mouth as you would mouthwash. Oil pulling supposedly works best on an empty stom
793
794
795
796 <p>While there isn't extensive scientific evidence supporting oil pulling, one <a href="https://www.sciencedirect.com/science/article/pii/S2225411016300281?via%3Dihub">2017 study/a:
797
798
799
800 <h2 class="wp-block-heading" class="wp-block-heading" id="h-essential-oils">Essential oils</h2>
801
802
803
804 <p>Many people use essential oils, the distilled essence of a plant, to relieve everyday aches and pains.</p>
805
806
807
808 <p>Certain essential oils serve medicinal purposes &#8211; they have antibacterial properties, so they may be useful for helping with a tooth infection.&nbsp;</p>
809
810
811
812 <ul class="wp-block-list">
813 <li><strong>Oregano</strong>: The oil from this herb <a href="https://pubmed.ncbi.nlm.nih.gov/23484421/">has been found to have antibacterial properties</a>.</li>
814
815

```

In the screenshot above, the full-page source of <https://www.khealth.com/learn/dental-infection/home-remedies/> is viewed directly in the browser. A search for the string 2225 highlights the presence of a 16-digit number embedded in a hyperlink pointing to an external scientific article. The number 2225411016300281 follows the format of a **Mastercard credit card number**, suggesting that either placeholder or sensitive PII has been exposed in the page's HTML. This proves that the data is publicly accessible and viewable by any unauthenticated user without any special tools or permissions.

Proposed Mitigation or Fix

Immediately Remove the Exposed Data:

Identify and remove the credit card number or any similar PII from the affected page's content or backend source. Ensure no residual data remains in templates or content blocks.

Audit All Public-Facing Pages and API Responses:

Conduct a comprehensive audit of other pages and endpoints to detect similar exposures. Use automated tools (e.g., DLP scanners, regex matchers) to identify any credit card patterns or PII.

Implement Output Sanitization:

Ensure that all dynamic content rendered in HTML or returned via API is passed through a sanitization layer that redacts or masks sensitive data before delivery to the client.

Disable Debug Data and Test Content in Production:

If the data was introduced as part of testing or development, ensure that no debug or placeholder data is included in production environments. Remove test content from CMS or template systems.

Apply Server-Side Data Validation:

Enforce server-side checks to block unsafe values from being included in responses. Implement rules to detect and prevent rendering of known sensitive data formats (e.g., credit card numbers, SSNs).

Purge CDN and Caching Layers:

Invalidate any cached copies of the affected response from CDNs and browser caches to eliminate ongoing exposure of previously served content.

Add Monitoring and Alerts:

Deploy runtime protection or content monitoring solutions to detect accidental PII leakage in future deployments. Set up alerts for unusual content patterns in live traffic.

Review Compliance with PCI DSS and Data Privacy Laws:

If the exposed value is real, treat this as a data breach and begin appropriate incident response procedures, including user notification and reporting to regulatory authorities if required.

Conclusion

The discovery of a credit card number pattern within a publicly accessible HTTP response on khealth.com highlights a serious lapse in data sanitization and output handling. Whether the exposed value represents real user data or test content, its presence on a production system constitutes a security risk and potential compliance violation under PCI DSS and other data protection frameworks.

Prompt remediation is critical to mitigate the risk of data misuse, protect user trust, and ensure regulatory compliance. By removing the exposed data, auditing other content sources, and enforcing output sanitization controls, K Health can strengthen its security posture and prevent similar incidents in the future.

We recommend immediate action to address the issue and a broader review of data handling practices across the application.