Software Security is Your Responsibility

August 2016

V1.0

Asia Pacific:	**The Americas:**	**Europe:**
Headquarters — Sydney, Australia	Headquarters — Chicago, USA	Headquarters — London, UK
Tel: +61 2 8907 0200	Tel: +1 630 874 7000	Tel: +44 1727 790300
Email: info@lansa.com.au	Email: info@lansa.com	Email: info@lansa.co.uk

Why invest in security?

Security has always been a concern for society. Individuals use guards, locks and keys to secure homes and offices. Organisations implement physical security mechanisms to control access to premises and identify employees. These security mechanisms might have been sufficient to protect information technology (IT) resources before the introduction of the internet. Now that everything is connected, attacking or hacking IT resources is a feasible and profitable activity. Attacks are more likely to be criminal or state sponsored than was the case in the past. Such attacks focus on economic gain (e.g. hacking and selling data) or business disruption for competitive advantage.

It is no longer safe to develop software without considering the whole of application security aspects of the software, e.g. the data it will use, the context in which it will run, the application architecture, and the development tools. Security is relative to the physical context of the IT resources and the nature of business operation. Security mechanisms that work in one organisation may be inadequate in another. Security is relative between organisations. If your organisation is more secure than your competition, hackers will attack the competitors before attacking you.

Don't view security as a collection or siloes. Security is a whole-of-business concern and includes physical assets, IT resources, data, and people (e.g. employees, contractors, suppliers and customers). Protecting IT resources requires constant vigilance, a security perspective, and an expanding list of security tools.

This document presents concepts, definitions and issues to consider when developing software. The information does not describe how to apply security measures, nor does it offer guidelines or best practice for developing secure applications.

Security then and now

Once upon a time perimeter security was sufficient to protect IT resources. Companies controlled the gateways and there was little need for security measures inside the organisation or the applications. Developers did not need to include security mechanisms in applications as authorisation occurred at the application entry point. Large applications such as Enterprise Resource Planning (ERP) software included role based access to parts of the applications as a way to manage user activity, but this security mechanism applied only in the application.

Now everything is accessible even when locked down and access controlled.

Companies require strong authentication (e.g. two factor), authorisation mapped and enforced, and secure applications, on top of network and perimeter security. Security mechanisms must consider the risks a company faces and security threats posed by employees, customers and partners. Developers must play a role in securing IT resources but their knowledge, skills and behaviour can be security weak points. Therefore, developers must ensure the code they write is not open to exploit. Application users and stakeholders have a responsibility to support developers by not asking for security shortcuts. If cost is your concern, remember that the cost of changes to applications escalates when applications reach the maintenance stage of the development cycle. Finding security flaws early in the development cycle reduces maintenance costs and removes potential security risks.

Security is your responsibility.

Threat, weakness and exploit examples

Name	Definition	Category
Application configuration errors	Examples are default passwords, hard coded user Ids and passwords, and testing features enabled by default.	Configuration
Application process weaknesses	Attackers can circumvent application processes.	Application logic errors
Authentication errors	Applications with flawed authentication processes allow attackers to bypass or compromise authentication and access data and/or application functions.	Authentication
Authorization failures	When applications fail to check a user's authorisation to use functions and/or data, attackers can exploit the resources by running functions or stealing data.	Authorization
Binary planting	Binary planting occurs when an attacker loads binary file(s) containing malicious code to a server. The malicious binary can steal data or delete files.	Malicious code or values
Broken or risky cryptographic algorithm	Attackers can exploit applications that use cryptographic methods and tools known to be insecure or easily decrypted.	Application logic errors
Brute force	Attackers use trial and error to hack an application by changing data after analysing application responses.	Application functionality manipulation
Buffer overflow	Attackers insert data into a memory buffer beyond the buffer size.	Application logic errors
Business logic errors	Attackers exploit flaws in an application's business logic or rules.	Application logic errors
Cache poisoning	Application must understand objects that they place in a cache. Caching malicious attack responses (e.g. malicious JavaScript) will amplify or spread the effect of an exploit.	Application functionality manipulation
Carriage return line feed injection	Attacker includes carriage return and/or line feed characters in input data.	Input validation

Name	Definition	Category
Catching null pointer exceptions	Programs should not catch null pointer exceptions. These exceptions indicate null pointer dereferences and require application code fixes.	Application logic errors
Code injection	Attackers insert malicious code into input data subsequently run by an application.	Injection
Command injection	Attackers insert operating system commands into input data subsequently run by an application.	Injection
Content security policy (CSP)	Compromising content security policy can allow attackers to use cross site scripting exploits.	Injection
Content spoofing	Attackers modify a website or present a counterfeit website purporting to be legitimate.	Spoofing
Cookie (or session) hijacking	Attackers monitor network traffic and extract unencrypted cookies. They use the stolen cookies to connect to the website and impersonate a valid user to obtains the user's details and/or operate the user's authorised business functions.	Authentication
Credential stuffing	Attackers use stolen or guessed credentials to gain access to an application.	Authentication
Credential/session prediction	Credential prediction occurs when an attacker guesses a user's credentials and operates an application by impersonating an authorised user.	Authentication
Cross-frame scripting (CFS)	Attackers embed a website into a frame on their own website and capture data from activity in the frame, e.g. using a key logger.	Injection
Cross-origin resource sharing (CORS)	Attackers insert values into an origin request HTTP header that force an application to provide resource content.	Injection
Cross-site history manipulation (XSHM)	Attackers use browser history to perpetrate exploits to gain information such as login status, resource mapping, user activity and parameter stealing.	Injection
Cross-site request forgery (CSRF)	Attackers send unauthorized commands to a website from a trusted user (also known as XSRF)	Spoofing

Name	Definition	Category
Cross-site scripting	Attackers gather data by inserting malicious code in a website by injecting scripts that run in a browser.	Injection
Cross-site tracing (XST)	Cross-site tracing occurs when an attacker uses HTTP TRACE to read HTTP headers. In an attack the server will send back all the data including the cookie and bypasses the HttpOnly cookie property.	Injection
Cryptanalysis	Attackers analyse a cryptographic cipher and break the cipher allowing them access to encrypted data.	Cryptographic exploits
Custom special character injection	Attackers can manipulate data when an application fails to validate non-printable (or special) characters and reserved strings used by the application.	Injection
Data validation weaknesses	Applications with inadequate input validation are vulnerable to attack.	Input validation
Denial of service	Attackers cause servers and/or network appliances to be unavailable or inoperable by flooding the servers or appliances with service requests.	Resource manipulation and depletion
Deserialization of untrusted data	Attackers can exploit applications that de-serialise data without knowing whether the data is trustworthy and does not include invalid data.	Application logic errors
Direct dynamic code evaluation, eval() injection	Attackers can exploit applications that fail to validate user input by passing code to an eval() statement with subsequent script execution.	Injection
Directory indexing	Attackers can exploit flaws in configuration files to discover web server directory content.	Configuration
Directory restriction error	Attackers can gain unauthorised access to files using relative paths or path traversal attacks when applications fail to enforce access policies.	Application logic errors
Domains and accounts	Allowing domains or accounts to expire.	Sensitive data protection
Double encoding	Attackers can encode user input data twice in hexadecimal format to avoid security and validation or force unexpected application behaviour.	Resource manipulation and depletion

Name	Definition	Category
Empty string password	Empty string passwords are easy to guess and susceptible to brute force attacks.	Application logic errors
Execution after redirect (EAR)	When developers use an HTTP redirect without a return after the redirect, and assume execution stops after the redirect, attackers can exploit the flawed assumption that execution stops after the redirect, when in fact, execution continues.	Malicious code or values
Failure to validate return values	Applications that fail to check return values are open to exploit. An attacker can manipulate return values and cause unexpected application behaviour.	Code quality
Flawed password recovery processes	Website allows an attacker to obtain or reset legitimate user passwords.	Authentication
Flaws or missing input validation	Flaws or missing input validation opens an application to injection and data manipulation attacks.	Input validation
Format string attack	Application input validation flaws can allow attackers to include format string parameters (%d, %s, %x) and execute commands.	Injection
Full path disclosure	Attackers can insert certain characters into a web page and obtain the path to the webroot of a server.	Injection
Hardcoding passwords	Attackers can use hard-coded passwords to compromise application security. The only remedy is to remove the hard-coded passwords during which time the application remains vulnerable to attack.	Application logic errors
Improper output data	Application produces data that can be used in an attack e.g. protocol errors and application data errors.	Application logic errors
Incorrect or missing file system permissions	Incorrect, inadequate or missing file system authorisation allows attackers access to a file system and its content for theft and/or data manipulation.	Authorization
Information leakage	Information leakage exploits occur when applications reveal sensitive data to by attackers.	Application logic errors

Name	Definition	Category
Insecure indexing	Flaws in search indexing processes can allow access to resources (e.g. files and data) not intended for public use. Attackers use search queries to find the resources then steal or delete files, or manipulate data.	Sensitive data protection
Insufficient entropy	When attackers can guess the result of random number generation the generating engine lacks entropy, i.e. predictable random numbers are not random.	Cryptographic exploits
Integer overflow errors	A multiplication or addition result overflows the maximum size of an integer causing incorrect data.	Data structure attacks
LDAP injection	Attackers can formulate LDAP queries and retrieve data when applications lack adequate input validation.	Injection
Least privilege faults	Applications requiring elevated authorisation privileges to execute a task should revert to least privileges after completing the task. Failure to so opens a weakness exploitable by attackers.	Authorization
Leftover debug code	Leaving debug or testing code in an application can open a back door to attackers.	Code quality
Log injection	Inserting invalid data in log files allows attackers to manipulate log entries or add fraudulent log entries.	Injection
Logic time bomb	A logic time bomb is malicious code in an application that remains dormant until triggered by an event. The code might manipulate data and/or delete files.	Malicious code or values
Man-in-the-browser attack	A man-in-the-browser attack is Trojan horse code that secretly modifies web pages, changes transaction data or creates transactions.	Spoofing
Man-in-the-middle attack	Man-in-the-middle attacks intercept communications between servers or browsers and servers to act as a proxy that manipulates or inserts data.	Spoofing

Name	Definition	Category
Memory leak	Memory leaks occur when developers fail to release allocated memory. Attackers can use memory leaks to crash an application or perpetrate denial of service attacks by allocating memory until exhausting the server's memory.	Code quality
Missing error handling	Web applications should include a default error page to avoid passing uncaught errors (e.g. 404) to an attacker.	Application logic errors
Null character injection	Attackers bypass valid data checks by including null characters %00 or 0x00 in user input data.	Injection
Parameter delimiter manipulation	Attackers manipulate parameter delimiters used by application input and cause unexpected behaviour such as bypassing authorisation and accessing data.	Injection
Parameter tampering (URL or web page)	Parameter tampering occurs when attackers manipulate URL parameters or form data.	Injection
Path or directory traversal	A directory (or path) traversal attack exploits authorisation flaws that allow attackers access to files and directories outside the web server's root directory.	Authorization
Predictable resource location	Attackers use brute force methods to guess hidden website or server content not intended for public use. Similar attacks include file enumeration and directory (or folder) enumeration.	Resource manipulation and depletion
Privacy violation	Missing or inadequate security measures guarding data, including passwords, social security numbers, and personal details, allow attackers to steal the data.	Sensitive data protection
Process control	Process control attacks occur when an application invokes commands or loads libraries from untrusted sources. The outcome is unexpected behaviour or results from the process.	Authorization

Name	Definition	Category
Reflected injection	Reflected attacks are delivered to users via email or a website. Users clicking a link or submitting a malicious form cause injected code to travel back to a vulnerable website which reflects the attack to the user's browser. The browser executes the injected code assuming it came from a trusted server. Reflected injection is a type of cross-site scripting exploit.	Injection
Regular expression denial of service	Attackers insert a regular expression that causes the expression evaluation to run for a long time, thereby using excessive server resources	Denial of service
Regular expressions	Flaws in regular expressions can provide unexpected data to attackers.	Application logic errors
Remote file inclusion (RFI)	Attackers exploit dynamic file upload features in applications to upload remote files including malicious code.	Malicious code or values
Resource injection	Attackers can alter application behaviour by changing the names or values of an application's resource identifiers e.g. file names.	Resource manipulation and depletion
Server configuration flaws	Many servers come with sample configuration files, scripts, and widely known accounts with default passwords. Attackers can exploit this data to bypass authentication and compromise authorization to server resources.	Configuration
Server-side includes (SSI)	Server-side includes is a technique for inserting content from a file into one or more other files. An attack uses server side includes manipulation or user input fields to inject scripts into HTML pages.	Injection
Session expiration errors	Attackers can reuse unused or expired session credentials.	Session management
Session fixation	Session fixation attacks hijack real user sessions by exploiting weaknesses in the way applications implement session ID management.	Session management
Session prediction	Attackers use trial and error to guess session ids and then access a server using a valid id.	Session management

Name	Definition	Category
Settings manipulation	When attackers can access application settings they can change values and influence the way the application operates.	Resource manipulation and depletion
Special element injection	Attackers inject format characters or reserved words into data input to exploit applications that fail to adequately validate input data.	Injection
Spyware	Spyware is software that collects data via a user's internet connection without the user's knowledge.	Resource manipulation and depletion
SQL injection	SQL injection occurs when an attacker inserts malicious SQL commands into an SQL statement. SQL injection attacks exploit flaws in data validation when applications use input data to build SQL queries dynamically.	Injection
Storing passwords as text	Storing or sending password as clear text allows attackers easy access to an application.	Application logic errors
String termination incorrect	Failure to properly terminate strings can lead to buffer overflow exploits, e.g. if an application expects input data as null terminated strings an attacker can input data without a null and cause a buffer overflow.	Application logic errors
Traffic flood	Traffic flood is a denial of service attack that uses many requests (e.g. UDP datagrams or DNS queries) to overwhelm a server.	Denial of service
Trojan horse	Attackers use an email link or a movie as a Trojan horse and when clicked it starts a malicious program that can erase data or capture keystrokes.	Malicious code or values
Unicode encoding	Encoding attacks exploit flaws in an application's Unicode data decoding process. An attacker can alter the decoding process and insert inappropriate data.	Code quality
Unreleased resource	When applications fail to release resources, an attacker can cause a resource leak and launch a denial of service attack by depleting a resource pool.	Resource manipulation and depletion

Name	Definition	Category
Unrestricted file upload	Uploaded files pose a threat if not managed correctly or restricted to expected mime types. Attackers can embed malicious code in uploaded files.	Code quality
Using deprecated or obsolete methods	Attackers can exploit deprecated or obsolete methods to manipulate the behaviour of an application.	Code quality
XML external entity processing (XXE)	Poorly configured XML parsers allow attackers to manipulate external references in XML documents to embed malicious data or access local files.	Application logic errors
XML validation missing	Flaws in XML document validation during parsing can allow attackers to manipulate the content of the document.	Application logic errors
XPath injection	Applications that assemble XPath queries from user input data without validating the data are open to exploit by attackers submitting invalid or malformed data to access data in XML documents.	Injection
XQuery injection	Applications that assemble XQuery queries from user input data without validating the data are open to exploit by attackers submitting commands and executing queries.	Injection