Black-Box Cybersecurity Assessment of Software as a Medical Device (SaMD): A Case Study

Abdulmajeed Alshammari[1], Shouki A. Ebad[2]
Department of Computer Science-Faculty of Science, Northern Border University, Arar, Saudi Arabia[1]
Center for Scientific Research and Entrepreneurship, Northern Border University, Arar, Saudi Arabia[2]

***Abstract*—Based on Sommerville's robust software reliability theory and eight design principles based on best practices (DPG), this study conducted a comparative assessment of two Dexcom software platforms: the Dexcom Clarity web portal and the Dexcom ONE+ iPhone application. The first phase of the study employed a "black-box" assessment framework, combining off-the-shelf security analysis tools with application-layer behavioral testing workflows developed specifically for this project. The initial two phases focused on the web portal, analyzing communication encryption and HTTP security header configurations. Subsequent phases involved a more in-depth implementation assessment, focusing on actual interactive features, data types, and automated responses triggered by glucose sensor alerts within the application. However, this approach also revealed that certain system elements required access to the source code for verification. The resulting workflow enables the assessment of all externally visible security attributes while identifying system components that necessitate source code access for validation. Of the eight assessment criteria, four were found to have issues, either in whole or in part. Key issues identified include: session cookies missing the `SameSite` attribute (G1); acceptance of clinically abnormal carbohydrate values without rejection or warning (G2); inclusion of the `unsafe-inline` directive in the Content Security Policy (CSP), increasing the risk of Cross-Site Scripting (XSS) attacks (G4); and the absence of a dedicated alert mechanism for Wi-Fi connectivity loss during critical medical functions (G7). Due to the requirement for source code access, this "black-box" approach could not evaluate the remaining four criteria (G3, G5, G6, and G8). Finally, this paper presents a reproducible evaluation procedure and offers applicable, standards-compliant mitigation recommendations for each identified issue.**

***Keywords*—SaMD; DPG; black-box testing; dexcom clarity; dexcom ONE+; input validation; CSP; XSS; CSRF; mHealth security**

I. Introduction

The International Medical Device Regulators Forum (IMDRF) defines "Software as a Medical Device" (SaMD) as software capable of autonomously performing medical functions without relying on specific medical hardware [1]. Continuous glucose monitoring (CGM) applications fit this category; rash analyses or missed alerts are more than false alarms, they can sway patient action on insulin dose decisions affecting patients' safety [2].

Regardless of the notable therapeutic potential, few studies have evaluated the reliability of available CGM software from a programming perspective [3], [4]. Past research so far has only touched upon connectivity security, regulatory compliance [5], [6], or the analysis of acknowledged vulnerabilities [7]. There has yet to be a study that has specifically utilized Sommerville's DPG framework on commercial CGM SaMD products.

Research on the safety of Software as a Medical Device (SaMD) has largely focused on regulatory compliance, known vulnerabilities, or communication security; however, little attention has been paid to how the observable behavior of commercial SaMD aligns with independent software engineering best practices [8].

Thus, this is the first time that framework has been applied to products from one company (rather than for all CGM on the market). Specific to FDA recommendations about systemically evaluating cybersecurity for medical device software, evidence-based evaluation is also emphasized [9].

This research gap is uniquely addressed in this study through the four-stage "black-box" evaluation of two Dexcom platforms, namely the Dexcom Clarity web portal and the Dexcom ONE+ iOS application. The first two stages analyse the web portal building security and HTTP header setup, whereas the third and fourth phases test by inputting data to application testing and observing real-time response of Sensory events. All test situations are documented with screenshots to support the evaluation findings.

Existing research on the security of software applications as medical devices (SaMDs) has addressed only a few perspectives. Nurgalieva et al. [3] described findings on privacy and security practices in mobile health applications in an exploratory study, but without conducting a comprehensive analysis of a specific commercial product. Papageorgiou et al. [6] conducted an in-depth investigation of mobile health applications, revealing a widespread vulnerability, but without applying a structured software engineering framework such as DPG. Previous work has focused on security at the limited connectivity level [5] or cataloging previously discovered vulnerabilities [7]. This study is unique because it systematically applies Sommerville's eight trustworthy programming practices to a single FDA-approved product and quantitatively evaluates whether the commercial version of the software adopts these principles, relying, where possible, on screenshots (an unprecedented approach in the cited literature) and also using a CVSS-based severity scale.

It's important to clarify from the outset that the black-box method used limits access to and knowledge of the source code of the evaluated platforms. Consequently, four of the eight principles of reliable programming (guidelines G3, G5, G6, and G8) cannot be externally verified with this approach. While

this limitation is discussed in detail in the Study Limitations section, it is mentioned here so that the reader can properly contextualize the evaluation results presented later in this article.

II. Background

A. Guiding Principles for Reliable Programming

Sommerville outlines eight fundamental principles that facilitate the development of reliable and secure software, particularly for critical systems such as medical software [10]. In summary, these principles are as follows:

- **G1**: Minimize the exposure of sensitive information.
- **G2**: Validate all data before processing.
- **G3**: Implement appropriate error and exception handling mechanisms.
- **G4**: Avoid programming practices that could lead to errors or increase the likelihood of their occurrence.
- **G5**: Ensure the system has recovery or restart capabilities in the event of a failure.
- **G6**: Ensure that array out-of-bounds errors do not occur during execution.
- **G7**: Implement timeout mechanisms when interacting with external services or connections.
- **G8**: Assign explicit names to all constants rather than embedding them directly in the code (i.e., avoid "hard-coding").

B. Dexcom ONE+ Platform

The Dexcom ONE+ is a continuous glucose monitoring (CGM) device classified as Software as a Medical Device (SaMD). It was developed by Dexcom, Inc., a medical device company headquartered in San Diego, California [11]. The product has obtained CE certification and FDA approval, and is compatible with both iOS and Android systems.

The sensor sends readings every five minutes to an accompanying application on a paired smartphone using Bluetooth Low Energy (BLE). It also offers a web-based option namely Dexcom Clarity, which can be accessed by healthcare providers and people managing diabetes to share data and observe the health status. In this study, the evaluation was performed on both platforms: at server and network levels (Phases 1 and 2) for the Dexcom Clarity platform, and at application level (Phases 3 and 4) for the Dexcom ONE+ application.

III. Methodology

The assessment findings for each DPG guideline were categorized into the subsequent four categories:

- FI (Fully Implemented).
- PI (Partially Implemented).
- NP (Not Implemented/Absent).
- N/E (Not Evaluated).

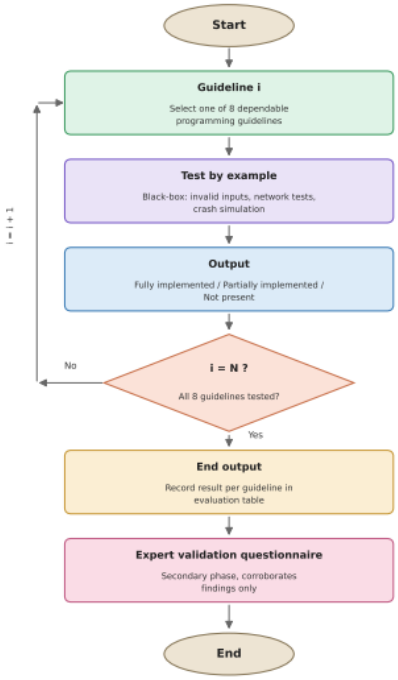


Fig. 1. DPG evaluation workflow applied to each guideline (Overview).

These categories were used according to the operational definitions detailed below. A guideline was classified as FI (Fully Implemented) when all associated tests passed without exception. A guideline was considered PI (Partially Implemented) if several aspects of it were compliant, while other relevant aspects were not implemented. A guideline was classified as NP (Not Implemented/Absent) if every associated test clearly revealed non-compliance. Finally, the N/E (Not Evaluated) designation was reserved for guidelines whose verification required consultation of the source code or internal implementation details not accessible via black-box testing.

There were four phases of evaluation across two platforms; Phase 1 and Phase 2, the Dexcom Clarity web portal; and Phase 3 and Phase 4, the Dexcom ONE+ iPhone app.

The overall process of evaluating each of the eight DPG criteria is shown in Fig. 1. This starts with a standard scanning phase, and undergoes black-box testing and ratings of findings; during this process each criterion is evaluated independently before being compiled into an expert review.

A. Phase 1 - Transport Layer Security Testing (Qualys SSL Labs)

Verified the security of the website connection using the Qualys SSL Labs tool (for the clarity.dexcom.com domain) [12]. The testing concentrated on a few security features, such as the TLS protocol versions supported, the cipher strength, the certificate chain validity and compliance with the "Perfect Forward Secrecy" standard. The tool provided an overall letter grade (scalar) from A to F, in addition it also validated the testing results against requirements of NIST SP 800-52 Revision 2 [13] to validate its connection parameters are within security best practices.

B. Phase 2 - HTTP Security Header Evaluation (MDN Observatory)

The security headers of Dexcom Clarity portal were assessed with the MDN HTTP Observatory tool [14]. It evaluated ten categories of security headers:

- Content Security Policy (CSP).
- Cookies.
- CORS (Cross-Origin Resource Sharing) policy.
- Redirection.
- Referrer-Policy.
- HTTP Strict Transport Security (HSTS).
- Subresource Integrity.
- X-Content-Type-Options.
- X-Frame-Options.
- Cross-Origin Resource Policy.

C. Phase 3 - Application Interface and Input Data Validation Testing (Dexcom ONE+)

Application-level testing (Phases 3 and 4) was conducted on an iPhone 16 with iOS 26.5.2 and the latest version of the Dexcom ONE+ app available on the App Store at the time of testing. The app was paired with a Dexcom ONE+ sensor, which features an all-in-one sensor/transmitter architecture similar to the Dexcom G7 generation. The test account's registered region was set to Saudi Arabia. Testing was conducted over Wi-Fi, without a cellular data connection. All experiments were conducted between June 3 and June 10, 2026. Each test was run only once.

At this time, an iPhone with the Dexcom ONE+ app was used to populate inputs in three fields with numbers for blood glucose, insulin, and carbohydrates. We tested four scenarios across three input fields: 1) entering values within the permitted range, 2) entering values outside the permitted range, 3) entering non-numeric characters or symbols, and 4) entering clinically implausible values.

D. Phase 4 - Sensor Reading Tests and Alert Scenarios (Dexcom ONE+)

A third round of tests was carried out at this point based on requirements defined for the G7 and G2 models. The app was monitored during typical, day-to-day use with the Dexcom ONE+ sensor as part of this study. The response of the app under conditions mimicking loss of Bluetooth or Wi-Fi connectivity was estimated by sequentially turning Bluetooth and Wi-Fi off/on, and responses to hyperglycemia as well as hypoglycemia alerts were verified.

IV. Results

A. Phase 1 - Transport-Layer Security (Qualys SSL Labs)

Table I presents the results of the SSL Labs assessment for Dexcom Clarity. The corresponding SSL Labs report is shown in Fig. 6.

B. Phase 2 - HTTP Security Headers (MDN Observatory)

Table II summarises the HTTP security header evaluation results. The complete results of the analysis performed by the MDN Observatory are presented in Fig. 7a and Fig. 7b.

C. Phase 3 - Input Validation (G2) - Dexcom ONE+

Table III presents the results of 12 test cases related to manual data entry. The results indicate that both blood glucose and insulin input fields utilize a numeric keypad, and the application correctly rejects any values falling outside the permitted range (i.e., below the minimum or above the maximum). The carbohydrate input field accepts clinically unrealistic values (such as 300 g and 999 g) without rejecting them or issuing any warnings, further confirming a clear lack of semantic validation regarding upper-limit checks for this field. Fig. 5a and Fig. 5b show how the app processes an input of 999 g of carbohydrates. Once accepted, this input is recorded in the meal history.

D. Phase 4 - Sensor and Alert Behaviour (G2, G7) - Dexcom ONE+

Table IV presents five sensor test scenarios. Hyperglycaemic readings (343 and 290 mg/dL) trigger a yellow alert, while a hypoglycaemic reading (89 mg/dL) triggers a red alert. Bluetooth interruptions are handled correctly; however, Wi-Fi disconnection triggers no user notification.

The value of 89 mg/dL used here corresponds to the internal threshold that triggers the hypoglycemia alarm on the Dexcom ONE+ app and not to a strict clinical diagnosis of hypoglycemia. According to current clinical guidelines, such as those published by the American Diabetes Association [17], hypoglycemia is generally defined at lower thresholds (e.g., below 70 mg/dL). Therefore, the discrepancy between the app's internal alarm threshold and the conventional clinical definition should be interpreted as a characteristic of the app's alarm logic and not as a statement that 89 mg/dL constitutes hypoglycemia in the general clinical sense. It should also be noted that some patients may experience symptoms of hypoglycemia at different glucose levels, depending on physiological factors.

E. Evidence Screenshots

This subsection summarizes the visual data collected during Phases 1–4 of this evaluation, in the form of captioned screenshots, to confirm the reproducibility of the presented results. Fig. 2a–3a and 3b–4 illustrate the sensor alarm and connectivity loss scenarios in Section IV-D. Fig. 5a–5b, 6, and 7a–7b present the input validation analysis for Section IV-C, as well as the results of the transport layer evaluations (including header evaluation) for Sections IV-A and IV-B, respectively.

F. Guidelines Not Evaluable (Black-Box Constraints)

Table V lists the four DPG guidelines that could not be assessed due to the black-box methodology.

TABLE I. SSL LABS ASSESSMENT - DEXCOM CLARITY

G	Guideline	Platform	Justification	Result
G1	Limit information visibility	Dexcom Clarity	TLS 1.3 confirmed; SSL Labs A+ rating. No plaintext PHI in transit. Meets NIST SP 800-52 Rev. 2 [13].	✓ FI
G1	Limit information visibility	Dexcom Clarity	HSTS header present with min-age 15,768,000 s (6 months). Preloading and includeSubDomains not yet configured - partial hardening.	∼ PI
G1	Limit information visibility	Dexcom Clarity	Valid certificate chain confirmed; strong cipher suites only (ECDHE + AES-GCM). Forward secrecy enabled. Cipher Strength score: 100/100.	✓ FI

TABLE II. SECURITY HEADERS ASSESSMENT - DEXCOM CLARITY

G	Guideline	Platform	Justification	Result
G1	Limit information visibility	Dexcom Clarity	All cookies use Secure + HttpOnly flags; SameSite missing → CSRF risk [15]. X-Frame-Options set to SAMEORIGIN - clickjacking prevented. Overall: partially compliant due to missing SameSite attribute.	∼ PI
G4	Minimize error-prone constructs	Dexcom Clarity	Enforced CSP contains `unsafe-inline` → XSS risk (CWE-79) [16].	× NP

Ethical Note: A brief explanation regarding the findings for G1 and G4 (specifically, the missing `SameSite` attribute and the presence of the `unsafe-inline` CSP directive) is provided here: these results stem from an analysis of the clarity.dexcom.com domain using a publicly available scanning tool, namely, MDN Observatory. This is a free, third-party service that anyone (including Dexcom's security team) can use at any time to analyze public domains. This process required no authentication, utilized no restricted access privileges, and employed no exploit techniques. This assessment had no understanding of zero-day vulnerabilities or targeted attacks, but was only looking for data obtainable with ordinary, publicly available analytic tools.

TABLE III. G2 MANUAL INPUT VALIDATION - DEXCOM ONE+

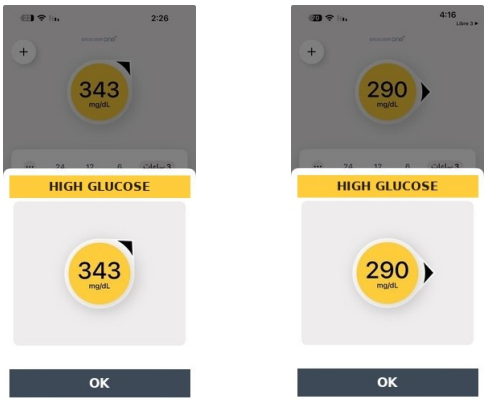
Field	Value	Result	Observation	G	Met
Blood Glucose	1 mg/dL	Rejected	Out of range (valid: 20–600 mg/dL)	G2	✓
Blood Glucose	20 mg/dL	Accepted	Min boundary accepted	G2	✓
Blood Glucose	601 mg/dL	Rejected	Out of range (valid: 20–600 mg/dL)	G2	✓
Blood Glucose	999 mg/dL	Rejected	Out of range (valid: 20–600 mg/dL)	G2	✓
Blood Glucose	Letters	Prevented	Numeric keyboard enforced	G2	✓
Insulin	1 u	Accepted	Min boundary accepted	G2	✓
Insulin	133 u	Accepted	Within range (1–300 u)	G2	✓
Insulin	500 u	Rejected	Out of range (valid: 1–300 u)	G2	✓
Insulin	Letters	Prevented	Numeric keyboard enforced	G2	✓
Carbs	300 g	Accepted	Impossible value stored - no upper-bound check	G2	×
Carbs	999 g	Accepted	Impossible value stored - no upper-bound check	G2	×
Carbs	Letters	Prevented	Numeric keyboard enforced	G2	✓

TABLE IV. SENSOR READING VALIDATION - DEXCOM ONE+

Scenario	Value	Result	Observation	G	Met
High glucose	343 mg/dL	Detected	Yellow alert triggered automatically (Fig. 2a)	G2	✓
High glucose	290 mg/dL	Detected	Yellow alert + rising arrow (Fig. 2b)	G2	✓
Low glucose	89 mg/dL	Detected	Red alert triggered automatically (Fig. 3a)	G2	✓
Bluetooth off	-	Notified	Readings stop; in-app alert shown (Fig. 3b)	G7	✓
Wi-Fi off	-	No alert	Control Center confirms Wi-Fi disconnected; no notification sent (Fig. 4)	G7	×

TABLE V. GUIDELINES REQUIRING SOURCE-CODE ACCESS

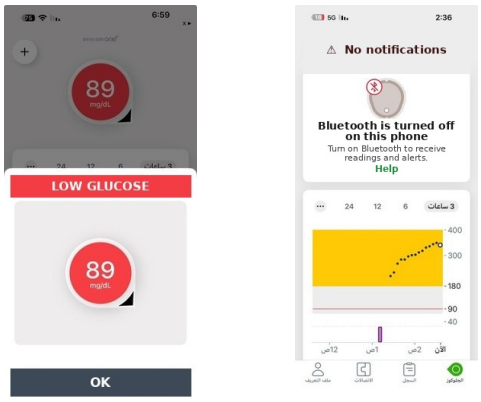
G	Guideline	Why Not Evaluated (Source Code Required)
G3	Handle all exceptions	Exception handler presence requires source-code inspection (try/catch chains).
G5	Provide restart capability	Post-crash state recovery is an internal implementation detail.
G6	Check array bounds	Compile-time/runtime memory access is not externally observable.
G8	Name all constants	Variable naming conventions are only visible in source code.



(a) High glucose alert at 343 mg/dL - yellow alert triggered automatically (G2).

(b) High glucose alert at 290 mg/dL with rising trend arrow (G2).

Fig. 2. High glucose alert scenarios.



(a) Low glucose alert at 89 mg/dL (G2).

(b) Bluetooth off - in-app alert (G7).

Fig. 3. Low glucose alert and Bluetooth disconnection notification.

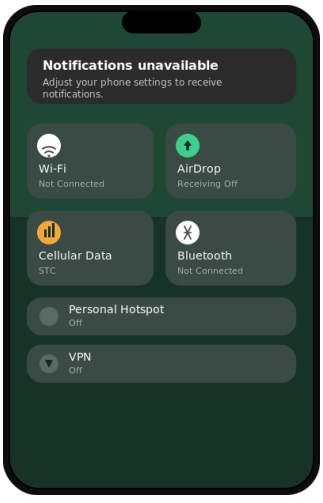
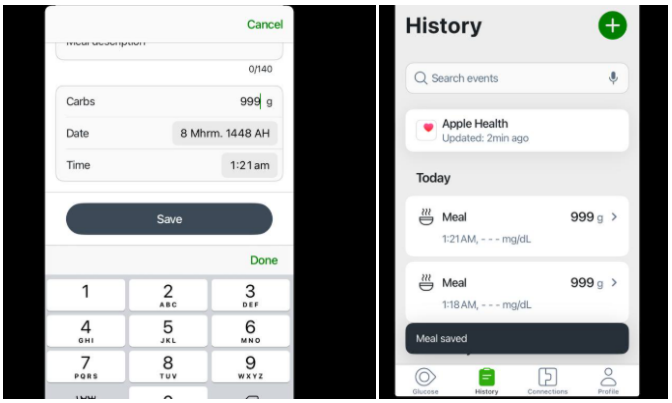


Fig. 4. Control Center showing Wi-Fi and Bluetooth both disconnected - no distinct alert is issued for the Wi-Fi disconnection (G7).



(a) Carbs field accepting 999 g - no rejection (G2).

(b) History showing 999 g meal saved successfully (G2).

Fig. 5. Carbohydrate input validation gap and its downstream effect on the meal history log.

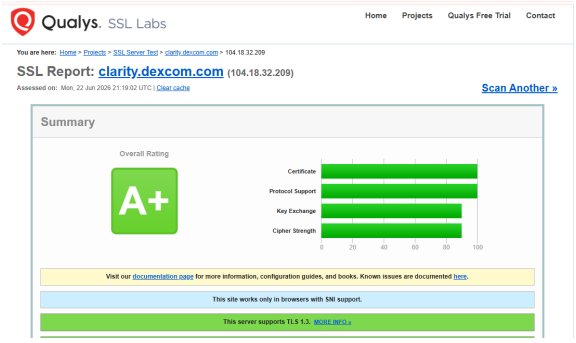


Fig. 6. Qualys SSL Labs A+ result for clarity.dexcom.com (G1 - Fully Implemented).

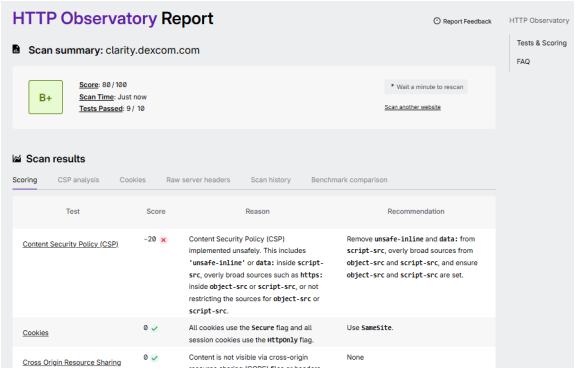
V. Analysis

A. Guideline G2 - Input Validation

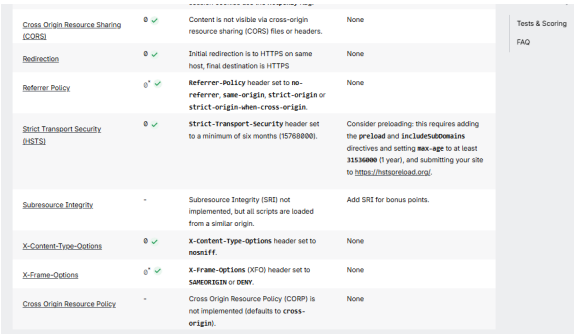
Clinically unrealistic values, such as 300 g or even 999 g, are accepted by the carbohydrate input field without rejecting them or displaying any warning message. This gives a strong indication of no semantic validation, especially regarding upper limits on this field.

This gap has real clinical significance: an inaccurate carbohydrate entry finds its way directly into the calculation of insulin dosages, and ultimately informs some decisions made by patients. The root of the problem dates back to CWE-20 (Improper Input Validation) [16], and it aligns with similar issues that have been identified in previous studies on digital health applications [6].

It's important to note that this study did not confirm whether the entered carbohydrates are taken into account by the app's insulin dosing algorithm, nor whether this leads to inconsistent dosing recommendations or actual clinical decisions. The absence of semantic validation is already a weakness in input interface design, but its clinical impact on outcomes remains speculative at best and has not yet been demonstrated in this study.



(a) MDN Observatory B+ score - CSP `unsafe-inline` penalises −20 points (G4).



(b) MDN Observatory remaining security header results for clarity.dexcom.com (G1, G4).

Fig. 7. MDN HTTP Observatory scan results.

B. Recommendation G1 - Information Disclosure

As for the session attribute, the session cookie has the `Secure` attribute as well as the `HttpOnly` attribute, which is a good indication that session management is being handled properly [15]. It does not have the `SameSite` attribute, however, which makes active sessions vulnerable to CSRF attacks (CWE-352) [15].

Thus, a malicious website might use an active victim's session to submit requests through clarity.dexcom.com without the user's knowledge, putting it at risk of revealing protected health information or triggering an inappropriate change to alert settings [15]. A small step to fix this is probably adding the `SameSite=Strict` attribute.

C. G4 Directive - Error-Prone Programming Practice

The result of the assessment reveals unsafe-inline is present in the current CSP, which is the worst problem encountered by this study. This directive allows inline scripts to run on clarity.dexcom.com, which could lead to XSS attacks that allow a hacker to hijack a user's session or manipulate blood glucose readings. It falls under CWE-79 and is a high severity vulnerability [16]. The analysis also indicates that the development team knows of the problem and is running a more conservative configuration, `CSP-Report-Only`, but not yet running it as the primary defense [14].

D. Guideline G7 - Latency and External Components

Bluetooth is the main method of connection between the Dexcom ONE+ sensor and the mobile application. When a Bluetooth connection was lost during testing, the application instantly stopped receiving sensor data and sent out a user notice (Fig. 3b). This behavior is in accordance with Guideline G7 on the management of key external components. When it comes to Wi-Fi, however, the story is a different one altogether. Cloud sync and remote monitoring are supported by Wi-Fi. Even though the device's control center evidently logged a network connectivity loss event, the application never triggered any comparable alert to notify the user (Fig. 4). The lack of a notification can also leave interruptions in the data stream unnoticed, a matter of grave concern for health providers that depend on continuous remote monitoring. This differentiated gap fails to satisfy the Guideline G7 requirements, therefore an independent alert mechanism is required to quickly inform the users whenever they lose their Wi-Fi connection.

E. Severity Classification (CVSS)

The CVSS v3.1 scores for these two confirmed vulnerabilities were calculated based solely on the security criteria applied during the assessment, as no exploitation attempts were performed. These CVSS v3.1 scores (which range from 0 to 10) are entirely based on the security metrics used in evaluation. It should be noted that this study did not involve exploitation attempts; therefore the scores do not represent the exploitability of vulnerabilities in real-world environments, but rather an upper bound on how impactful a successful exploit will be. The CVSS system was selected to provide a standardized framework enabling a clear and structured comparison of the severity of reported issues.

It's important to note that the CVSS score indicates the severity of a vulnerability in a hypothetical attack scenario, not the actual probability of such an attack occurring on the system. Therefore, these scores should not be interpreted as quantitative estimates of the actual risk, but rather as a benchmark for assessing the severity of the two confirmed vulnerabilities identified in this study.

- G4 - CSP unsafe-inline (CWE-79): AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N → Estimated CVSS v3.1 base score: 8.1 (High)
- G1 - Missing SameSite attribute (CWE-352): AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N → Estimated CVSS v3.1 base score: 5.4 (Medium)

VI. Conclusion

This study was a four-stage "black-box" assessment of two Dexcom platforms - the Dexcom ONE+ iOS application and the Dexcom Clarity web portal - using the Dependable Programming Guidelines (DPG) paradigm developed by Sommerville [10]. The evaluation was centered around four DPG criteria, particularly the ones suitable for black-box testing techniques:

- G2 was only partially met: The carbohydrate input function in Dexcom ONE+ accepts clinically implausible values without warning.

- G1 was only partially met: The `SameSite` cookie attribute is missing from Dexcom Clarity sessions.
- G4 was not met: The Content Security Policy implemented by Dexcom Clarity includes the `unsafe-inline` directive, creating an active XSS attack surface.
- G7 was only partially met: Bluetooth disconnection triggers a notification in Dexcom ONE+, whereas Wi-Fi disconnection does not.

A. Proposed Corrective Measures

Based on the findings of the evaluation, this study makes several important recommendations. First, an application can define a configurable upper threshold for carbohydrate intake. Above this threshold, the user receives warnings or is asked for confirmation before validation. A value of 150 g per meal [17] may be a practical starting point, but this threshold should not represent an absolute clinical limit; it can be adjusted if the healthcare professional or the platform wishes to classify meals based on their size, patient characteristics, or potential clinical relevance. Second, all session cookies used by Dexcom Clarity must add the `SameSite=Strict` attribute. Third, the current Content Security Policy (CSP) that Dexcom Clarity is using should remove the `unsafe-inline` directive. Finally, the Dexcom ONE+ app should have an explicit notification mechanism for Wi-Fi connection losses and it would work just like the Bluetooth disconnect alert. All of these precautions, when viewed together, align with the IEC 81001-5-1 standard governing lifecycle management for health software [18].

B. Study Limitations

This study has limitations. It used a black-box testing approach; therefore, the assessment could only cover four of the eight DPG framework criteria. It was not possible to evaluate criteria G3, G5, G6, and G8 because their evaluation requires access to the source code, which was not available in this study.

Second, the study covered products from only a single company (Dexcom Clarity and Dexcom ONE+). Therefore, the findings have not been independently verified and cannot be generalized to continuous glucose monitoring (CGM) systems produced by other companies.

Furthermore, this study presents two distinct contributions. The findings presented here, such as the lack of SameSite and the insecure content security policy, are specific to our characterization of the Dexcom Clarity and Dexcom ONE+ platforms as they were at the time of testing and should not be considered representative of other SaMD products. However, the evaluation process itself, which consists of a four-stage structure and a DPG-based scoring system, was designed to be generalizable to the evaluation of other continuous glucose monitoring (CGM) platforms or SaMDs. It is important to note that the generalizability of the methodology (not the results) has not been empirically tested and constitutes a line of future research.

All testing was performed exclusively on the iOS version of the Dexcom ONE+ app. It was not possible to verify whether the Android version performs similar data validation or uses the same alert mechanism in the event of a connection loss, nor to assess potential differences between the operating systems of the two platforms. This further limits the generalizability of our study findings.

Testing was conducted during a single session in June 2026; therefore, the results reflect the state of the platforms at that specific point in time. Dexcom may have subsequently updated some of the configurations described here.

This study did not attempt to exploit vulnerabilities, nor did it involve repeat testing. Additionally, multiple test accounts were not used to verify the consistency of observed behaviors over time. Because each test scenario was executed only once, the application's behavior cannot be considered deterministic or representative under all conditions. This behavior could be influenced by one or more uncontrolled variables (such as the network state at the time of testing or background application updates), and this observation should be confirmed through repeated testing.

ACKNOWLEDGMENT

The authors extend their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, KSA for funding this research work through the project number NBU-FFRGS-2026.

The authors acknowledge the use of Qualys SSL Labs and MDN HTTP Observatory as publicly available evaluation tools.

REFERENCES

[1] International Medical Device Regulators Forum (IMDRF), "Software as a Medical Device (SaMD): Key Definitions," IMDRF/SaMD WG/N10 FINAL:2013, Dec. 2013. [Online]. Available: https://www.imdrf.org/documents/software-medical-device-samd-key-definitions. [Accessed: 22-Jun-2026].

[2] Y. C. Kudva, R. E. Carter, C. Cobelli, R. Basu, and A. Basu, "Closed-loop artificial pancreas systems: Physiological input to enhance next-generation devices," *Diabetes Care*, vol. 37, pp. 1184–1190, 2014.

[3] L. Nurgalieva, D. O'Callaghan, and G. Doherty, "Security and privacy of mHealth applications: A scoping review," *IEEE Access*, vol. 8, pp. 104247–104268, 2020.

[4] K. Fu and J. Blum, "Controlling for cybersecurity risks of medical device software," *Commun. ACM*, vol. 56, pp. 35–37, 2013.

[5] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel, "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses," in *Proc. IEEE Symp. Security and Privacy (S&P)*, Oakland, CA, USA, 2008, pp. 129–142.

[6] A. Papageorgiou, M. Strigkos, E. Politou, E. Alepis, A. Solanas, and C. Patsakis, "Security and privacy analysis of mobile health applications: The alarming state of practice," *IEEE Access*, vol. 6, pp. 9390–9403, 2018.

[7] C. M. Mejía-Granda, J. L. Fernández-Alemán, J. M. Carrillo-de-Gea, and J. A. García-Berná, "Security vulnerabilities in healthcare: An analysis of medical devices and software," *Med. Biol. Eng. Comput.*, vol. 62, pp. 257–273, 2024.

[8] S. A. Ebad, R. Zaghdoud, and A. Ben Miled, "Metrics and Quality Attributes of AI-Based Software as a Medical Device," *Engineering, Technology & Applied Science Research*, vol. 15, no. 4, pp. 24644–24651, Aug. 2025, https://doi.org/10.48084/etasr.11442.

[9] U.S. Food and Drug Administration, "Cybersecurity in Medical Devices: Quality System Considerations and Content of Premarket Submissions," Guidance for Industry and FDA Staff, Sep. 2023. [Online]. Available: https://www.fda.gov/regulatory-information/search-fda-guidance-documents/cybersecurity-medical-devices-quality-system-considerations-and-content-premarket-submissions. [Accessed: 22-Jun-2026].

[10] I. Sommerville, *Software Engineering*, 10th ed. Harlow, UK: Pearson Education, 2015, ch. 13, pp. 354–360.

[11] Dexcom, Inc., "About Dexcom," 2026. [Online]. Available: https://www.dexcom.com/en-ca/about-us. [Accessed: 22-Jun-2026].

[12] Qualys SSL Labs, "SSL Server Test," Qualys Inc., 2026. [Online]. Available: https://www.ssllabs.com/ssltest/. [Accessed: 22-Jun-2026].

[13] National Institute of Standards and Technology, "Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations," NIST Special Publication 800-52 Rev. 2, Gaithersburg, MD, USA, 2019.

[14] MDN Web Docs, "HTTP Observatory," Mozilla Foundation, 2026. [Online]. Available: https://developer.mozilla.org/en-US/observatory. [Accessed: 22-Jun-2026].

[15] OWASP Foundation, "Session Management Cheat Sheet," OWASP Cheat Sheet Series, 2024. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html. [Accessed: 22-Jun-2026].

[16] MITRE Corporation, "Common Weakness Enumeration (CWE)," 2024. [Online]. Available: https://cwe.mitre.org. [Accessed: 22-Jun-2026].

[17] American Diabetes Association, "Standards of Medical Care in Diabetes - 2024," *Diabetes Care*, vol. 47, suppl. 1, pp. S1–S321, 2024.

[18] IEC 81001-5-1:2021, *Health Software and Health IT Systems Safety, Effectiveness and Security - Part 5-1: Security - Activities in the Product Life Cycle*. Geneva, Switzerland: International Electrotechnical Commission, 2021.