Weaknesses in Session Token Handling:

There are various ways in which an application's unsafe handling of tokens can make it vulnerable to attack.

Disclosure of Tokens on the Network

This vulnerability arises when the session token is transmitted across the network in unencrypted form, enabling a suitably positioned eavesdropper to obtain the token and masquerade as the legitimate user. Suitable positions for eavesdropping include the user's local network, within the user's IT department, within the user's ISP, on the Internet backbone, within the application's ISP, and the IT department of the organization hosting the Application. In each case, this includes authorized personnel of the relevant organization and any external attackers who have compromised the infrastructure.

In the simplest case, where an application uses an unencrypted HTTP connection for communications, an attacker can capture all data transmitted between client and server, including login credentials, personal information, payment details, etc. In this situation, an attack against the user's session is often unnecessary because the attacker can view privileged information and log in using captured credentials to perform other malicious actions. However, there may still be instances where the user's session is the primary target. For example, suppose the captured credentials are insufficient to perform a second login (e.g., in a banking application, they may include a number displayed on a changing physical token, or specific digits from the user's PIN). In that case, the attacker may need to hijack the eavesdropped session to perform arbitrary actions. Alternatively, suppose there is close auditing of logins, and notification to the user of each successful login. In that case, an attacker may wish to avoid performing his login to be as stealthy as possible.

In other cases, an application may use HTTPS to protect key client-server communications yet may still be vulnerable to the interception of session tokens on the network. There are various ways in which this weakness may occur, many of which can arise specifically when HTTP cookies are used as the transmission mechanism for session tokens:

■ Some applications elect to use HTTPS to protect the user's credentials during login but then revert to HTTP for the remainder of the user's session. For example, many webmail applications behave in this way. In this situation, an eavesdropper cannot intercept the user's credentials but may still capture the session token, as shown in Figure -1.

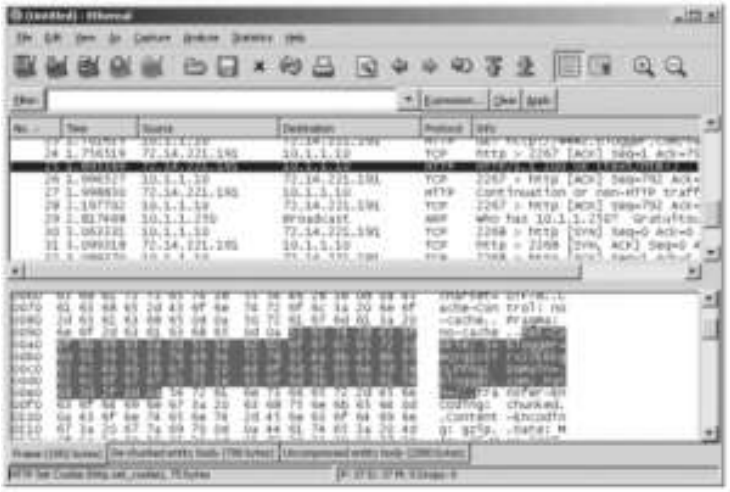

Figure -1: Capturing a session token transmitted over HTTP

■ Some applications use HTTP for pre-authenticated site areas, such as the site's front page, but switch to HTTPS from the login page onwards. However, in many cases, the user is issued a session token on the first page visited, and this token is not modified when the user logs in. The user's session, which was initially unauthenticated, is upgraded to an authenticated session after login. In this situation, an eavesdropper can intercept a user's token before login, wait for the user's communications to switch to HTTPS, indicating that the user is logging in, and then attempt to access a protected page (such as My Account) using that token.

■ Even if the application issues a new token following successful login and uses HTTPS from the login page onwards, the token for the user's authenticated session may still be disclosed if the user revisits a pre-authentication page, either by following links within the authenticated area, by using the Back button, or by typing the URL directly.

■ In a variation on the previous case, the application may attempt to switch to HTTPS when the user clicks the Login link; however, it may still accept login over HTTP if the user modifies the URL accordingly. In this situation, a suitably positioned attacker can modify the pages returned in the pre-authenticated areas of the site so that the Login link points to an HTTP page. Even if the application issues a fresh session token after successful login, the attacker may still intercept this token if he has successfully downgraded the user's connection to HTTP.

■ Some applications use HTTP for all static content, such as images, scripts, style sheets, and page templates. A warning alert within the user's browser often indicates this behavior. An attacker can intercept the user's session token when the user's browser accesses a resource over HTTP, and use this token to access protected, non-static areas of the site over HTTPS.

■ Even if an application uses HTTPS for every single page, including unauthenticated site areas and static content, there may still be circumstances in which users' tokens are transmitted over HTTP. For example, if an attacker can somehow induce a user to request HTTP (either to the HTTP service on the same server if one is running or to http://server:443/ otherwise), their token may be submitted. The attacker may attempt this by sending the user a URL in an email or instant message, placing auto-loading links into a website the attacker controls, or using clickable banner ads.