Secure Software Notes

Application Architectures

- This module uses a four-tier application architecture:
    - Client Tier - Responsible for handling the client
    - Web Tier - Responsible for handling the web server
    - Business Logic Tier - Responsible for handling the app server
    - Database Tier - Responsible for handling the database server

HTTP Requests

- Hyper-Text Transfer Protocol (HTTP) Requests are constructed as follows:
    - Request line
    - Several headers
    - Optional message body
    - Mandatory blank line
- Forms of HTTP Requests:
    - GET - Get entity from database
    - POST - Create a new populated entity in the database
    - PUT - Update a whole pre-existing entity in a database table
    - HEAD - Returns information about an API
    - DELETE - Delete an entity in a database table
    - PATCH - Update part of a pre-existing entity in a table
    - OPTIONS - Returns information about a resource
- HTTP can be encrypted (HTTPS) by using secure connections such as Transport Layer Security (TLS). Security issues when using HTTPS:
    - SSL version is outdated
    - Weak cyphers (Keys with less than 128 bits is considered too weak)
    - The Certificate Authority issuing the SSL Certificates may not be trustworthy
    - MD5 Collision - 2 strings may have the same encrypted string causing an MD5 Collision
    - Invalid Certificates:
        - Domain name mismatch
        - Expiration date passed
        - No chain of trust
    - Misconfigurations such as HTTP and HTTPS mixing or cookies that are not secure

HTTP Encoding

Encoding Schemes

- HTTP is a plain text protocol and therefore does not allow special characters to be inputted
- Special characters are thus replaced with their encoded form when sent in a URL
- There are two main types of encoding:
    - Using an ampersand (&) followed by an alias (e.g. " or ')
    - Using the HTML encoding decimal with a '&#' prefix (e.g. &#39 or &#34)
- Encoding is used to prevent Cross-Site Scripting Attacks (XSS)

Password Hash Cracking

- Hashing supports many forms of encoding such as MD5, SHA-1 and SHA-256
- There are three possible solutions to crack a hashed password:
    - Take a list of popular passwords or passwords you believe the victim would use and perform hashing calculations on each string until you find the correct combination
    - Brute force attack the system by attempting all possible character combinations and all hashing methods
    - Use a pre-computed hash table known as Rainbow Tables which can be created prior to the attack or downloaded

HTTP Authentication

- Authentication protects from Spoofing (pretending to be someone else and use their details to gain access to a system)
- Types of Authentication:
    - Basic - Transmits a base 64 encoded credentials within an HTTP request header (think API key)
    - Digest - Similar to basic but is passed through an MD5 hashing algorithm before sending
    - NTLM - Based on Microsoft Windows authentication
    - Form - (Most popular in the web today) Based on username and password sent over a secure connection

Hacker's Toolkit

- Methods of hacking a system are:

    - Trapping and modifying traffic (requests) over HTTP / HTTPS
    - Exposing application components
    - Fuzzing
    - Regular Expression modifications
    - Spiders
    - Resubmitting requests

Assessment Proxies (Man-in-the-Middle Software)

- Spider - 'Crawls' around the website and indexes all pages on a site.
    - Used for displaying a hierarchical structure of the website; allowing the hacker to see the overall layout of the system
    - There are two types of spiders:
        - Automatic spidering - Spider would look at all the links in an initial page and work their way through different pages until all pages have been catalogued
        - User-directed spidering - The user directs the spider by logging all their progress through a proxy server
- Intruder - Submits large numbers of requests using a fuzzing
    - Used for exploiting SQL Injection vulnerabilities
    - Fuzzing is the process of providing invalid, unexpected or random data as input parameters, then monitoring outputs to see when the program crashes
    - Fuzzing is used to see what inputs crashed the system and allows the hacker to target an attack against the security flaw
- Proxy - Allows requests to be intercepted and manipulated on the fly
    - Used for changing data inputted by a client to the app server by the hacker
- Repeater - Repeats requests sent to the app server
    - Used as a log for confirming and exploiting vulnerabilities

Application Scanners

- Used to discover vulnerabilities in a system such as SQLi or XSS with little to no intervention from the user
- A scanner typically spiders a website and attempts to manipulate query strings and parameters to discover vulnerabilities

Application Mapping

HTTP Fingerprinting

- Can be used to identify the web server used to host the application which is then used to identify vulnerabilities
- Ways of identifying server-side technology:

    - File extensions of server-side scripts
    - Cookies; all frameworks have different cookie types which are distinguishable

Web Application Security

- Types of vulnerabilities:
    - Broken authentication
    - Broken access control
    - SQL injection
    - Cross-site scripting
    - Information leakage
- A web app should assume that all user inputs could be malicious and defend against all possibilities
- Key problem factors contributing to web app compromisation:
    - Under-developed security awareness
    - Third-party packages may contain security flaws
    - Frameworks may contain security flaws
    - Custom development may not contain security as time has been spent on customisation and additional work as a framework is not supporting
    - It is possible for novice programmers to create a powerful application
    - Resource, time and cost constraints
    - Focus on functionality

Core Security Concepts

- Vulnerability - a weakness or flaw that may provide an opportunity to a threat agent
- Threat agent - an entity that may act on a vulnerability
- Threat - any potential danger
- Risk - the likelihood of a threat agent exploiting a discovered vulnerability
- Exposure - a system configuration issue or a mistake in software that allows access to information or capabilities that can be used by a hacker as a stepping-stone into a system or network
- Countermeasure - an administrative operation or logical mitigation against potential risks
- The CIA triangle:
    - Confidentiality - is the concept of preventing the disclosure of information to unauthorised parties
    - Integrity - is the concept of protecting the data from unauthorised alteration
    - Availability - is the concept related to the access of the software or the data or information it handles

- Authentication - the process of determining the identity of a user, by providing one or many of three methods:
    - Something you know (password, pin)
    - Something you have (token, phone)
    - Something you are (biometrics)
- Authorisation - the process of applying access control rules to a user process determining whether or not a particular user process can access an object
- Accounting / Auditing - it is a means of measuring activity; can be done by logging crucial elements of activity as they occur to create audit logs
- NonRepudiation - the concept of preventing a subject from denying a previous action with an object in a system

Cross-Site Scripting

Varieties of XSS

- Reflected XSS - the process of taking user-supplied inputs and inserting them into the HTML of the server's response; i.e. changing returned variables in the URL to malicious scripts (for example: adding a cookie to the webpage so the hacker can hijack the user's session)
- Stored XSS - the process of adding malicious code to forms so when they are displayed to other users in the browser, the code executes
- DOM-Based XSS - the process of inserting an attack payload in the response page and manipulating the DOM to reflect the hacker's objective

XSS Attacks In Action

- Payloads - the part of transmitted data that is the actual intended message; examples of payloads include:
    - Capturing a victim's session token
    - Inducing user actions
    - Virtual defacement
    - Injecting trojans
    - Exploiting any trust relationships (e.g. form autocomplete, trusted sites and plugins)
    - Escalating the client-side attack (e.g. logging keystrokes, capturing browser history, port scanning local network)
- Delivery Mechanisms (Reflected) - the process of presenting an attack to the victim(s) by guiding them to a targeted URL; examples include:
    - Phishing by bulk emailing URL
    - Spear phishing (targeting a specific user)
    - Sending URL via third-party apps (Whatsapp, Facebook, etc...)
    - Hosting a malicious website
    - Pay for banner advert linking to the URL

- Delivery Mechanisms (Stored) - the process of presenting an attack to the victim(s) by gathering data from inputs of a user; examples include:
    - Forms
    - Uploading of files
    - Instant messaging
    - Recording to app logs

Finding and Exploiting XSS Vulnerabilities

- Ways of exploiting XSS vulnerabilities include:
    - Bypassing anti-XSS filters (a filter may be set up to find script tags, however, you can capitalise characters or insert spaces which may bypass a filter)
    - Finding and exploiting reflected, stored or DOM-based XSS

SQL Injection

- Injection flaws - allows attackers to relay malicious code through the web app to another system; attacks include:
    - Calls to the OS via system calls
    - Use of external programs via shell commands
    - Calls to the backend database via SQL (i.e. SQLi)
- Interpreter - code such as SQL that reads user-supplied information and code combined to execute an instruction
- Attacking data stores - the process of inputting malicious code to:
    - Break processes
    - Retrieve information
    - Delete information

Exploiting a Basic Vulnerability

- Adding special characters such as an inverted comma will break an SQL statement at run-time
- After an inverted comma, a hacker is able to add any statement to inject into the query

Injecting into Different Statement Types

- SELECT - reads data
- INSERT - adds data
- UPDATE - amends data
- DELETE - removes data

Finding SQLi Bugs

- Most data is stored as strings and numbers
- Injecting into strings - strings are wrapped with inverted commas and by adding another into the string, the hacker is able to inject whatever code they wish into the system
- Injecting into numbers - numbers can be submitted as equations such as 3+1 and that may not be wanted in the system

The UNION Operator

- Is used in SQL to combine the results of two or more SELECT statements into a single result state
- Using the method above (Injecting into strings), a hacker can use a UNION operator to modify the query as they wish
- For the statement to work, the two parts of the statement must have compatible data types
- NULL can be used as it can be converted into any data type

Bypassing Filters

- Filters are methods that may remove or sanitise certain characters or may block common SQL keywords
- To bypass filters you can use the following information:
    - Inverted commas are not required to inject into numeric data fields
    - Changing strings into ASCII codes or Unicode
    - Using capitals/lower case letters or a combination
    - Using SQL Comments

Security During the Development Cycle

- If security is not handled before publication/release, the following may arise:
    - 50 to 200 times more expensive to fix
    - Bad PR - fewer customers joining the site as they know it has been breached
- If security is not taken care of during the SDLC, there could be a potential snowball effect where design flaws slip into implementation, then implementation flaws slip into testing, etc...
- These are some activities you can implement during the SDLC to mitigate security flaws:
    - Code Review (Manual) - a form of code analysis that coders undertake to find bugs that hackers may exploit (costly, thorough)

    - Code Review (Automated) - a form of code analysis that bots undertake to find bugs that hackers may exploit (fast, cost-effective)
    - Threat Modelling - a structured activity for identifying and managing threats

Threat Modelling

- A procedure to identify threats and vulnerabilities in the earlies stage of the development cycle
- Benefits include:
    - More secure application
    - A structured method for finding and mitigating security flaws ensuring a fast and cheap security checking process
    - Provides a summary of potential security flaws and recoveries upon breach
- Elements of threat modelling:
    - Assets - valuable data and equipment that should be secured
    - Threats - potential attacks a hacker can instigate
    - Vulnerabilities - flaws in the system that can allow an attacker to realize a threat
- Steps of threat modelling:
    - Identify assets - assets that are important for the system such as databases or servers
    - Describe architecture - this can include frameworks, folder structures, versions, APIs in use, etc...
    - Decompose application - break down application regarding its process, including all sub-processes running in the application
    - Identify threats - list and identify threats in a descriptive manner to review further
    - Document threats - classify threats with parallel instances so that threats can be identified in the application in a structured and repeatable manner
    - Rate threats - rate severity of the threats
- A popular threat modelling tool is the STRIDE model:
    - Spoofing - consists of using another user's credentials without their knowledge
    - Tampering - consists of a hacker gaining access to the system and modifying its behaviour
    - Repudiation - consists of hiding actions they have performed such as removal of logs or spoofing credentials as another user as to not receive blame or identification
    - Information disclosure - consists of gaining access to a restricted system and accessing data which in turn they may choose to leak
    - Denial of service - consists of preventing valid users access by launching DOS attacks and occupying the server's resources

    - Escalation of privilege - attempting to acquire additional privileges by spoofing or tampering
- When to use threat modelling:
    - Every time there is a change in the system architecture
    - After a security breach
    - After new vulnerabilities have been introduced
    - As soon as the architecture is ready

Secure Design Principles

- Least privilege - a principle in which a person or process is given only the minimum level of access right (privilege) that is necessary for that person or process to complete an assigned operation
- Separation of duties - a principle which states that the successful completion of a single task is dependant upon two or more conditions that need to be met and just one of the conditions will be insufficient in completing the task by itself
- Defence in depth - a principle where single points of complete compromise are eliminated or mitigated by the incorporation of multiple layers of security safeguards and risk-mitigation countermeasures
- Fail-safe - a principle that aims to maintain confidentiality, integrity and availability (CIA Triangle) by defaulting to a secure state (e.g. whitelisting)
- Economy of mechanism - a principle that ensures a simple design of a system; the greater the complexity, the greater the potential vulnerabilities
- Complete mediation - a principle that ensures that authority is not circumvented in subsequent requests of an object by a subject (i.e. even if you have checked the privileges of a user before, check them again)
- Open design - a principle where the implementation details of the design should be independent of the design itself, which can remain open (i.e have design diagrams so a developer can view the design of the system without the need for accessing the actual system)
- Least common mechanism - a principle that disallows the sharing of mechanisms that are common to more than one user or process if the users and processes are at different levels of privilege
- Psychological acceptability - a principle that aims at maximising the usage and adoption of the security functionality in the software by ensuring that the security functionality is easy to use (e.g. backup emails - confirm once use forever), you should focus on the following when maximising psychological availability:
    - Ensure people understand why security measures are in place
    - Be transparent about security
    - If people don’t see the value, they will not use it
- Weakest link - a principle that states that the resiliency of your software against hacker attempts will depend heavily on the protection of its weakest components

- Leveraging existing components - a principle that focuses on ensuring that the attack surface is not increased and no new vulnerabilities are introduced by promoting the reuse of existing software components

Secure Development Principles

Input Validation

- Performed to minimise malformed data from entering the system however should not be used as the primary method for preventing XSS or SQLi.
- Below are some definitions for data validation:
    - Integrity checks - ensure that data has not been tampered with and is the same as before
    - Business rules - ensure that data is not only validated, but business rules are correct
    - Validation - ensure that data is strongly typed, has correct syntax, within length boundaries, contains only permitted characters or that numbers are correctly signed and within range boundaries
- Types of input validation:
    - White list validation - involves defining what is authorised and by definition, everything else is not authorised
    - Regular expressions - involves defining a set of shorthand rules which are compared to a string to see if any rule is met
    - Client-side and server-side - use client-side for faster responses, however, server-side validation should always be used as client-side verifications can always be bypassed

Canonicalisation

- Also known as standardisation or normalisation
- Is a process for converting data that has more than one possible representation into a standard, normal or canonical form

Output Encoding

- Is a process for translating special characters into some equivalent that is no longer significant in the target interpreter (e.g. encode ‘shutdown --’ so the database is not turned off)
- Sanitization - the process of converting something that is considered dangerous into its innocuous form
- Input sanitisation occurs after the user has supplied data and before it is processed, and can be accomplished using any one of the following methods:
    - Stripping - removing harmful characters from user-supplied input
    - Substitution - replacing user-supplied input with safer alternatives

    - Literalization - using properties that render the user-supplied input to be treated as the literal form

Error Handling

- If an error has not been handled correctly by the developer, the server would return the raw error and will provide much information to a hacker attempting to gain access
- Validate all input to prevent this from occurring
- Use non-verbose error messages to ensure that enough information is provided to the end-user but is not providing specific details to a malicious hacker

Authentication

- The process of verifying that an individual, entity, process or website is what or who it claims to be

Authorisation

- Ensures that an authenticated user has the appropriate privileges to access resources
- The resources a user has access to should depend on their role

Auditing and Logging

- Logging usually means recording various events that happen within an application and the focus lies on the interests of the developer as it records specific events or parameters

Session Management

- Session hijacking occurs when an attacker impersonates the identity of a valid user and interjects themselves into the middle of an existing session, routing information from the user to the system and from the system to the user through them

Secure Communication

- An architectural decision must be made to determine the appropriate method to protect data when it is being transmitted
- The primary benefit of transport layer security is the protection of web application data from unauthorised disclosure and modification when transmitted between clients and web application servers, and between the web application server and the back-end and the back-end and non-browser based enterprise components

Secure Resource Access

- Obscurity is not the same as security and therefore don't try to hide sensitive resources, but ensure they are heavily guarded

Secure Storage

- When securing storage you need to identify what data must be protected and how you are going to provide the protection
- A few areas where mistakes are commonly made include:
    - Failure to encrypt critical data
    - Insecure storage of keys, certificates and passwords
    - Improper storage of secrets in memory
    - Poor choice of algorithm
    - Attempting to invent a new encryption algorithm
    - Failure to include support for encryption key changes and other required
- The easiest way to protect against cryptographic flaws is to minimise the use of encryption and only keep information that is absolutely necessary
- Never hard code keys into code

Cryptography

- When it comes to cryptographically protecting information, the predominant flaw in software is the lack of encryption of sensitive data

Cryptography in Depth

Basic Cryptosystems

- The idea behind a cryptosystem (cypher system) is to conceal information from all parties that are not entitled to access it, this is done by replacing the information by what appears to be a sequence of random symbols
- An alternative approach to secret communication is to hide the message itself; this is called steganography
- A sophisticated approach to concealment is to both encipher a message and then conceal it
- This module introduces two principal tools in cryptography:
    - Substitution
    - Transposition

Elements of Cryptography

- The encryption and decryption processes are composed of two parts:
    - The algorithm
    - The key
- Any person/machine listening to ciphertext transmissions is called an eavesdropper or interceptor
- If the eavesdropper is attempting to obtain the plaintext, they are said to be attacking the system
- If they succeed in obtaining the key, it is said that they have broken/cracked the system
- The study of cryptology is broken down into:
    - Cryptography - the design of cryptographic systems
    - Cryptoanalysis - the investigation of attacks on systems
- The algorithm incorporates a key as part of the encipherment process
- Kerckhoff's Principle - The same algorithm can be used with different keys
- You should change keys regularly or else all future and past plaintexts will be vulnerable
- Symmetric cryptography - the key is considered to be the 'same' for both encryption and decryption; the understanding is that the keys are so closely related that knowledge of one implies knowledge of the other (e.g. an encryption key could have all characters shift positively by 3 Unicode characters, and the decryption would have to shift negatively by 3 Unicode characters)
- Asymmetric cryptography - uses two keys, which while related, are such that knowledge of the encipherment key does not give knowledge of the decipherment key

Substitution

- Mono-alphabetic substitution cypher - involves the substitution of each letter in the alphabet with another letter in the alphabet; the decipherment involves replacing the substituted letter with its original value

Digital Certificate

- Digital certificate/public key certificate - An electronic passport that allows a person, computer or organisation to exchange information securely over the internet using a public key infrastructure
- A certificate provides identifying information, is forgery resistant and can be verified because it was issued by an official trusted agency
- The certificate contains:
    - The name of the certificate holder
    - A serial number

    - Expiration date
    - A copy of a public key used for encrypting messages and signatures
    - The digital signature of the certificate-issuing authority (CA) so that a recipient can verify that the certificate is real
- Public key - a large numerical value that is used to encrypt data
- Private key - a large numerical value that is mathematically linked to the public key and is used to decrypt data
- In asymmetric cryptography, whatever is encrypted with a public key may only be decrypted by its corresponding private key and vice versa
- Public Key Infrastructure - supports the distribution and identification of public encryption keys, enabling users and computers to both securely exchange data over networks such as the internet and verify the identity of the other party
- PKI consists of hardware, software, policies, standards to manage the creation, administration, distribution and revocation of keys and digital certificates

Network Protocols

- Secure Socket Shell (SSH) - a protocol that provides administrators with a secure way to access a remote computer; it also provides strong authentication and secure encrypted data communications between two computers connecting over an insecure network
- Secure Sockets Layer (SSL) - a protocol for securing connections between application clients and servers (was discontinued in 2015 and replaced with TLS)
- Transport Layer Security (TLS) - a protocol that provides privacy and data integrity between two communicating applications by working at the OSI Presentation layer by
    - Providing end-to-end security by eavesdropping, tampering and message forgery
    - Certificates
    - Handshake

Cryptographic Keys - Key Servers

- Key Distribution Centre (KDC) - a server that distributes keys/passwords to users on a secured connection, these are distributed as the system is set up
- Key Encryption Keys (KEKs) - the keys that are distributed by a KDC
- Steps for a KDC Communication:
    - When device A attempts to communicate with device B, device A requests a key from the KDC by sending their KEK and device B's details
    - The KDC sends back 2 session keys, one encrypted using device A's KEK, the other using device B's KEK

    - Device A then encrypts the message using the session key and sends the encrypted message and the encrypted session key for device B
    - Device B decrypts the session key and then uses the session key to decrypt the message

Cryptographic Keys - Asymmetric Key Cryptography

- Asymmetric Key Cryptography - defined above as public-key cryptography, assumes that all devices on a network know a device's public key
- Steps for Asymmetric Key Cryptography:
    - Device A creates a session key
    - Device A obtains device B's public key
    - Device A uses the public key to encrypt a session key and sends it to device B
    - Device B uses the private key to decrypt the message and retrieves the session key
    - Both parties have a copy of the session key and can now communicate securely

Cryptographic Keys - Key Agreement

- A key transport protocol securely distributes keys to users whereas with a key agreement protocol, parties negotiate the key to be used
- For example, the Diffie-Hellman Key Exchange protocol (DHKE) has the following process:
    - Both devices agree on a number (X)
    - Device A has a secret (Y), device B has a secret (Z)
    - Device A mixes secret (Y) and number (X) = XY
    - Device B mixes secret (Z) and number (X) = XZ
    - Each device sends their mix and re-mix with their secret so both end up with the same mix = XYZ
    - Note = mixes cannot be undone

Cryptographic Keys - Key Management

- Security of any system ends up with the keys
- Key management involves the storage, deliverance and protection of keys throughout the lifetime of a key
- Symmetric Keys
    - Generation - must be generated at random to ensure it is difficult to reproduce them
    - Distribution - is handled with a key hierarchy as discussed in key servers, at the top of the hierarchy there is the master key used to distribute KEKs and then the KEKs are used to distribute session

keys (the further down the hierarchy, the more often a key is changed and least safe a key is)
    - Storage - typically stored on a physically stored system (e.g. internal business infrastructure or a smart-card)
    - Usage - keys should only have one purpose and should carry a label indicating its use which should be securely bound to the key
    - Change - should be changed regularly to reduce the likelihood of a successful attack
    - Destruction - all data that used a key must be overwritten with up-to-date versions with the new key
- Asymmetric Keys
    - Generation - involves the use of sophisticated mathematical algorithms and a generator has to rely on externally generated keys or purchased software
    - Distribution - not an issue
    - Storage - the public key is not needed to be stored in secret as it should be available to the public but it must be tamper-proof, however, the private key needs to be stored securely
    - Usage - should only have one purpose
    - Change/Destruction - when a private key is compromised, the public key is useless and all data encrypted with that key is liable for hacking

Security Testing

Static and Dynamic Analysis

- Static Analysis
    - Performed in a non-runtime environment
    - Typically a tool will inspect the program code for all possible run-time behaviours to identify flaws, back doors and malicious code
    - Better at detecting bugs earlier in the process of development than dynamic analysis
- Dynamic Analysis
    - Performed in a runtime environment
    - Will monitor system memory, functional behaviour, response time, performance metrics, etc...
    - Better at exposing subtle flaws that are difficult to detect as a developer
- Both forms of analysis can be completed by a tester or automated

Basics of Security Testing

- In an SDLC, developers should test continuously and reduces the following bugs:

    - Memory
    - Input
    - Performance
    - Insecure behaviour
    - Undefined behaviour
- The process of security testing can be divided into 4 categories:
    - Functional security tests that can verify that security controls of your software work as expected
    - Non-functional tests against known weaknesses and faulty component configurations
    - Holistic security scan - when an application or infrastructure is tested as a whole
    - Manual testing and code reviews - sophisticated work that still cannot be quite delegated to machines
- To increase security testing productivity, security software should be used to automate testing such as spidering, DDOS, proxy and fuzzing.
- Detecting memory problems is a large deal to ensure no information is leaked from memory
- Use fuzzing for random input validation such as Unicode or unknown characters
- Testing should be done at more than one level, you can have a bottom-up or top-down approach:
    - Bottom-up - test each method and entity and work up to larger systems until the whole system is tested
    - Top-down - test larger systems and work down until methods and entities are tested
- Performance testing should be implemented to ensure DDOS attacks cannot disrupt service

Penetration Testing (Pentesting)

- Reconnaissance Stage - the more information gathered, the higher the likelihood of success, therefore a significant amount of time should be spent obtaining as much information as possible about the target
- Scanning Stage - Comprehensive coverage is needed in the scanning stage, AI can be used to focus on scanning important areas
- Gaining and Maintaining Access - involves taking control of one or more network devices in order to extract data or to launch attacks on other targets, once a system is scanned, pentesters need to ensure that the system does not have any loopholes
- Covering Tracks and Reporting - this involves removing traces of the attack on the system (e.g. user logs, access channels, error messages)