



BOTSWANA COMMUNICATIONS REGULATORY AUTHORITY

WEBSITE APPLICATION SECURITY GUIDELINES

Reviewed & Approved By


Position	Head of CSIRT
Name	Mr. Emmanuel Thekiso
Signature	 <small>SIGNIFLOW</small>
Date	04 November 2025

Table of Contents

1	OVERVIEW.....	3
2	TERMS AND DEFINITIONS.....	4
3	INTRODUCTION.....	5
4	PURPOSE AND OBJECTIVE.....	6
5	SCOPE.....	6
6	COMMON WEBSITE APPLICATION ATTACK VECTORS AND VULNERABILITIES 7	
6	GUIDELINES.....	11
7	COMPLIANCE.....	16
8	GUIDELINES REVIEW.....	16
9	GUIDELINES ENFORCEMENT.....	16
10	FURTHER INFORMATION.....	16

1 OVERVIEW

- 1.1 Web application security involves safeguarding both the application and its data as well as web services such as Application Programming Interface (APIs) from various vulnerabilities and cyber-attacks. The components of include, authentication/authorization, input verification and validation, secure communication and storage, secure headers and regular patching/updates. All web sites should undergo security testing before being hosting on production infrastructure.
- 1.2 Web servers and websites are a target of numerous exploit attempts. When improperly secured and misconfigured, they introduce a significant risk to the internet users such as data theft, misinformation, denial of services, and many others. With the advent of AI, most website attacks can be easily automated and launched indiscriminately against thousands of targets at a time.
- 1.3 A web application Developer has the primary responsibility to securely develop (code) web applications, whereas Service Providers that accept to host client websites that do not conform to minimum security guidelines run the risk of attacks against their hosting environments.
- 1.4 Such risks and threats call for holistic approach to guidelines that will assist Developers and Service Providers to guard against insecure web applications online. Failing to put security measures in place may result in adverse consequences such as damaged

company reputation and relationships, revoked licenses, and/or legal action.

2 TERMS AND DEFINITIONS

2.1 The following terms and definitions apply for this document:

2.1.1 **Adversary:** the person who carries out a cyber-crime.

2.1.2 **API:** means the Application Programming Interface.

2.1.3 **Content:** is information stored by website database system and rendered to the user's internet browser. Information may be in the form of documents, images, audio/video clips, or any other type.

2.1.4 **Cybercrime:** criminal activities carried out by means of computers or the Internet.

2.1.5 **Information:** data or other knowledge that has value to the Authority.

2.1.6 **Information Security:** preservation of Confidentiality, Integrity and Availability of information, including Authenticity, Accountability, relevance, non-repudiation, and reliability.

2.1.7 **SSL Secure Socket Layer:** cryptographic protocols for web browsers and servers to provide authentic encryption of data communications security over the internet.

2.1.8 **Threat:** means any deliberate form of unjustified act or attempt to expose, alter, disable, destroy, steal or gain unauthorized access to or make unauthorized use of an asset, potentially exploiting and endangering an end user of a website, e.g. phishing, cross-site scripting, Denial of Service.

- 2.1.9 **User:** is a Person or Entity that utilizes a web application.
- 2.1.10 **Vulnerability:** weakness of a web application that can be exploited by one or more threats.
- 2.1.11 **Website:** client-server program run on the browser to access online services such as banking, retail, webmail, auction etc.
- 2.1.12 **Web server:** is a computer that runs websites, store, process and deliver web pages to the users. This intercommunication is done using Hypertext Transfer Protocol (HTTP).
- 2.1.13 **Website Security:** the process of protecting websites and online services against different security threats that exploit vulnerabilities in an application's code.

3 INTRODUCTION

- 3.1 Website applications are a gateway to accessing online services, and with the rise of web-based businesses and online products, the nature of and global architecture of the internet exposes websites to various threats and attacks of different complexities and levels from all over the world.
- 3.2 Websites are compromised because of improperly file and directory permissions, installing the web server with default settings, unnecessary services enabled (including content management and remote administration), lack of proper security policy, procedures and maintenance, improper authentication with external systems.

3.3 Some developers leave default accounts with their defaults or no passwords, misconfigurations in web server, operating systems and networks, bugs in server software.

3.4 One of the most common problems is misconfigured Secure Socket Layer (SSL) certificates or expired or no SSL certificate and weak encryption algorithm, use of self-signed certificates, default certificates and HTTP strict transport security not enforced and secure cookies not used.

4 PURPOSE AND OBJECTIVE

4.1 The purpose and objective of these Guidelines is to ensure that basic security safeguards are utilised and follow checklist and steps in protecting websites from exploitation, good software development hygiene, continuous patching of discovered vulnerabilities, using up-to-date encryption, with proper authentication.

4.2 Failure to adhere to simple best practice when administering a web server can result in security incidents.

5 SCOPE

5.1 The scope of this website/webserver guidelines is the protection of the Confidentiality, Integrity and Availability of information.

5.2 This document outlines the security Guidelines to be implemented in the website architecture design and finished products of web applications.

- 5.3 The Guidelines applies to all websites that belong to the Botswana registered domain names (.bw domain namespace) with records in the registry, and all websites redirected to .bw domain name. These Guidelines aim to guide administrators to adhere and reduce the success of various exploit attempts and to protect against the simplest vulnerabilities inherent to web servers.
- 5.4 It will especially apply to all entities/Registrars hosting .bw domains and Registrants who own the websites.

6 **COMMON WEBSITE APPLICATION ATTACK VECTORS AND VULNERABILITIES**

- 6.1 Attackers can exploit website and remotely gain control of the organisation's infrastructure, have access to confidential information, and even disrupt the overall services. The following are common website attacks:

6.1.1 **Man-in-the-middle attacks (sniffing):** attacker have access to information by intercepting and altering communications between an end user and the webserver. The attacker acts as proxy such that all communications between the user and the webserver passes through him.

6.1.2 **Phishing attacks:** an attacker tricks users into submitting login credentials in a website that looks legitimate or redirects the user to a malicious site hosted by an attacker web service through a shared link.

- 6.1.3 **Website defacement:** - The attacker maliciously attacks the visual appearance of a web page by inserting or substituting pages with malicious content. Defaced website pages may expose visitors to offensive content, propaganda or misleading information. Attackers use a variety of methods such as malicious code injections or file traversal vulnerabilities to access a legitimate site to deface it.

- 6.1.4 **Webserver misconfigurations:** this is the configuration weakness in a web infrastructure that can be exploited to launch various attacks on the web server such as enabling directory listing, improper permissions in the *config* or *env* files, and use of default credentials and ports.

- 6.1.5 **HTTP response splitting attack:** - involves adding header response data into field so that the server split response data into two responses. The attacker can redirect the user to malicious website whereas other responses will be discarded.

- 6.1.6 **Web cache Poisoning attack:** - attacker forces the webserver cache to flush its actual cache content and replaces it with malicious records.

- 6.1.7 **SSH Brute Force Attacks:** - SSH is used to create an encrypted communication tunnel between the host and server to transmit unencrypted data over an insecure network. Attackers can brute force SSH login credentials to gain unauthorized access to the SSH tunnel, and then transmit malware and exploits to the victim once inside the host server.
- 6.1.8 **Website password cracking:** attackers try to exploit weakness to hack privileged user passwords. The most common passwords are root, toor, administrator, admin, demo, guest, Qwerty, and other technology-specific default credentials. Attackers use different methods such as social engineering, spoofing, phishing, or stealer malware.
- 6.1.9 ***Denial of Service (DoS) & Distributed Denial of Service (DDoS):*** flooding a targeted web server with illegitimate traffic such that the server is unable to effectively process legitimate user requests and disrupts the normal functioning of the server. DoS attacks originate from a single source/system, whereas multiple systems such as botnets are used to launch a DDoS attack.
- 6.1.10 ***Cross-Site Scripting:*** injecting a script into the page output of a web application causing the browser to treat the malicious script as part of the page, therefore running a malicious script.

- 6.1.11 **Broken Authentication:** when a web application's authentication mechanisms are poorly implemented or flawed, allowing attackers to bypass authentication processes, impersonate users, or access unauthorized resources. An example is using insecure credential storage such as weak hashing algorithms (e.g., MD5).
- 6.1.12 **Insecure Deserialization:** tampering with data objects during the process of converting serialized data back into objects usable by the application (deserialized) which could lead to serious implications such as remote code execution.
- 6.1.13 **Memory Corruption (Buffer Overflow):** A buffer overflow vulnerability occurs when a when a program writes more data to a buffer (a temporary storage area in memory) than it can hold. The excess data corrupts nearby space in memory and may alter other data. As a result, the web server might report an error or behave differently such as leaking sensitive information stored in memory. Improper input validation such as accepting large strings without checking their length may lead to this attack.
- 6.1.14 Other website application critical risks can be found in the standard awareness document for developers and web application security – OWASP Top 10.

6.1.15 **Clickjacking:** occurs where an attacker tricks users into interacting with hidden or unintended elements on a webpage. The adversary may overlay malicious content using multiple transparent or opaque layers (iFrames) over legitimate website elements to deceive users into performing actions they did not intend, such as clicking a button, submitting a form, or enabling permissions. Websites that do not implement protections like *X-Frame-Options* or *Content-Security-Policy* headers are vulnerable to being framed and abused.

6.1.16 **Using Components with Known Vulnerabilities:** using components such as libraries and packages to implement a certain website functionality without first verifying their legitimacy, or by using deprecated versions of those components.

7 GUIDELINES

7.1 These guidelines are intended to ensure that all software developers and website domain hosting companies understand the dangers and risks of insecure website applications.

7.2 To ensure that basic security safeguards are adhered to: -

7.2.1 Authentication and Authorization (Access):

7.2.1.1 Implement strong password policies.

7.2.1.2 Implement Multi Factor Authentication (MFA) for login.

7.2.1.3 Utilize a “segmented application architecture” that implements a zero-trust model and allows only the desired behaviour while blocking the rest.

7.2.1.4 Limit login attempts to minimise the risk of credential stuffing.

7.2.1.5 Ensure accounts performing data queries have read-only access unless modifications are needed.

7.2.1.6 Implement logging and monitoring to capture failed login attempts, account privilege changes and/or other potentially suspect activities occurring on your web application or hosting environment.

7.2.1.7 Regularly review all user and service permissions.

7.2.2 Patching and Diligence:

7.2.2.1 Always run software and firmware patches, and package updates.

7.2.2.2 Disable insecure and unused ports, as well as default accounts, credentials and configurations.

7.2.2.3 Ensure that the only secure services such as HTTPS, SSH, SFTP, etc are running.

7.2.2.4 Employ Web authentication and encryption technologies such as SSL/TLS.

- 7.2.2.5 Implement secure installation processes and a system hardening process, ensuring that unnecessary, unused features or frameworks are removed or not installed at all.
- 7.2.2.6 Perform regular backups of Web content and occasional backups of operating system and application configurations.
- 7.2.2.7 Fortifying network architecture to avoid single point of failure e.g., implement load balancing and redundant internet to absorb normal to large volume of consumer traffic when needed.
- 7.2.2.8 Establish multiple routes to deliver website content (peering) and use anycast services to improve latency and resilience.
- 7.2.2.9 Implement DNS Security (DNSSEC) to protect DNS records from being tampered with by ensuring that responses to DNS queries are authentic and unaltered.
- 7.2.2.10 Protect against Application Layer attacks e.g. use Web Application Firewall.
- 7.2.2.11 Implement server-side input validation and sanitization checks.

7.2.3 Headers and Other Configurations:

7.2.3.1 Configure secure headers such as Content-Security-Policy (CSP), X-Frame-Options, X-Content-Type-Options, Strict-Transport-Security (HSTS), Permissions-Policy, Cross-Origin-Embedder-Policy (COEP), Cross-Origin-Opener-Policy (COOP), Cross-Origin-Resource-Policy (CORP), Cache-Control, Set-Cookie (with Secure and HttpOnly flags), and others.

7.2.3.2 Web servers should be configured to prohibit access to files that may not be intended for public consumption and do not make arbitrary directories.

7.2.3.3 Separate Web server content and related subdirectories from operating system and application directories.

7.2.4 **Change Control:**

7.2.4.1 Establish revision control mechanisms

7.2.4.2 Test change(s) and updates on a test system if available before making the change in the production environment.

7.2.4.3 Backup relevant information affected by the change prior to implementing the change document, all changes being made to the system, application, or web content.

7.2.4.4 Keep detailed audit trails for important transactions to prevent tampering or deletion.

7.2.4.5 Notify changes (includes description, contact person, date, and time of change etc.) to all parties potentially impacted by the alteration to configurations.

7.2.5 **Secure Storage:**

7.2.5.1 Validate all inputs to prevent injection attacks like SQL Injection.

7.2.5.2 Classify application data being processed, stored or transmitted by level of sensitivity, and apply controls accordingly.

7.2.5.3 Use separate databases for different applications to limit cross-application vulnerabilities.

7.2.5.4 Enforce encryption for data at rest especially when storing sensitive and Personal Identifiable Information, as well as passwords. You may replace sensitive data with tokens that are meaningless if intercepted and use salted hashing algorithms for password storage.

7.2.5.5 Encrypt all connections between the database and the application using protocols like TLS.

7.2.5.6 Employ key management infrastructure

7.2.5.7 Limit access to the database only from trusted IP addresses or domains.

7.2.5.8 Protect APIs used for database interactions with strong authentication and access control.

7.2.5.9 Disable caching for responses containing sensitive data and avoid storing data in temporary memory unnecessarily.

7.2.5.10 Use less complicated formats like JSON, avoid serialization of sensitive data.

8 COMPLIANCE

- 8.1 Any website hosted by Service providers that does not conform to these guidelines may lead to the takedown of the domain where the site poses as a risk and a source of malicious activities especially on the .bw domain name space, in accordance with the Acceptable Use Policy and the Registration Terms and Conditions found at nic.net.bw/legal policies.

9 GUIDELINES REVIEW

- 9.1 These guidelines shall be subject to review through consultation of stakeholders as and as when need be.

10 GUIDELINES ENFORCEMENT

- 10.1 These guidelines are not enforced on website owners, developers, and hosting companies. However, they are deemed appropriate to question the negligence of the owner of a website or any hosting service provider through which any suspected malicious activities may be deposited through a .bw domain.
- 10.2 Any Authorised Registrar or Registrant found to be in violation of the ccTLD policy may be subject to disciplinary procedures and/or legal action deemed appropriate by the Authority.

11 FURTHER INFORMATION

- 11.1 Further information, clarification and advice on these guidelines can be obtained from the CSIRT Unit email (info@cirt.org.bw).