Solver: Ng Wei Xuan Eddie

1)

a)

i) SPF policies are published in the **Domain Name System (DNS)** as SPF or TXT records.

ii) **Domain portion** of "MAIL FROM" command is used by the SMTP server to look up the relevant policy.

iii) SPF policy contains information about **which hosts are permitted to use the domain owner name in the "MAIL FROM" or "HELO" identity**. In other words, it specifies authorized senders.

iv) Receiver takes **IP address of SMTP client** and **SPF record of sending domain** to verify whether the client is authorized to send mail for that domain. **MTA Check Routine** decides whether to accept or reject the mail. In **MTA Check Routine**, mail receiver must evaluate the check host() function using the following arguments: (1) **IP address of SMTP client emitting the client** (2) **Domain portion of "MAIL FROM"** (3) **"MAIL FROM" or "HELO" identity**. One of the 4 qualifiers (+, -, ~, ?) is then returned. '+' indicates **Pass** which is a legitimate user that is authorized to use the identity. '-' indicates **Fail** which means Client is not authorized to use the identity and recipient should use SMTP reply code 550. '~' indicates **Soft Fail** which means it is more critical than Neutral thus further checks advisable. '?' indicates **Neutral** which means no policy is found thus no assessment can be done.

b) Soft Fail (~) indicates more critical than Neutral (?) thus further checks are advisable. Hard Fail (-) indicates SMTP client emitting the email is not authorized to use the identity hence the recipient should use SMTP reply code 550. Soft Fail does not result in the rejection of email however, Hard Fail results in the rejection of the email.

c)

i) NTU can publish the **cloud service provider's public verification key** as **DNS resource record** in NTU's domain (ntu.edu.sg).

ii) Cloud service provider can **sign the mails it sends using its own private signature key** and **giving NTU's domain as the Signature & Signing Domain IDentifier (SDID)** found in the DomainKeys Identified Mail (DKIM) header fields.

Solvers Opinion/Note: This question is asking about how DominKeys Identified Mail (DKIM) can be used

d) In an SPF alignment check, the **domain in the visible FROM: field** is compared against the **domain in the MAIL FROM: field used for accessing the SPF records** to ensure both fields match.

2)

a)

i) User **authorizes the authorization request made by Client Application** to access a resource owned by the user that is stored at a resource server.

ii) Client Application **asks user for authorization to access a resource** stored on a resource server.

b)

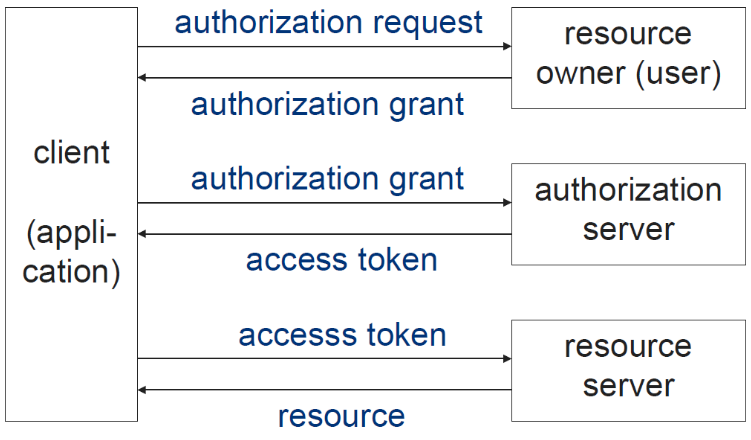
i) Application Client sets the redirect_URIs.

ii) Redirect_URIs define where tokens that give access to user data stored on resource server are sent to.

iii) Redirect_URIs need to be carefully handled as it **contains sensitive information** such as **access token** from the authorization server after the user successfully authorize the client application to access resources owned by the user.

c)

Step 1: **Application (Client) asks user for authorization** to access a resource
Step 2: If the **user authorizes the request**, the **application receives an authorization grant**
Step 3: **Application requests access token from authorization server**; authenticates itself and presents the authorization grant
Step 4: If **application is authenticated** and **authorization grant is valid**, the **authorization server issues access token to the application via** UA to **redirect URI** nominated in step 3. The **redirect URI** is **checked against app's policy** to ensure its validity
Step 5: **Application requests the resource from the resource server** presenting the **access token**
Step 6: If **access token is valid**, the resource server **delivers the resource to the application**



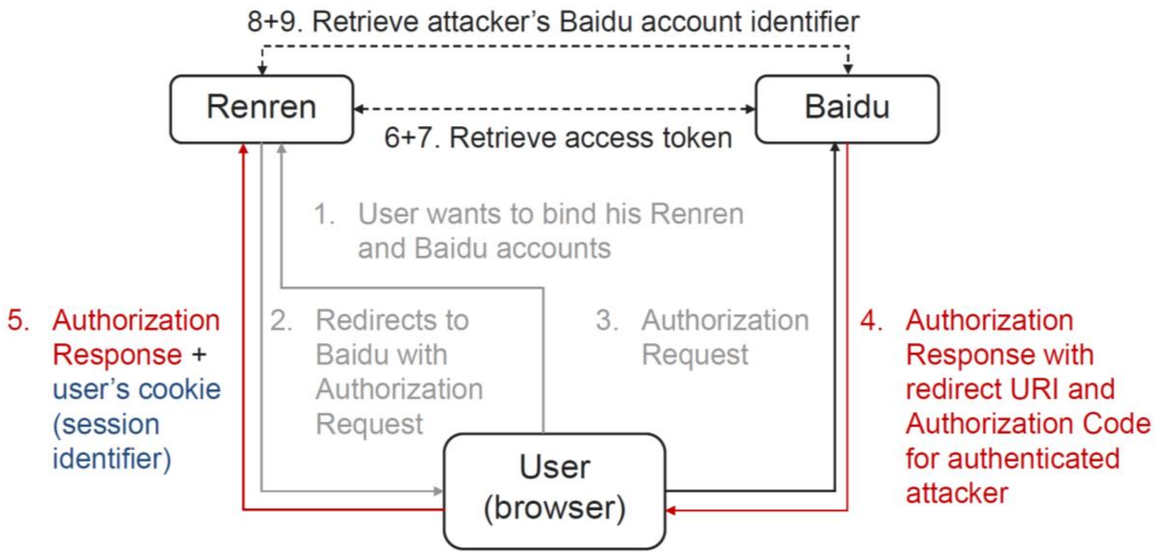
d)

i) State variable **links authorization request** made by the Application (Client) and the **authorization response** so that the **receiver can check** that a **response matches** the

corresponding actual request made.

ii) **Binding Attacks** becomes possible (E.g. Renren-Baidu Binding Attack). Reason being, request can no longer be bounded to response thus a user's request can be matched to an attacker's crafted response.

Solvers Opinion/Note: FYI: Renren-Baidu Binding Attack


8+9. Retrieve attacker's Baidu account identifier
Renren
Baidu
6+7. Retrieve access token
1. User wants to bind his Renren and Baidu accounts
5. Authorization Response + user's cookie (session identifier)
2. Redirects to Baidu with Authorization Request
3. Authorization Request
4. Authorization Response with redirect URI and Authorization Code for authenticated attacker
User (browser)

- Attacker has an account with Baidu and runs OAuth to get an Auth Response for his own ID
- Attacker posts this response as a link to a forum
- User with active Renren session clicks on that link
  - State not included in Renren Auth Request → Request cannot be bound to Response hence the User's Request and Attacker' Response will matched
- User Agent (browser) will follow redirect URI and redirect response to Renren
- Renren retrieves attacker's Baidu ID and binds it to user's Renren ID (steps 6-9 of the binding protocol)

e)

i) **Authorisation server** issues id tokens

ii) Id tokens **contains user attributes signed by the authorization server**. User attributes includes the following fields shown in the table.

Fields	Descriptions
iss	– Identifier for the server that issued the token
sub	– Issuer's unique identifier for the "subject" (User)
aud	– Identifier for the client requesting the token (Relying Party)
nonce	– Set by client when making an authorization request
iat	– Issued at time when token was issued
exp	– Expiry time of token

auth_time	– Time User was authenticated
acr	– Strength of authentication

iii) Id tokens are **used by Client Application (Relying Party) to establish a session with the User Agent (Client),** based on claims in the id tokens. Id tokens cannot be used to query other services. Calls from Application to third party services should still use the access token.

3)

a) XSS
Parties involved: Attacker, Client (Victim), Server ("Trusted by Client). Malicious scripts injected into trusted websites and the victim's browser has no knowledge the script should not be trusted. Browser thus executes the malicious script thinking it's from a trusted server. The malicious script can access any cookie, session tokens or sensitive information retained by the browser. XSS exploits the trust a client has on the server; client only authenticates the 'last hop' of the entire page but not the true origin of all its parts.

CSRF
Parties involved: Attacker, "Trusted" User, Server. CSRF attacks are executed by stealing the identity of an existing user and sending requests to a website using that identity. CSRF attacks target functionality that causes a state change on the server such as changing the victim's password or making purchases with the victim's credentials. CSRF exploits the "trust" (E.g. cookies, authenticated SSL session) a target website has in a user to execute actions at the website with the user's privileges.

b) (1) Filter server outputs and browser inputs by differentiating between HTML elements and user data. (2) Block execution of scripts in browser altogether. (3) Authorize scripts explicitly.

c) Sanitize User Inputs
(1) Removing single quotes from inputs
(2) Escape/Encoding by replacing illegal characters by a safe encoding or escape with a backslash
(3) Using of PHP check functions to perform escaping of certain characters
Change Mode of Execution
(1) Usage of Stored Procedure which are parametrized commands called by transmitting name of procedure and parameter values
(2) Usage of Prepared Statements which are bound parameters. Script first compiled with placeholders instead of user input; placeholders are then replaced with actual user input when executing the compiled script

d) Stagefright bug is an **integer overflow vulnerability** in Android core component (StagefirghtLibrary) implemented in C++ which is used as a backend engine for playing multimedia formats such as MP4 files. The bug affected Android 2.2 devices and newer. **Exploitation of the bug** allows an **attacker to perform arbitrary operations** on the victim's device **through remote code execution and privilege escalation**. **Video sent via MMS** was a **possible avenue of attack** through the libStageFright mechanism. **Many text messaging apps**

(E.g. Google's Hangout) **automatically process video** so it is ready for viewing as soon as the user open the message thus **attack can happen without the user knowing it** and **attacker can take over the device.**

e) **"checkm8"** is a major iOS bug which affects every Apple device with an A5 through A11 chipset (iPhone 4S to iPhone X). This exploit provides the foundation required to customise jailbreak for every vulnerable model of device. This bug exploited the flaw in Apple's "bootrom" memory in the processor that contains the fundamental code that run first when a device power on. Since bootrom is foundational to a system, it can create powerful jailbreaks that do not depend on vulnerabilities specific to a particular iOS version. This bug cannot be fixed since it is running from bootrom level and bootrom cannot be updated.

4)

a) These programs could **possibly be crafted by hackers or individuals with malicious intents**. The programs **could install malicious malware** or **even backdoor in the IT systems**. Running these programs before reviewing or testing in a sandboxed condition could **result in destroying of IT system's data or operations completely.**

b) (1) Information Gathering
(2) Vulnerability Identification
(3) Exploitation
(4) Gaining Access and Privilege Escalation
(5) Covering Tracks
(6) Reporting

c) (1) System Level: Each app runs under its own unique user id (**Application Sandboxing**)
(2) Component Level: **Mandatory Access Control (MAC)** system by a reference monitor based on access permission labels of components

d)

Android vs iOS	
Source-Code	
Open-Source (Android)	– Led to rapid development and improvement of Android by community contributions, also means that when new vulnerabilities are uncovered, they are fixed very rapidly
Closed-Source (iOS)	– Should make it more secure (security by obscurity) and, in many ways, it does – Means that security vulnerabilities are fixed in a hierarchical manner, often taking longer to push a fix to market than Android
App Vetting Process: Filtering Malware	
Google	– Does not thoroughly test before they go live onto the Google Play Store – E.g. Simple Android photo app named Meitu requires authorization to access location, phone status and identity, and a host of sensitive cellular functionality that has absolutely nothing to do with photo editing

Apple	– Tightly controls which apps are available on its App Store, vetting all apps to avoid allowing malware through – Apple App store, user reviews and app ratings are barely available
Updates: Latest Versions are more secure	
iOS	– Big events that usually prompt mass upgrades
Android	– Users are often slow in updating – Vulnerabilities need to be patched across a wide range of manufacturers and versions
Vulnerabilities	
– Both found to contain many serious security vulnerabilities iOS has had a total of 1277 vulnerabilities whereas Android has had a total of 1381 vulnerabilities –mostly code execution, overflows, DoS, and Information leakage problems for both	
iOS	– Struggled with loopholes that allowed apps to execute standard library code directly, bypassing security restrictions (currently they are patched but does not mean that no similar problems in future)
Android	– Serious problems with Stagefright vulnerability
Conclusion	
– Android seems to be less secure because of updates and app vetting problems and also customizations by device manufacturers; – iOS more secure due to its tight controls on hardware and software and its ability to impose tighter security – But it would only take one piece of perfectly formed iOS malware to do as much damage as thousands of copycat Android threats $\rightarrow$ A vulnerability in Web Browser results in first "Jail Breaking" techniques whereas in Android, only affects certain applications or systems is affected due to sandboxing	

e) (1) High degree of outsourcing of IT infrastructure
(2) Consumers not in full control of security
(3) Sharing (multi-tenancy) of resources by unrelated consumers
(4) Data are distributed over wider area or accessed by greater number of devices
(5) Different security and privacy regulations in different countries

5)

a) <u>Permissioned Blockchain</u>
Accessible by members only where there is a trusted Certificate Authority (CA) that issues and verifies TLS certificates, enrolment certificates and transaction certificates. Trust established between parties with known identities thus all nodes can be assumed to be honest.

<u>Permisionless Blockchain</u>
Trustless e-cash environment with no trusted parties and anyone can join and access the blockchain by downloading the software and creating a digital wallet. There is no need to prove identity, no membership service and need not put a deposit in advance to join. Each party identity is not known hence Sybil Attack can happened since identity can be forged or faked.

b)

i) Proof-of-Work is **computed by solving a cryptographic puzzle specific to a block of transactions**. A cryptographic puzzle is a hash function with hand input *t* and *s* is a solution to the puzzle when *h(t,s)* has a required number of leading zeros.

ii) The effort for computing a Proof-of-Work **serves as statistical evidence** that a computationally expensive task has been performed.

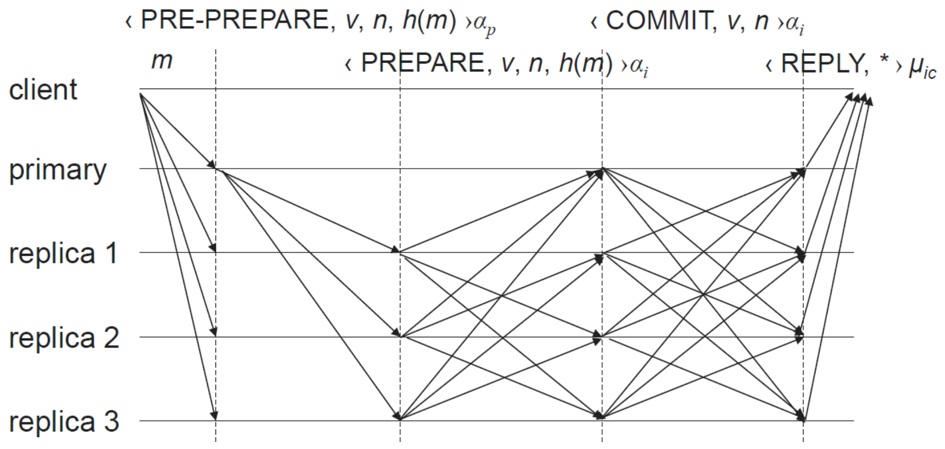
iii) Sybil attack involves creating a large number of new identities to rig the current log if agreement uses a majority algorithm. Yes, Proof-of-Work can defend against Sybil attacks as identity can be faked but Proof-of-Work cannot; Proof-of-Work must be computed by solving cryptographic puzzles that are computationally expensive. Sybil attacks thus fails if honest nodes have more than 51% of CPU power since their branch will grow faster than competing branches intending to rig the logs.

c)

i) Consensus protocol is required as in a blockchain, there is no trusted adjudicator to decide which version of the transaction history to accept as main chain. Reason being, two nodes can work on a different next block and broadcast their blocks concurrently and nodes may receive these blocks in different order. As a result, there can be concurrent 'truths' about the current state of the blockchain.

ii) Ability of a distributed computer network to correctly reach a sufficient consensus despite malicious nodes in the system failing or sending out incorrect information. This prevents against catastrophic system failures by reducing the influence of malicious nodes.

iii) Parties involved are the Clients that send authenticated messages to all replicas which provide services. Clients and replicas share secret authentication keys; replicas share secret authentication keys. In other words, two secret keys for any pairs of replicas for both directions. Primary is the replica in charge of ordering messages and assigns sequence numbers to them; backup replicas follow the message ordering given by the primary.


‹ PRE-PREPARE, v, n, $h(m)$ ›α_p
‹ COMMIT, v, n ›α_i
m
‹ PREPARE, v, n, $h(m)$ ›α_i
‹ REPLY, * › μ_{ic}
client
primary
replica 1
replica 2
replica 3

(1) Client c sends $m = < REQUEST, o, t, c > a_c$ to all replicas where o is the operation, t a timestamp, a_c an authenticator.
(2) Primary p assigns sequence number n to message m, sends $< PRE - PREPARE, v, n, h(m) > a_p$ to all backups where v is the current view, $h(m)$ hash of m, a_p an authenticator.
(3) Every backup i agreeing with the sequence number n sends $< PREPARE, v, n, h(m) > a_i$ to all other replicas.
(4) Once replica i has $2m$ matching $PREPAREs$ for message m, it sends $< COMMIT, v, n, i > a_i$ to all other replicas.
(5) Once replica i has $2m + 1$ matching $COMMITs$ for message m, it sends $< Reply, v, t, c, i, r, u > \mu_{ic}$ to the client where r is the result of the operation, μ_{ic} an authenticator between replica i and client.

iv) PBFT is suitable for permissioned blockchains as identity of all parties are known and a trust relationship is established between these parties. Hence these parties can be deemed to be honest and has no incentive to deliberate send incorrect information. However, in permissionless blockchains as no trust has been established and with anonymous identity, a consensus protocol such as PBFT can result in Sybil attack to occur.

--End of Answers--