Communications Security Establishment
Centre de la sécurité des télécommunications

CANADIAN CENTRE FOR CYBER SECURITY

Security Considerations for Your Website

MANAGEMENT

TLP:WHITE

Canada

FOREWORD

ITSM.60.005 Security Considerations for Your Website is an UNCLASSIFIED publication issued under the authority of the Head, Canadian Centre for Cyber Security (Cyber Centre).

For more information, you can contact our Contact Centre:

Contact Centre
contact@cyber.gc.ca
(613) 949-7048 or 1-833-CYBER-88

EFFECTIVE DATE

This publication takes effect on October 6, 2021.

REVISION HISTORY

Revision	Amendments	Date
1	First release.	October 6, 2021

ISBN 897-0-660-40569-8
CAT D97-4/60-005-2021E-PDF

OVERVIEW

Your organization's website is a critical component of your business. It provides access to your services and visibility of your products. However, cyber threats can compromise your website, which can harm your business functions, revenue, and reputation.

Websites are the gateway from the Internet into your organization. Threat actors can exploit vulnerabilities and misconfigurations to steal, alter, or delete sensitive information (e.g. vendor portals, customers data, sales leads, operational and financial information). Data breaches put you at risk of legal and regulatory issues and possible financial penalties. In addition, if your website is compromised, a threat actor may be able to target other organizations and individuals with whom you are affiliated.

This document introduces cyber security best practices that your organization should integrate into the design and maintenance of its website. In Section 2, we include a checklist of key measures that your IT decision makers and web development team managers can implement to improve the security of your organization's website. In the sections 3 through 9, we provide further details on the security considerations listed in the checklist.

Keep in mind that even if you address all the items on the checklist, it is impossible to eliminate risk entirely (i.e. residual risk remains). The security measures that you take depend on your organization's environment, requirements, and resources. You should scope and tailor the guidance in this document to meet your organizations' needs and accepted risk tolerance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

This document introduces cyber security best practices that your organization should integrate into the design and maintenance of its website. In Section 2, we include a checklist of key measures that your IT decision makers and web development team managers can implement to improve the security of its website. You can use this checklist in the following ways:

- **Decide whether to build or buy** by determining whether your organization has the capacity to build its website and identifying what components need to be outsourced;
- **Design your web services** with the appropriate security controls built in;
- **Review your operational website** to identify security vulnerabilities;
- **Procure web services** from providers who use appropriate security controls to protect your web services; and
- **Define commonly understood roles and responsibilities** to address the security of your website.

Following the checklist, this document describes each of the listed security considerations in more detail. Whether you develop your own website or purchase web hosting services, you should ensure that your website has controls in place to protect the confidentiality, integrity, and availability of data. Your organization is legally responsible for protecting all customer data that is processed on your website, even if your organization does not host the website.

Many Canadian businesses use some form of managed services for their business operations. If you choose to use a cloud service provider (CSP) or a managed service provider (MSP) to assist you with your web services, we highly recommend that you review the following publications: *ITSM.50.030 Security Considerations for Consumers of Managed Services* [1][1] and *ITSM.50.062 Cloud Security Risk Management* [2].

1.1 DATA SENSITIVITY

Before designing your website, you should identify high-value and sensitive data (e.g. personal, financial, business critical, or proprietary information) to ensure that you implement measures in your website design and architecture to protect it. Data sensitivity is measured by assessing the possible harm that could result from the inability to protect the confidentiality, integrity, and availability of information. Confidentiality protects information from unauthorized disclosure. Integrity protects information from unauthorized changes. Availability ensures that information is available when it is required.

When categorizing your data, you should assign sensitivity levels to each of these three areas (confidentiality, integrity, availability) to create a three-part data security profile. Data sensitivity is categorized as high (H), medium (M), or low (L). For example, consider a data set that has a data security profile of H/M/L. You can interpret the three parts of the data security profile in the following ways:

- A **data confidentiality compromise** (H) has a critical or prohibitive impact;
- A **data integrity compromise** (M) has a major impact; and
- A **data availability compromise** (L) has a moderate impact.

[1] Numbers in square brackets refers to resources cited in the Supporting Content section of this document.

Data classification can be time sensitive. For example, the sensitivity of election result data is higher the day of an election than the day after. Classify your data based on the highest level of expected injury that could result if that data is compromised. See Annex 1 of *ITSG-33 IT Security Risk Management: A Lifecycle Approach* [3] for more information on classifying and categorizing your data.

Data Group	Confidentiality	Integrity	Availability	Who Needs Access	Storage Location
Data group A	High	High	High	System administrator	Internal network
Data group B	Medium	High	Medium	Finance	Service provider
Data group C	Medium	Low	Low	Users Guests	System provider

Figure 1: Sample Data Security Profile

2 SECURITY CONSIDERATIONS CHECKLIST

The following checklist summarizes measures that you should implement when developing and maintaining your organization's website. When implemented, these measures work together to address common web application risks, as defined by the Open Web Application Security Project's (OWASP) *Top 10 Web Application Security Risks* [4]. We describe each of the security measures listed in more detail throughout this document.

Table 1: Checklist for Building or Procuring Secure Web Services

Section	Security Measures	Addressed (A), Developing (D), Not Applicable (NA)	Notes
Secure Architecture			
3.1	Segregate web service components: • Isolate your web server from your network • Isolate your application server and database		
3.2	Protect your Internet-facing web services: • Install a proxy server • Use a WAF		
3.3	Implement redundant components		
3.4	Isolate administrative interfaces		
Access Control			
4.1	Plan and test access controls		
Authentication Mechanisms			
5.1	Use strong password policy		
5.2	Enable multi-factor authentication (MFA)		
5.3	Have a secure account recovery process		
5.4	Use measures to protect passwords, such as: • Use Hypertext Transfer Protocol Secure (HTTPS) • Use hash functions and salt passwords • Consider password entry options • Do not hardcode database credentials and application programming interface (API) keys		
5.5	Use account lockouts, login delays, and completely automated public Turing tests to tell computers and humans apart (CAPTCHA) to prevent brute-force attacks		
Secure Session			
6.1	Use session management toolkits		

6.2	Create random session identifiers that adhere to a minimum length		
6.2	Use cookie security flags		
6.3	Store cookie data on a server		
6.4	Expire session data		
6.6	Add layers of session authentication (anti-cross-site request forgery [CSRF] tokens, reauthentication for higher risk operations)		
Input Validation			
7.1	Validate early, but coordinate		
7.2	Confirm expected input lengths		
7.3	Restrict input format as much as possible		
7.4	Ensure validation code is central		
7.5	Filter special characters		
7.6	Hide Structured Query Language (SQL) error messages from users		
7.7	Block multiple parameter instances		
7.8	Validate after encoding		
7.9	Secure file upload		
7.10	Test business logic		
7.11	Conduct penetration testing		
Configuration			
8.1	Turn off directory browsing		
8.2	Remove unnecessary web directory files		
8.3	Lock down web service components		
8.4	Disable browser caching of credentials		
8.5	Use vulnerability scanners		
8.6	Automate deployment		
Operations			
9.1	Monitor website activities		
9.2	Establish an incident response plan		
9.3	Establish patching strategy		
9.4	Promote security awareness		

3 SECURE ARCHITECTURE

Your web services receive and respond to requests from users on the Internet. To ensure the ongoing security of your web services and the sensitive data that you collect, process, and store, you must ensure that you design a secure architecture. In the context of this document, the term *architecture* refers to how your web services and their underlying components are arranged to provide service securely and effectively. A secure architecture incorporates principles such as segregation and redundancy to ensure your web service components are protected from compromises.

3.1 SEGREGATE WEB SERVICE COMPONENTS

When you design your web services, you should segregate, or separate, your web service components from one another. By separating them, you can ensure that if one component is compromised, the other components are protected. For example, if a threat actor successfully accesses a part of your infrastructure, they won't be able to access the other components.

A simple web service has the following components: a web server, an application, and a database. The web server accepts and formats requests from the client browser across the Internet and then passes these requests to the application for processing. The web server also receives responses from the application and presents them to the client browser (i.e. the presentation tier). The application conducts the main processing in the web service, applying business logic to requests (i.e. the application tier). The database is your persistent data repository, which houses your sensitive data (i.e. the persistent tier).

A simple design could have all components residing on a single web server. This design works from a functionality perspective, but it leaves your data vulnerable to threats. If the web server is compromised, and everything is stored on that one web server, then your data is compromised as well. To secure your architecture, use techniques such as physical separation (where possible), firewalls, security policies, and access controls. If you are using a service provider, you should identify how they separate web service components.

3.1.1 ISOLATE WEB SERVER FROM YOUR NETWORK

The web server is the front end of your web application. The web server is considered *web-facing*, meaning that it is visible and provides access to users over the Internet. Your web server communicates with the user's browser, collecting user input and presenting output to the user. Minimal data is processed by the web server. Instead, the user's data is passed to the application for processing.

Threat actors target web-facing components, such as your web server, because they are entry points to your network and data.

You should physically separate your web server from other web service components, including your application server and database. Put the web server on a different physical server and apply security controls between it and the rest of your network. You should also implement firewalls, intrusion detection systems (IDS) or intrusion prevention systems (IPS), security policies, and permissions to protect your network. If it is not feasible to physically separate your web server, you should implement compensating security controls to achieve a level of risk that is acceptable to your organization.

If using an MSP or a CSP to host your web services, physical separation may be difficult to achieve if the service provider has a multi-tenancy infrastructure. In a multi-tenancy infrastructure, multiple customers (i.e. you and other paying

customers) share the same physical infrastructure. In this case, accounts are separated by logical controls, such as with permissions.

In a traditional scenario, best practice dictates physically separating all the main service components by placing them on different physical servers and implementing physical controls, such as a firewall, between them.

In a shared tenancy infrastructure, the business case for physical separation may be more difficult to justify due to increased costs. However, it is best practice to physically separate your web server from other network components, such as your application server and database.

3.1.2 ISOLATE APPLICATION SERVER AND DATABASE

The application server is the back end of the web application. The data passes from the web server to the application server, where it is processed as per your application business logic (i.e. rules that determine how data is created, stored, or modified). The data is stored in a database or passed back to the web server. The database is the data repository for the web application.

You should isolate your application server and database server from each other. If a threat actor breaches your network and attacks the components in the network, you want to make it as difficult as possible for the threat actor to access sensitive data. As a best practice, you should also implement system-specific controls, such as applying permissions to the application server and the database and installing firewalls between the components.

Consider the relationships between back-end components: a database contains unstructured data, a content management system structures the data, and a search engine searches indexes to organize the data. By isolating these components and applying the proper security controls, you can prevent a threat actor from accessing everything if one of these components is compromised. For example, if a threat actor accesses the content management system server, they can see templates but not the data. If a threat actor accesses the database, they can only see unstructured data that is not easily readable. If the threat actor accesses search engine components but not the formatted data, they may see part of the indexing function but not the data that they are looking for.

You should focus on segregating the web application; it is a high-value target because it brings many of the backend components together. However, if segregating data is not feasible, you should implement additional protections. Examples include requiring authentication to each of the systems that provide data to the application, implementing *two-person integrity* (e.g. two people must log in to perform actions), enabling MFA, and monitoring system access.

3.2 PROTECT YOUR INTERNET-FACING WEB SERVICES

Threat actors target web-facing components because they serve as entry points to your network. Where possible, you need to identify and stop malicious requests before they reach your network.

You can protect your Internet-facing web services by using the following techniques:

- Insert a proxy server in front of your web services;
- Use a web application firewall (WAF), which an application firewall that filters, monitors, and blocks Hypertext Transfer Protocol (HTTP) traffic;
- Implement access controls and authentication methods (see sections 4 and 5); and

- Monitor and use an IDS or IPS (see section 9).

3.2.1 INSTALL A PROXY SERVER

A proxy server provides a layer of security by acting as an intermediary between your services and the requests from users. It can act like a firewall and filter for web requests. You should set up a proxy server in front of your web services as it can provide the following functions:

- **Authenticate users**: A proxy acts as a security control for authentication. It can authenticate users before they access the web server on your network.
- **Increase performance**: A proxy caches content for multiple customers so that commonly requested data can be accessed quickly.
- **Apply network access translation (NAT)**: A proxy hides your network from the Internet, reducing your attack surface.
- **Monitor:** A proxy can facilitate a known path of access for your customers, which is where you can focus your monitoring and intrusion protection.
- **Handle Secure Sockets Layer (SSL) Offloading**: Some proxy devices can also handle SSL offloading to remove the burden of encryption and decryption from web servers. By using purpose-built hardware accelerators, web proxies can perform SSL termination and acceleration more efficiently, which improves performance and enables web servers to focus central processing unit (CPU) cycles on handling web requests.

Define which of the above proxy services you need. If you require authentication services, you should consider requirements for certificate management and certificate lifecycling.

You may also choose to use an MSP for proxy services. In this case, the proxy service resides in the provider's infrastructure. Before you select an MSP, review which proxy services are offered.

3.2.2 USE A WAF

You should install a WAF between your web services and the Internet. A WAF is an application firewall that filters, monitors, and blocks HTTP traffic. It can inspect Transmission Control Protocol/Internet Protocol (TCP/IP) packets on the wire before they reach the web server. A WAF includes features such as the following examples:

- **Track sessions**: A WAF monitors session token information to detect tampering.
- **Slow down denial-of-service (DoS) attacks**: A WAF can identify indications of a potential DoS attack, giving your network an opportunity to react and minimize damage.
- **Patch**: A WAF can provide a short-term stop gap to patch web applications. If a vulnerability is found in a web application, it may take time to implement a fix because of your development and quality assurance processes. Once a problem is identified, you can set up the WAF rules to block an attack until you can implement the fix on the web application.
- **Remediate log deficiencies**: A WAF can address logging deficiencies in the web application. A WAF can inspect packets before the messages get to the web application. This capability can also offload some of the logging tasks from the application server.

Define which of the above WAF services you require. Note that configuring a WAF can be complicated. You should define WAF rules as specifically as possible. However, if you establish rules that are too strict, you risk blocking legitimate traffic or generating many false positives, which can clutter up your logs and make it difficult to detect real issues.

3.3 IMPLEMENT REDUNDANT COMPONENTS

You should build redundancy into your design to avoid a single point of failure to help you effectively return to operations and minimize the impacts if an issue occurs. All hardware components are subject to failure at some point. Redundancy (i.e. replication) is an architectural principle that ensures that multiple resources serve the same function. By implementing redundancies in your web service components, you can maintain the availability of systems and services and reduce the risk of data loss. In addition, redundancy can reduce the impacts of a DoS attack, which can target web-facing components to intentionally disrupt services. You should prioritize creating redundancies for web-facing components.

Your redundancy plan should focus first on web-facing components, such as proxy servers, but it should also consider internal components, such as web servers, load balancers, databases, and application servers. If load balancing is required, you should determine which load balancing distribution mechanisms are available to you (e.g. source IP, source port, quintuple) and verify that these mechanisms are compatible with your web servers and applications.

Ensure you back up your data regularly. It is not enough to create redundant components and back ups; you need to test them rigorously to ensure they work properly.

3.4 ISOLATE ADMINISTRATIVE INTERFACES

You should isolate your website's administrative interface. Administrative or privileged users have elevated access to your network. When assigning privileges to users, you should always consider the potential impact should their accounts be compromised. You should also have separate log-in interfaces for general users from administrative or privileged users. Login interfaces for web applications are Internet-facing, making them a target for attack. Issues with the user login interface could escalate quickly if an administrator uses that same interface. If something goes wrong with your website, it is your administrator that needs to get in to fix the issues.

You should isolate your website's administrator interface. In the case of remote work environments, administrative or privileged users should use a virtual private network (VPN) to log in to and access your network.

Additionally, you should store administrative user credentials separately from general user credentials, such as in a different database instance.

4 ACCESS CONTROL

Access control refers to the rules and the techniques used to ensure that entities (i.e. systems, processes, and people) only have access to the systems and data that they require to perform authorized functions. Access controls define who can access what resources on your website and restrict what information they can see and use. Access control is central to protecting the confidentiality, integrity, and availability of data. Whether you host your website, or a service provider hosts your website, your organization owns its access control policy.

4.1 PLAN AND TEST ACCESS CONTROLS

Application or network access requires access control planning. When defining access controls, you should apply the principles of least privilege and default deny. The principle of least privilege ensures that entities have the minimum access they need to perform their authorized functions. In the principle of default deny, requests are denied unless the source of the request is authorized by your organization as per the allow list.

Access control is complicated in a web application. There are multiple layers of mechanisms at play when resources or data are being accessed. The access control layers in your web application should include the following:

- **URL-based access control on the web server or web platform**: Disallow certain URL elements. If you only want access to your site from certain geographic locations, this may include geo-location protections.
- **File system and server permission**: To pass through a web server, filesystem permission and web server permission must be examined.
- **Application (business logic) access control**: Access control to define what the user can and cannot do in the application functions.
- **Data layer access control**: Complements application access control by controlling access to database elements.
- **Application (presentation layer) access control**: Controls the limits of the application that a user can view.

When designing access controls for each layer, you should always consider defence in depth and the user's identity and role. Define the roles and the functions that each role can perform. For each function, you should be as specific as possible when defining what the entities can do and see (e.g. read, update, or delete data), and what data these entities can access in your database. You should create an access control matrix that uses rules or roles to link resources to objects. For example, create a table that distinguishes administrative roles, such as system administrators and application administrators. You should create an additional table to describe what data they can see. It is crucial that you test your access controls to ensure they are implemented properly.

Ensure that you define requirements for reviewing and updating privileges regularly (e.g. when a user changes roles or leaves the organization).

5 AUTHENTICATION MECHANISMS

Authentication refers to the mechanisms to verify a user's identity. Authentication works with authorization; once a user proves their identity, the user is then authorized to ensure that they only have access to the resources they need.

With authentication requests coming from the untrusted Internet, you need to securely verify identities to protect your website and sensitive information. Your approach to web application authentication should include the following elements:

- Strong password policy;
- Secure account recovery;
- Certificate authentication;
- Built-in password protection mechanisms; and
- Enable account lockouts, login delays, and CAPTCHA.

5.1 ESTABLISH A STRONG PASSWORD POLICY

Establish a strong password policy that includes a minimum number of characters and the expiry dates of passwords. You should not enable password hints as they make it easier for threat actors to guess passwords. We find that the following standards produce the best deterrent effect:

- Passphrases should be at least 4 words and 15 characters long;
- Passwords should be at least 12 characters long; and
- Passcodes and personal identification numbers (PINs) should only be used if passphrases and passwords cannot be used.
  - Use randomly generated PINs where available.

Be sure to rigorously test your password implementation. For more password best practices, see *ITSAP.30.32 Best Practices for Passwords and Passphrases* [5].

Credential stuffing attacks are becoming more common. In a credential stuffing attack, threat actors use previously stolen credentials to attempt to log in to other websites and systems until matches are found. Credential stuffing attacks work because people tend to reuse their passwords across multiple websites.

To protect your customers against credential stuffing, encourage them not to reuse their passwords and prevent them from using passwords that are known to have been leaked. Do not allow users to enter their email addresses as their usernames. Allowing email addresses as usernames gives threat actors a plausible username and password combination.

There are online resources available that identify compromised passwords, such as the website, "Have I Been Pwned" and its Pwned Passwords service. Through this service, users can see if any of their information has been subject to a data breach. This service was created after the National Institute for Standards and Technology (NIST) released guidance specifically recommending that user-provided passwords are checked against existing data breaches.

With password requirements becoming more complex, users may have difficulties remembering complex passwords without writing them down, which is a risk to security of the passwords. Increasingly, websites use password management services supplied by vendors, which can not only store and manage passwords but also generate complex passwords for users. For security tips on password managers, refer to *ITSAP.30.025 Password Managers Security* [6].

5.2 ENABLE MFA

We highly recommend enabling MFA for all types of accounts (e.g. administrators, users). MFA requires users to have at least two different authentication factors to log in to their accounts. As identified in our publication, *ITSP.30.031 v3 User Authentication Guidance for Information Technology Systems* [7], there are three main types of authentication factors:

1. Something you know (e.g. a PIN, password, or passphrase);
2. Something you have (e.g. a hard token, smart card); and
3. Something you are (e.g. a biometric like a fingerprint).

Note: There is increasing recognition for additional authentication factors to further strengthen authentication, such as your location and user behavior (i.e. what you are doing).

It is easier to implement a type 2 authentication factor (i.e. something you have) because it doesn't require the user (or browser component) to know or remember the factor (e.g. a password or PIN), and you can fit the input into an HTML form. However, it may not be feasible to issue hardware token to all website clients; instead, you could issue hardware tokens to special users or administrators.

Many vendors have proprietary solutions for MFA tokens. In most cases, the vendor provides the API for different development languages that the developer integrates into the authentication application. If you use one of these solutions, ensure that it uses at least two of the authentication factor types listed above. You should also ensure that the vendor can support electronic distribution of the second factor.

MFA is not a perfect solution, but this measure requires threat actors to work harder to succeed at hacking an account. It is possible for threat actors to hijack user sessions or compromise more than one medium to defeat out-of-band authentication mechanisms. However, by using per-use tokens and expiring tokens, you can limit or eliminate the window of time in which a threat actor can use the targeted login credentials.

Additionally, MFA does not prevent person-in-the-middle attacks from occurring. A threat actor can also send a phishing message with a link to a spoofed log-in page to trick the recipient into entering their credentials.

5.3 HAVE A SECURE ACCOUNT RECOVERY PROCESS

You should define a secure account recovery process. If a customer loses one or both of their authentication factors, this recovery process enables them to reset their passwords and log into their accounts. You must carefully implement account recovery to minimize the risks of threat actors masquerading as legitimate customers. You can also notify the customer any time abnormal log-in activity occurs (e.g. from a new device or location).

Your account recovery process should include the following steps, at a minimum:

1. Ask the user to identify themselves (e.g. username, customer number, emails address);

2. Ask private security questions;
   - Choose these questions carefully and avoid asking questions that rely on answers based on publicly available information. Instead, use questions relating to previous transactions or other account details;
3. Generate an email containing a time-sensitive, one-time link;
   - This email provides a second authentication factor by communicating with the user on another medium;
4. Allow the user to enter a new password;
   - The user must meet the password complexity requirements;
   - The user enters the new password twice to validate the change; and
5. Alert the user that the password or account details have changed.

5.4 USE MEASURES TO PROTECT PASSWORDS

In addition to a strong password policy, you should apply additional measures to protect passwords. When users log into your website, do not allow passwords to be sent in plaintext. Instead require the use of HTTPS and hashing. You should review your coding practices to ensure that credentials and API keys are never allowed to be hardcoded into your web services.

5.4.1 USE HTTPS

You should require the use of HTTPS for data in transit. Although HTTP basic authentication is a widely used authentication scheme and is supported by many browsers and servers, it has a critical flaw. HTTP sends data, including credentials, over the Internet in plaintext, which allows anyone to read that data. This plaintext may show in the URL line. HTTP is vulnerable to relay attacks and eavesdropping. HTTPS is not a separate protocol from HTTP. HTTPS requires Transport Layer Security (TLS)/SSL tunnel encryption over the HTTP protocol. TLS/SSL encrypts authentication data to prevent relay and eavesdropping attacks. We recommend using TLS 1.3 or better. For additional recommendations on encrypting data in transit, see *ITSP.40.062 : Guidance on Securely Configuring Network Protocols* [8].

5.4.2 USE HASH FUNCTION AND SALT PASSWORDS

As per *NIST Special Publication 800-63-3 Digital Identity Guidelines* [9], you should hash and salt passwords using a suitable one-way hash function.

A hash function inputs the password, a salt value, and a cost value to generate a password hash. A salt value refers to a random string of characters that is added to the password and is known only to the server. Salt values should be at least 32 bits in length. Passwords are salted before the hash function takes place to make the password longer and more difficult to guess even if a threat actor intercepts the password hash file on the server. The purpose of the cost value is to make a password guessing attack expensive or cost-prohibitive for threat actors.

Use only approved hash algorithms as per ITSP.40.062 [8] and a cryptographically secure random bit generator to generate salt values. Further recommendations are given in NIST SP 800-63-3 [9]. Do not store the hashed values and salt values in the same place on the server side. Storing them separately makes it more difficult for a threat actor to intercept both.

Hashing alone does not provide sufficient protection. You should also consider the following points:

- Encrypt stored hashes and use HTTPS to encrypt passwords in transit to the server;
- Isolate hash values, salt values and encryption keys from each other on the server; and
- Store client-side passwords in a form that is resistant to offline attacks.

5.4.3 CONSIDER PASSWORD ENTRY OPTIONS

To support the use of password managers, allow cut and paste functionality in password fields. You may also want to allow users to have the option to view passwords so that they can verify their entries.

5.4.4 DO NOT HARDCODE DATABASE CREDENTIALS AND API KEYS

In your coding best practices, before development begins, you should state that database credentials and API keys are not to be hardcoded. Because credentials are needed throughout the code, credential management and updates are complicated and prone to error when hardcoding is involved. Additionally, web applications access the backend database with database and API credentials. If one web application is compromised and its code is exposed with embedded credentials, then the threat actors can access your database.

Instead, you should put credentials in a separate environment and make run-time reference from the application. You should also mandate that all log-in pages use a standard API.

There are credential management software solutions available so that credentials are only accessed when needed. You should expect that customers may use these solutions to manage their credentials; do not block the cut and paste option at login.

5.5 USE ACCOUNT LOCKOUT, LOGIN DELAY, AND CAPTCHA

During a brute force attack, a threat actor repeatedly attempts to log in through trial and error until they find the correct credential combination. To prevent brute-force attacks, you should not allow users to use their email addresses as their usernames (refer to subsection 5.3). You should also apply some additional security controls, including account lockouts, login delays, and CAPTCHAs.

You should monitor failed login attempts to identify potential brute force attacks and alert your administrator as soon as possible. You can use a WAF to help with your detection and response activities.

With account lockouts, users have a maximum number of allowed login attempts before their accounts are locked. This threshold gives users or threat actors a finite number of chances to login before the account is frozen, forcing them to follow the account recovery process. Your developers should review the authentication mechanisms used. If using a basic or a digest authentication mechanism, you need to customize it to support account lockout. There are few vendors that provide authentication backends with built-in account lockout capabilities, including Lightweight Directory Access Protocol (LDAP), Active Directory (AD), and SQL Server.

Add a delay to each login attempt. When carrying out brute force attacks, many threat actors use automated tools. A delay of 0.5 seconds is usually enough to interrupt these tools.

You should also add a CAPTCHA, or a similar mechanism, to confirm that a user is not a robot if they fail to login after one or two attempts. For example, a CAPTCHA mechanism may ask the user to look at a picture and type the characters they see. You do not need to develop this mechanism. You can purchase it off the shelf and integrate it into your interface. You may want to reach out to CAPTCHA and alternative vendors to discuss the feasibility of using CAPTCHA mechanisms.

6 SECURE SESSIONS

A session is an exchange of information between two or more entities, such as two devices or a user and a web server. If sessions are not handled securely, threat actors can interrupt or hijack sessions to intercept data or impersonate authenticated users.

Session management is the process of initiating, controlling, maintaining, and ending the exchanges between two or more entities. During a session, a unique ID is tagged to all the HTTP requests made by a user from the time they log in until they log out. Session management enables the server to recognize the user between requests. When a user is authenticated, they are issued a session ID so that they do not have to authenticate for every request.

Session tracking preserves certain data between sessions. The server can track a user's data from one session to another (e.g. show their purchase history).

A cookie is a packet of data that contains a user's session information. When a user visits your website, the server sends a piece of information back to the user's browser. The browser stores this information and sends it back to the server in every subsequent communication to that site.

These concepts greatly help the functionality of websites, but they also increase the risks. Threat actors can steal or subvert user sessions using attacks like cross-site scripting (XSS) or CSRF. In an XSS attack, a threat actor compromises a web server and injects malicious code into a website. When users visit the website, their browsers execute this malicious script, putting cookies, session tokens, or sensitive information at risk. In a CSRF attack, users are tricked into executing unwanted actions in their browsers, such as logging out, downloading account information, or uploading site cookies.

6.1 USE SESSION MANAGEMENT TOOLKITS

Programming languages such as Hypertext Preprocessor (PHP), .NET and Active Server Pages (ASP) have session management features built in. There is no need to reinvent session management mechanisms unless you have an in-house specialist on session management. Using these established products makes it easier for you to test and update these mechanisms. You should review these toolkits to ensure they meet your application's objectives and that they can support future service enhancements. If you choose to use a toolkit, be sure to document the security configurations for repeatability.

Note: Because toolkits are not inherently secure, you should follow the guidance in subsections 6.2 to 6.6 to secure sessions.

6.2 USE RANDOM SESSION IDENTIFIERS THAT ADHERE TO A MINIMUM LENGTH

You should use randomly generated session identifiers to prevent threat actors from inferring the next session token. If session tokens are based on sequential values or use data like timestamps when created, then they can be inferred.

You should also implement a minimum length for session identifiers to prevent brute-force attacks. OWASP recommends a minimum length of 128 bits.

6.3 USE COOKIE SECURITY FLAGS

To ensure the security of cookies, use the following cookie flags:

- **Secure**: The *Secure* flag indicates to the browser that this cookie is to be sent only in secure connections, such as a TLS connection. Using TLS encrypts the cookie header so it cannot be read.
- **HTTPOnly**: The *HTTPOnly* flag tells the browser to disallow any access to the cookie from a JavaScript component. Some attacks, such as cross-site scripting, can control the victim browser's JavaScript and send the cookie to malicious sites.
- **SAME-SITE**. The *SAME-SITE* flag allows cookies to be sent only to the same domain where the cookies originated.

6.4 STORE COOKIE DATA ON A SERVER

During user authentication, your web server generates a session ID and sends it to the user's browser. All subsequent requests to the web server will include this value, which enables the server to recognize the user without having to authenticate them again. However, many web services also track sessions, including the user's individual preferences. You should store this sensitive session data on your web service servers and store only the minimum amount of session data that is stored in the user's browser. A user's browser is less secure, often due to vulnerabilities, (e.g. out-of-date software) which threat actors can exploit through XSS and CSRF attacks. In addition, laptops and mobile devices can be lost or stolen.

6.5 EXPIRE SESSION DATA

Threat actors can use discarded token data to impersonate users through CSRF. You should apply session timeout mechanisms to limit the window in which threat actors can use session credentials. Additionally, users may not log out after their sessions. You should implement a session timeout mechanism that occurs after a defined period (e.g. after 15 minutes) of user inactivity.

6.6 ADD LAYERS OF SESSION AUTHENTICATION

You should create layers of session authentication. Authentication is not a one-time activity in secure, modern web applications. Authentication should be considered to extend beyond the log-in page and through the entire session in which the user is interacting with the application. A form of authentication can be performed for every single request by using a combination of user behaviours, attributes, request timing, IP addresses, and browser details to determine the legitimacy of the requests. You should consider additional authentication mechanism when performing high-risk functions, as well as the MFA principles introduced in section 5.2.

You should implement anti-CRSF tokens each time a form is submitted. You should also ensure that the application reauthenticates the user when they are performing higher risk operations (e.g. requests to reset their password). This type of activity mitigates XSS and CSRF attacks. For more details on input validation, see section 7 below.

7 INPUT VALIDATION

Input validation is the process of verifying that users and applications can only input properly formed data, such as in fields, forms, or queries. Web applications are complex due to the number of moving parts between the source of the request in the client browser, the Internet, and your web infrastructure's back-end destination resource. All inputs on your website should be considered untrusted and controlled as much as possible. Input validation is an effective mitigation for web development risks.

You can validate inputs in the following areas of your web service:

- **User browser**: User request initiation;
- **WAF**: Filtering before request reaches your web service network;
- **Web server**: Presentation layer of your web services;
- **Application business logic**: Request processing; or
- **Database**: Persistent storage.

If you are using a service provider, ensure you review the web application and conduct your own functional testing. Input validation must be considered a part of your development process, and the web application owner is still responsible for the risk.

7.1 VALIDATE EARLY, BUT COORDINATE

As a best practice, you should validate inputs as early as possible in the processing process to reduce strain on your servers. Consider all areas where validation can occur. For example, consider credit card numbers. If you accept credit card numbers as an input, validation begins at the user's browser, as opposed to waiting for the application business logic (e.g. expected length). Validating at the user's browser prevents unnecessary noise going to your web infrastructure. However, validation at the user's browser is limited, and it can be bypassed at the user end.

You should complement browser validation with stronger validation methods. For example, web filtering provides a flexible add-on to drop invalid requests. Generally, web application validation is more secure. You should validate both the front end (i.e. form) and the backend (i.e. data that is submitted through a service) and don't just rely on one area.

7.2 CONFIRM EXPECTED INPUT LENGTH

Buffer overflow occurs when inputs for one function exceed the allotted buffer for data, impacting the web application in an unexpected way. This can impact your web services by potentially exposing sensitive data. Refer to NIST's National Vulnerability Database to see details in CVE-2014-0160 Detail [10], which describes the Heartbleed exposure in 2014.

You can minimize buffer overflow by imposing input limitations in line with expected input lengths and buffer capacity. Test buffer overflow prevention at run-time or in code reviews. Ensure that your web application can handle buffer overflow gracefully and that there are no unexpected alterations that occur in the database. You can also use Java, Perl, and PHP, which have built-in protections against buffer overflow.

7.3 ENSURE VALIDATION CODE IS CENTRAL

Input validation is prevalent and can occur almost anywhere in your code. As such, input validation can be quite difficult to test effectively. Centralize your validation code in functions rather than scattering it across your code. Clearly identify in your code where the input validation occurs. This enables reuse of input validation for inputs of similar type and facilitates effective testing across your code. You should establish this in your coding standards before development begins and require that code reviews verify effective input validation.

7.4 RESTRICT INPUT FORMAT AS MUCH AS POSSIBLE

In a free-form field, the user can enter any text they want. This type of input requires the most complex input validation because of the increased risk that something malicious can get through. To avoid this risk, you should restrict the format as much as possible and limit free-form client input where possible. Instead use limited option fields where possible, such as drop-down lists, radio buttons, or checkboxes, which are much easier to test. Do not ask for the same data more than once. Confirm inputs to ensure expected input type.

7.5 FILTER SPECIAL CHARACTERS

Filtering out special character combinations can protect you from several different types of cyber threats and attacks, such as malicious input injection. Where possible, use existing code frameworks and libraries, which have the benefit of maturity and community input, instead of creating these filters from scratch.

Operating system command line injection occurs when applications allow users to send command line arguments on the operating system with the privileges of service. This is particularly risky because threat actors can use malicious operating system code injection to escalate privileges or carry out DoS attacks. To mitigate this attack, you should validate input constraints, blocking special characters such as semi-colons (;), slashes (/), pipes (|), and database or application key words.

In HTTP header injection attacks, the user of the application is targeted as opposed to the web application itself. When the user logs in to a web application, the HTTP response reacts based on who the user is. For example, when logging in to a banking website, the HTTP response accepts inputs, such as user ID, and redirects the user to a particular page that contains their banking information. However, a threat actor can use special characters to create new header items, which can be used to spoof a user's cookies or redirect the user to an invalid site. This is often used to help facilitate XSS and CSRF. To mitigate this type of attack, you should validate input constraints, blocking special characters, such as "\r" "\n" and other HTTP header editing commands.

SQL injection occurs when a threat actor adds SQL in the input field to view or alter the data in a database. Through this attack method, a threat actor can alter or destroy a database; the threat actor only needs rudimentary SQL skills to carry out the attack. To mitigate this type of attack, you should block escaping mechanisms (i.e. quotation marks or double quotation marks) and special characters associated with SQL (e.g. DELETE, UPDATE) where possible and depending on web application permissions.

7.6 HIDE SQL ERROR MESSAGES FROM USERS

SQL error messages can provide a lot of data about your database, including queries to specific database tables and fields. Do not allow error messages to be displayed to users, as this can provide threat actors with insight into what your database looks like.

7.7 BLOCK MULTIPLE PARAMETER INSTANCES

You should block multiple parameter instances, where possible, to filter requests before they are processed by the application server. The implementation in the code depends on which application platform you use. Some platforms have implicit protection against multiple parameters of the same name. You can also constrain your inputs at the front end and use WAF to block certain keywords specific to SQL statements.

HTTP allows multiple parameters of the same name; as such, multiple values of the same parameter name can occur in a single request. Different server platforms (e.g. ASP.NET, JSP, PHP) handle this in different ways, using both values or just the first value or the second value, which can cause confusion. The application behaviour can change, and input validation can be bypassed.

You should test these validations during the development phase using code reviews and development testing.

7.8 VALIDATE AFTER ENCODING

You should set encoding on web pages and ensure all pages support the same encoding. Disable best-fit mapping where possible and normalize data via encoding before you validate the data. Your customers and software platforms can be global. As such, there can be a confusion between characters of different languages which look quite similar but mean something completely different to your application.

Use Unicode standard Unicode Transformation Format 8 (UTF-8) where possible. UTF-8 is the most common standard in the world for web pages and supports backward compatibility. Currently, the latest established version of the UTF-8 was v.12.1 Use standard normalization libraries. Most modern software providers use Unicode, as a universal language, to uniquely identify characters, as data is processed through different platforms and devices, to prevent corruption of data.

Conduct testing as part of the development process, using code reviews as well as run-time testing.

7.9 SECURE FILE UPLOAD

File upload is a form of input into your web application. In many instances, file upload can become a part of your site's content. There is a risk that the upload could include executable content or some type of malware. Additionally, an upload that is larger than expected can cause availability issues for your web application.

You should investigate why file upload is needed and if there are alternative forms that can acquire this data, such as form text and text area fields. Although these alternative forms require more input from the end user, they enable safer storage. Additionally, it is easier to perform searches on the data when it entered in these forms because the content is consistent (i.e. text-based). For example, consider a job-hunting application. Form-based input provides more benefit for both the

reviewer, as they can run searches on keywords, and the applicant, as they don't have to worry that reader software can't parse the content of their resume.

If file upload is deemed necessary, you should filter file upload based on file extension and size limits and analyze the content. You can use the "file" command, which checks for file types and understands the type of content inside. To extract the metadata or content for closer inspection, you can also use additional tools and libraries to extract the file in text or Extensible Markup Language (XML) format. For graphics, you can use tools to strip out metadata and detect abnormalities.

7.10 TEST BUSINESS LOGIC

Business logic refers to your organization's business decisions and how these decisions are translated into coding logic on the application server. Risks may be introduced if the business decisions are unclear or are not translated effectively to the coding logic. Business logic flaws can allow improper inputs to enter your space, which may cause a variety of problems.

You should clearly define your requirements and use cases, including misuse cases. When defining security requirements, look at the business process in its entirety. Test the business logic during and after the development phase to address use and misuse cases. You should ensure that the use cases address any instances of concurrency, which is when multiple processes need to access the same resource at the same time.

7.11 CONDUCT PENETRATION TESTING

You should conduct penetration testing to determine how resilient your input validation is to malicious activities. Penetration testing, or ethical hacking, involves testers who are authorized to attempt to find and exploit security vulnerabilities. There are different types of penetration testing (e.g. testers are given more or less information about your systems); however, your testers should not be a part of your development team so that they can simulate threat actor behaviours.

8 SECURE CONFIGURATION

Secure configuration is a broad topic, especially when considering security configurations on individual technologies. This document focuses on general configurations specifically for securing web services. If you are looking for recommendations on secure configurations for specific technologies, you should contact the vendor. Vendor-recommended security configurations are usually a good baseline that you can apply and then customize to your organization's needs. For additional information on generic secure configuration best practice, see the various publications available on our website (www.cyber.gc.ca).

8.1 TURN OFF DIRECTORY BROWSING

To prevent leaking potentially sensitive information to users, turn-off directory browsing on your web server. Directory browsing occurs when you can see and access the folders and files that create the website instead of seeing the website as it is meant to look to users. These folders can contain information that is not meant to be viewed by the public, including images, scripts, or backups. Directory browsing is usually implemented on a per-directory basis, and you need to turn off directory browsing in each directory.

You should also create an empty index.htm file in each directory. If the base file, like index.htm or index.php is not available, then by default, anyone can see all the files and the sub-directories listed in the browser. Test each directory and sub-directory.

8.2 REMOVE UNNECESSARY WEB DIRECTORY FILES

As a part of your secure deployment policy, you should review your web directories before and after deploying your website to mitigate the risks associated with data leaks. Web directories can contain information that is not meant to be viewed by the public, including images, scripts, or backups.

You should also define requirements for life cycling web directory files. You should also remove unnecessary web operation files from the web directories. Focus on old version files (multiple default.asp files), source code (.bak), backup files (could contain passwords), and source control files (.svn and .git). To make these files easier to identify, you can turn on *last access time* on the file system, which enables you to gain a visual of files that used seldomly, if ever. Verify during your code reviews.

8.3 LOCK DOWN WEB SERVICE COMPONENTS

To decrease your threat surface, you need to lock down various components of your web services.

You should prioritize locking down, or harden, your web service hosts by removing unused ports, files, and code, and lifecycling unused accounts. If your web servers are left exposed, there is an increased risk that threat actors can exploit these ports and services. These measures can supplement the protections offered through firewalls and security policies.

You should also remove any unnecessary plug-ins in your Configuration Management Systems (CMS). The more plug-ins you have, the greater your attack surface is, especially if the plug-ins are not official.

Additionally, harden the SQL server and remove any unused stored procedures and default accounts in your database. Monitor the SQL server's outbound connection to detect intrusion.

8.4 DISABLE BROWSER CACHING OF CREDENTIALS

Browsers offer to save user passwords for convenience. However, password storage on the user's browser is not secure. If the user's mobile device or laptop is stolen and the threat actor accesses the website, the user's credentials are automatically displayed. When coding this in an HTML form, ensure to set the *autocomplete* element to *off*, which tells the browser to turn off credential caching for this form.

8.5 USE VULNERABILITY SCANNERS

You should integrate vulnerability scanning into your secure deployment process. Tools such as the open source OWASP Zed Attack Proxy (ZAP) are commonly used to identify vulnerabilities specific to web applications. You can use these tools to identify vulnerabilities, which can also be used by penetration testers to identify entry points into your web services.

8.6 AUTOMATE DEPLOYMENT

Managing secure configurations on all the moving parts of your website can be complicated. Automating the environments is a best practice as it ensures they are built in a repeatable (e.g. can be redeployed if needed) and automated way (i.e. no human intervention). You should define the roles and responsibilities for carrying out configuration management and verification.

You can use security configuration templates and containers. To lower your attack surface, you should containerize to just an image based on a web server. By using templates and containers, you can ensure that new web server versions do not have zero-day bugs. During deployment remember to disable any remote login capabilities so that Secure Shell (SSH) is not allowed in production.

9 SECURE OPERATIONS

Once you establish your web services, you need to continuously maintain their security. Your monitoring activities should evolve as technology changes and as you discover new cyber threats. Review your access control models periodically to ensure that they reflect account access changes (e.g. as personnel changes occur). You should revalidate inputs when you make changes to your website and review your security configurations for every new release from the vendor.

Threat actors may pose as your organization to target your clients; phishing, ransomware, and social engineering are very common attacks. Threat actors can provide false information to solicit information from your customers or direct them to a spoofed, or false, website or a link that injects malicious code into their systems. Spoofing can negatively impact your web presence and reputation. Phishing, ransomware, and social engineering are very common attacks.

9.1 CONDUCT MONITORING ACTIVITIES

Define your monitoring strategy to identify potential attacks and anomalies specific to web application risks referenced in the OWASP Top 10 [4]. When conducting monitoring activities for your web application, consider the web service architecture components referenced in Section 3 of this document. Monitor your network traffic, as well as any unsuccessful connection attempts. You should analyze logs and notify the appropriate parties of any unusual activities in a timely manner to ensure an effective response.

9.2 ESTABLISH AN INCIDENT RESPONSE PLAN

Establish an incident response plan, including incident playbooks, to define the actions that your organization needs to carry out when an incident is detected. For example, you should include steps for identifying, communicating, containing, and remediating incidents. You should clearly define the roles and responsibilities for incident response, especially if you have multiple teams or organization involved to support your monitoring requirements. Test this plan rigorously to ensure its effectiveness.

If you are working with an MSP or a CSP, work with your service provider to establish a common incident response plan and ensure that you have access with the appropriate data and are notified appropriately. Be sure to coordinate testing activities with your service provider. You should identify the information and the reasonable turnaround times that will enable you to respond appropriately.

For a full list of security monitoring capabilities, see the security control catalogue in Annex 3a of ITSG-33 [3], specifically the auditing and accountability [AU] and incident response [IR] security control families.

9.3 ESTABLISH A PATCHING STRATEGY

Patch management is your process for acquiring, testing, and installing patches and upgrades on your systems. Your patch management strategy for your web applications will be similar to the patching strategy for your other infrastructure. However, you need to consider and prioritize the architecture components that are exposed to the Internet.

You may want to subscribe to vulnerability and application feeds (e.g. Mitre Common Vulnerabilities and Exposures [CVE] list) so that you can identify vulnerabilities as they become known. There are several well-known and low-cost tools, that you can use to scan for vulnerabilities in your network. Be sure to consider vulnerabilities in library dependencies and containers as well. Another option is to scan for old or outdated software components to update. If your organization cannot implement patches quickly, look at your firewall rules to reduce risks.

If you use managed or cloud services, work with your service provider to apply security patches and updates in a timely manner that meets your requirements. For cloud services, the entity responsible for patching depends on your cloud service model (e.g. in a software-as-a-service model, the CSP is responsible for updating and patching).

9.4 PROMOTE SECURITY AWARENESS

As technologies and cyber threats evolve, you can continue to protect your systems, information, and customers from compromises by promoting security awareness. Promote best practices like not clicking on links or downloading files by changing the way that you communicate with your customers. For example, if you need to contact a customer about their account, do not include a link or a downloadable file in the communication. Instead, direct the customer to log in to your website or contact you via the contact information listed on your website. You can recommend that they verify any communications they receive from you and report any suspicious activity.

You can also direct customers to our Get Cyber Safe campaign material, which promotes individual Canadians' security awareness.

10 PARTNERSHIPS

The advantages and challenges of building secure websites are shared by all organizations that have a presence on the Internet. Below are some resources that you can use to develop a secure website.

10.1 TALK TO YOUR PEERS

In any industry sector, the issues and concerns you are dealing with, are not necessarily unique. Understandably, it may not be an easy scenario to discuss security issues with competitors. However, keep an eye open for opportunities to discuss product and vendor options, as well as development challenges. You may be pleasantly surprised by how much you can learn from others' experiences.

10.2 OWASP

The OWASP Foundation is a non-profit foundation that aims to improve the security of software. You can review the OWASP Top 10 [4] to review their list of critical security risks to web applications.

10.3 CANADIAN ANTI-FRAUD CENTRE

If your organization is the target of fraud, such as a threat actor posing as your organization, contact your local police and file a report online through the Canadian Anti-Fraud Centre's Fraud Reporting System.

10.4 CONTACT US

For more information on cyber security, visit our website (cyber.gc.ca) or contact our Contact Centre:

Contact Centre
contact@cyber.gc.ca
(613) 949-7048 or 1-833-CYBER-88

11 SUPPORTING CONTENT

11.1 LIST OF ABBREVIATIONS

Term	Definition
AD	Active Directory
API	Application programming interface
ASP	Active Server Pages (programming language)
AU	Auditing (security control family)
CAPTCHA	Completely automated public Turing tests to tell computers and humans apart
CMS	Configuration Management System
CPU	Central processing unit
CSRF	Cross-site request forgery
CSP	Cloud service provider
CVE	Common vulnerabilities and exposures
DoS	Denial-of-service attack
GC	Government of Canada
H	High (in terms of data sensitivity)
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDS	Intrusion detection system
IP	Internet Protocol
IPS	Intrusion prevention system
IR	Incident Response (security control family)
IT	Information Technology
L	Low (in terms of data sensitivity)
LDAP	Lightweight Directory Access Protocol
M	Medium (in terms of data sensitivity)
MFA	Multi-factor authentication
MSP	Managed service provider
NAT	Network access translation
NIST	National Institute of Standards and Technology (United States)
OWASP	Open Web Application Security Project
PHP	Hypertext Preprocessor (programming language)

Term	Definition
PIN	Personal identification number
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
TLS/SSL	Transport Layer Security/Secure Sockets Layer
URL	Uniform resource locator
VPN	Virtual private network
WAF	Web application firewall
XML	Extensible Markup Language
XSS	Cross-site scripting
ZAP	Zed Attack Proxy (OWASP security scanner)

11.2 GLOSSARY

Term	Definition
Access control	Certifying that only authorized access is given to assets (both physical and electronic). For IT assets, access controls may be required for networks, systems, and information (e.g. restricting users on specific systems, limiting account privileges).
Administrative privileges	The permissions that allow a user to perform certain functions on a system or network, such as installing software and changing configuration settings.
Architecture	The arrangement of web services and their underlying components to provide secure and effective service.
Authentication	A process or measure used to verify a user's identity.
Availability	The ability for the right people to access the right information or systems when needed. Availability is applied to information assets, software, and hardware (infrastructure and its components). Implied in its definition is that availability includes the protection of assets from unauthorized access and compromise.
Code injection	Introducing malicious code into a computer program by taking advantage of a flaw in the program or in the way it interprets data by users.
Compromise	The intentional or unintentional disclosure of information, which adversely impacts its confidentiality, integrity, or availability.
Confidentiality	The ability to protect sensitive information from being accessed by unauthorized people.
Cookie	A packet of data that contains a user's session information.
Cyber attack	The use of electronic means to interrupt, manipulate, destroy, or gain unauthorized access to a computer system, network, or device.
Denial-of-service attack	Any activity that makes a service unavailable for use by legitimate users or that delays system operations and functions.
Encryption	Converting information from one form to another to hide its content and prevent unauthorized access.
Hashing	The process of using a mathematical algorithm against data to produce a numeric value that is representative of that data [10].
Host website	Stores all files need to run a website and connect it to the Internet.
Input validation	The process of verifying that users and applications can only input properly formed data, such as in fields, forms, or queries.
Integrity	The ability to protect information from being modified or deleted unintentionally or when it is not supposed to be. Integrity helps determine that information is what it claims to be.
Least privilege	The principle of giving an individual only the set of privileges that are essential to performing authorized tasks. This principle limits the damage that can result from the accidental, incorrect, or unauthorized use of an information system.
Multi-factor authentication	A form of user authentication that requires two or more different authentication factors to verify a claimed identity. The three most commonly recognized factors are: (1) something you know (e.g. a password), (2) something you have (e.g. a physical authentication token), and (3) something you are (e.g. a biometric).
Penetration testing	A test in which ethical hackers are authorized to attempt to find and exploit vulnerabilities on an organization's networks or systems.

Term	Definition
Redundancy	An architectural principle that ensures that multiple resources serve the same function so that the availability of systems and services can be maintained.
Residual risk	The likelihood and impact of a threat that remains after security controls are implemented.
Session	An exchange of information between two or more entities, such as two devices or a user and a web server.
Session management	The process of initiating, controlling, maintaining, and ending the exchanges between two or more entities. During a session, a unique ID is tagged to all the HTTP requests made by a user from the time they log in until they log out. Session management enables the server to recognize the user between requests.
Two-factor authentication	A type of multi-factor authentication used to confirm the identity of a user.
Two-person integrity	A requirement for at least two authorized persons, each capable of detecting incorrect or unauthorized security procedures with respect to the task being performed [10].
Virtual private network	A private communications network usually used within a company, or by several different companies or organizations to communicate over a wider network. VPN communications are typically encrypted or encoded to protect the traffic from users on the public network carrying the VPN.

11.3 REFERENCES

Number	Reference
1	Canadian Centre for Cyber Security. *ITSM.50.030 Security Considerations for Consumers of Managed Services*. October 2020.
2	Canadian Centre for Cyber Security. *ITSM.50.062 Cloud Security Risk Management*. March 2019.
3	Canadian Centre for Cyber Security. *ITSG-33 IT Security Risk Management: A Lifecycle Approach*. November 2012.
4	Open Web Application Security Project. *Top 10 Web Application Security Risks*. 2017.
5	Canadian Centre for Cyber Security. *ITSAP.30.032 Best Practices for Passwords and Passphrases*. September 2019.
6	Canadian Centre for Cyber Security. *ITSAP.30.025 Password Managers Security*. September 2019.
7	Canadian Centre for Cyber Security. *ITSP.30.031 v3 User Authentication Guidance for Information Technology Systems*. April 2018.
8	Canadian Centre for Cyber Security. *ITSP.40.062 Guidance on Securely Configuring Network Protocols* . August 2016.
9	National Institute of Standards and Technology. *Special Publication 800-63-3 Digital Identity Guidelines*. June 2017.
10	National Institute of Standards and Technology. National Vulnerability Database. "CVE-2014-0160 Detail". April 2014.
11	National Institute of Standards and Technology. Computer Security Resource Centre. "Glossary". N.D.