COMP 3010 Assignment 3

Submitted by: Colin Dyson, 7683407

Due: Wednesday, March 20, 2019

1. HTTPS or Hyper Text Transfer Protocol Secure is an extension of the original HTTP which uses encryption to protect against eavesdropping and tampering in web communications. It was originally developed to secure highly sensitive online transactions, such as those within a bank server, but has since become more widely used than HTTP on most servers. It helps to protect user identity and general internet privacy. The only major difference between HTTPS and regular HTTP is that before sending a message, HTTPS requires the message to be encrypted. Once the encrypted message has reached it’s destination, it is then decrypted for reading. The encryption and decryption steps are carried out by TLS, or Transport Layer Security. TLS is a cryptographic protocol which aims to ensure secure network communication. Using TLS enables features such as message authentication, to ensure a message has not been tampered with, identity authentication, which is often used to ensure that the server a client is connected to is not a disguised alternative, and the aforementioned encryption of messages sent between servers and clients. In order to support HTTPS, a server administrator must create a public key certificate, which are distributed by trusted certificate authorities.
2. In a stream based client/server interaction, the first message sent by the client will likely be some identifying credentials, such as a username and password. Naturally, these should be encrypted before being sent to the server. The server will then pass those credentials on to the authentication server. The message should still be encrypted at this point, since the line of communication between the authenticator and the web server may not be secure. The authentication server will receive the message, decrypt, check the validity of the credentials, then return the result to the web server. If they are valid, the server will then continue to accept messages from the client and performing whatever processing is requested. In other words, over TCP, a server will validate client identity by sending the credentials to the authentication server. If they are valid, the connection continues until ended by the client. If they are invalid, the server will inform the client of this, and likely close the connection. For a connectionless system, the server and authentication server will communicate every time a client sends a message to the server. This means the client’s credentials must be provided to the server within the rest of the message for every message sent. Before the server operates on the request, it must perform the same validation as before.
3. In order to determine if a particular server behind a redirect server is up, the redirect can simply attempt to connect to the server. If the connection times out, it is safe to assume the server is not currently online. To collect a list of all available servers, the redirect server can iterate through all known addresses, attempting a connection with each one. Every address which receives a connection can then be added to a list of currently online servers. Such a list should be refreshed at some regular interval to keep an up to date roster of available redirect locations. To determine the least heavily loaded server, the request to check whether a server is online can include a request for that particular server’s current workload. The simplest way to determine this would be to use the Runtime class. Upon receiving the message, each server could return to the redirect server the results of Runtime.getRuntime().totalMemory() and Runtime.getRuntime().freeMemory(). These values can inform the redirect server how much free memory each server has compared to it’s total system memory. Presumably, if the server is the only thing running on each machine, the server with the most free memory will be the best candidate for the next incoming redirect. A reverse proxy server is a server which retrieves requested resources from it’s associated web servers on the behalf of a client. When a client connects, the server will transparently redirect any requests to the appropriate web servers, then collect the results and return them to the client as though the reverse proxy server itself provided the data. This can help improve performance by distributing the load of incoming requests to several servers. This allows the computation to be handled by separate machines while the proxy is free to accept new connections from clients. Using a reverse proxy server is typically more secure, since the characteristics of the origin servers remain hidden to potential clients. Additionally, these proxies can be supported with security firewalls to ensure only ‘safe’ messages are sent to origin servers.
4. See a3/q4
5. See a3/q5
6. Same origin policy enforces browsers to disallow scripts from attempting to access data from other pages on a website unless the pages share the same origin. The origin is a unique combination of the protocol, domain and port. In other words, a script written on one page of a website may not access data located on another page if they do not share an origin. This prevents client side scripts from accessing secured data by traversing the DOM of the current page. This can be too restrictive for certain page designs. Cross-origin sharing policy allows pages to request resources from another domain. The browser will first send a request with the Origin header to the alternative domain. The server will then respond with a list of origin sites that have been set to allow cross-origin resource sharing with. If the page that the browser is currently displaying is on the list, the browser will then perform the cross-origin request to retrieve the defined resources.