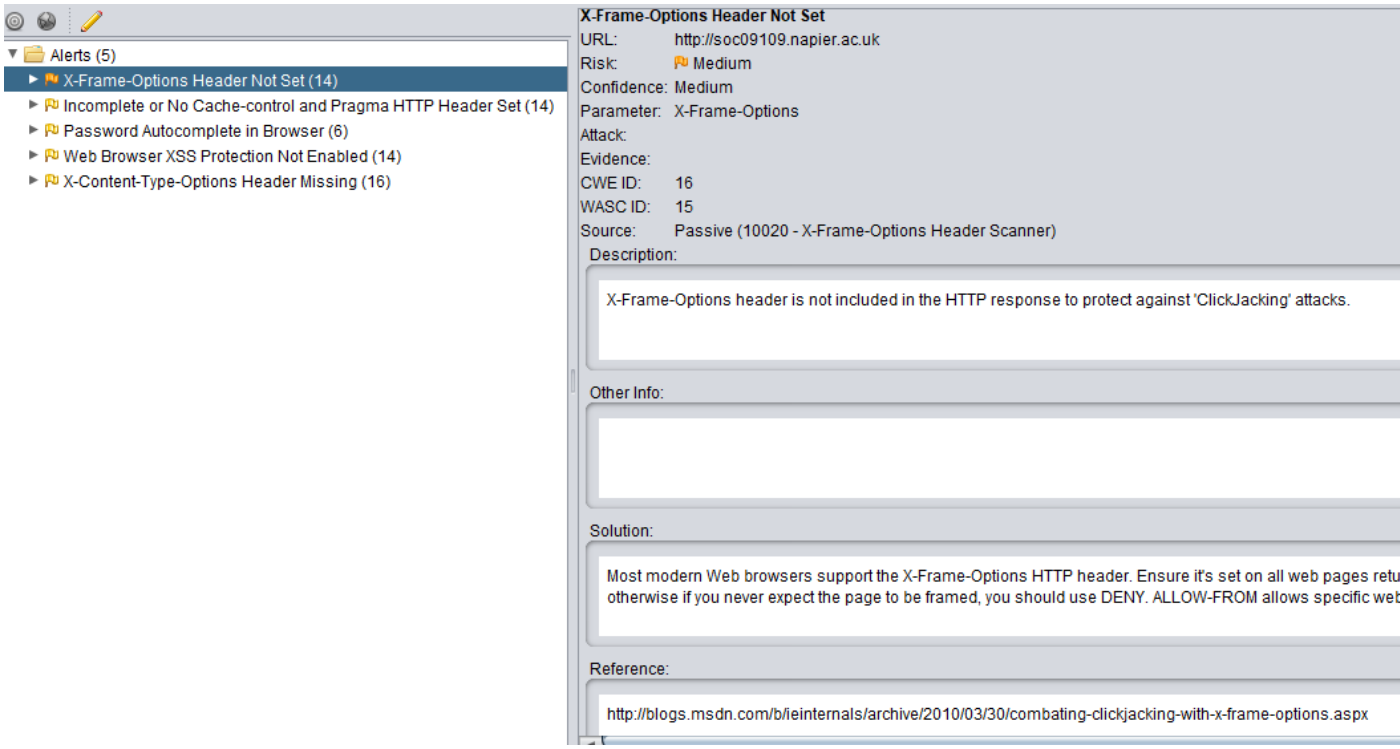
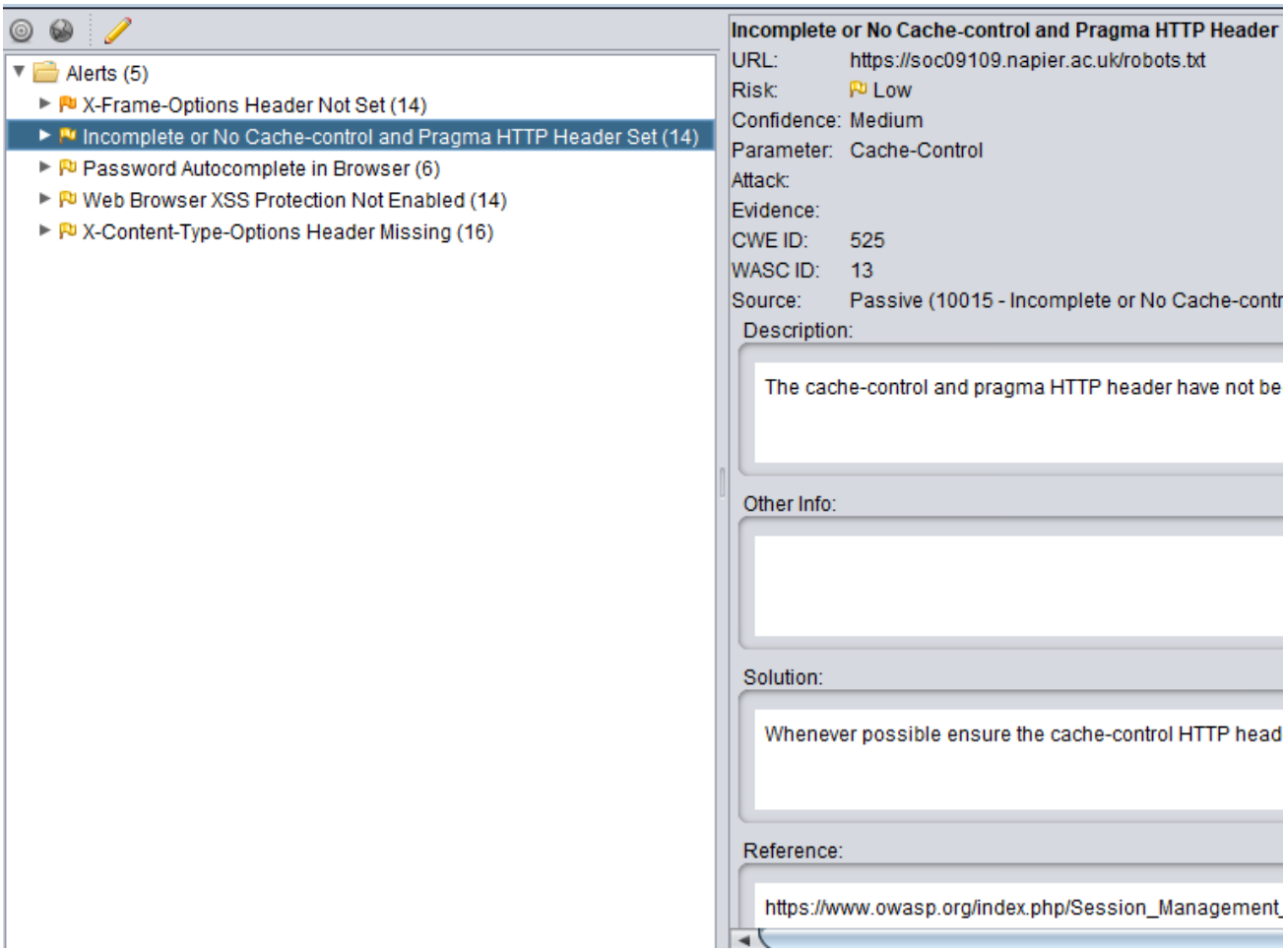
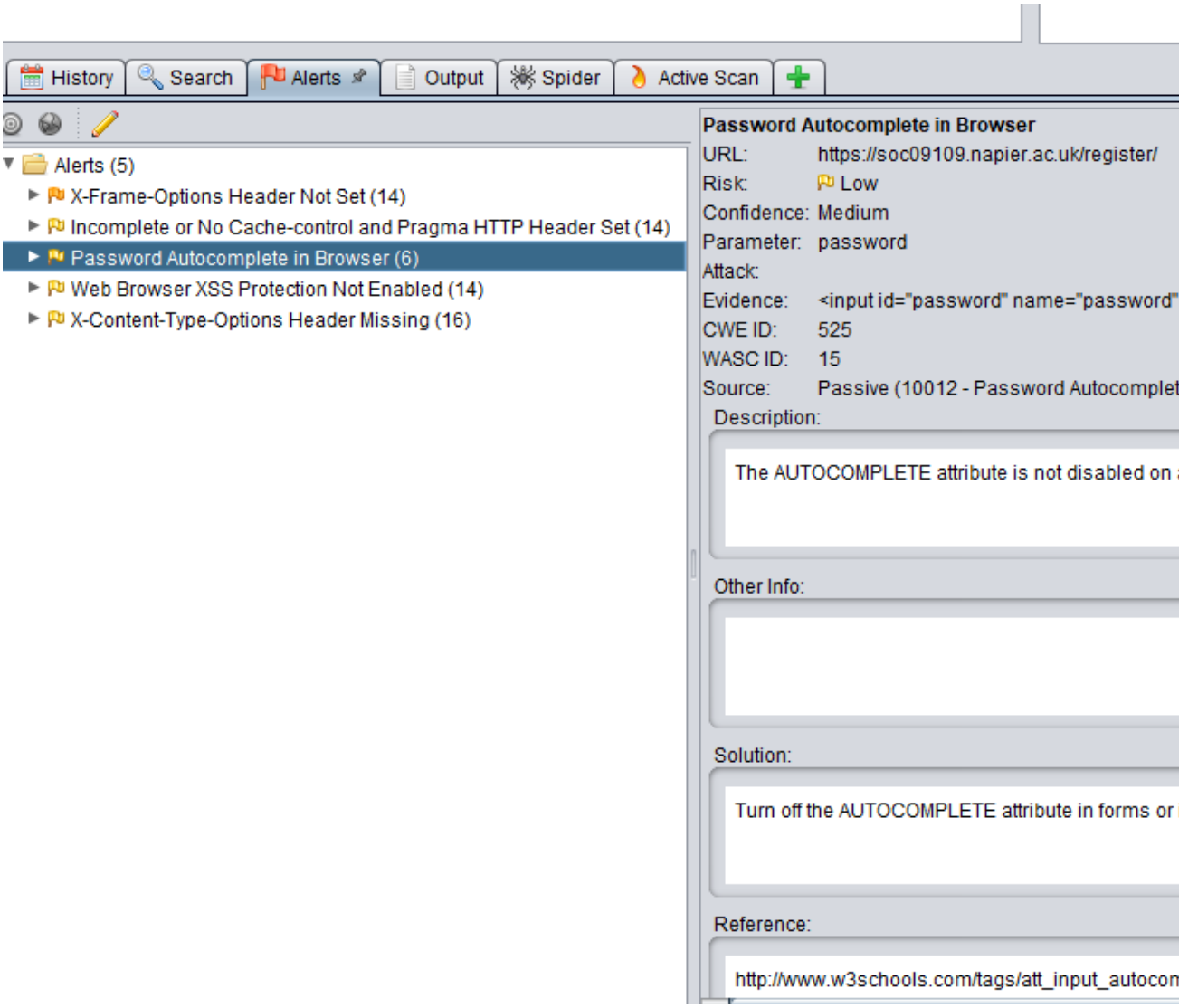



1) X-Frame-Options header is not included in the HTTP response to protect against 'Clickjacking' attacks. Clickjacking, also known as a "UI redress attack", is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both. The most popular way to defend against Clickjacking is to include some sort of "frame-breaking" functionality which prevents other web pages from framing the site you wish to defend.




2) Even after the session has been closed, it might be possible to access the private or sensitive data exchanged within the session through the web browser cache. Therefore, web applications must use restrictive cache directives for all the web traffic exchanged through HTTP and HTTPS, such as the “Cache-Control: no-cache,no-store” and “Pragma: no-cache” HTTP headers [5], and/or equivalent META tags on all or (at least) sensitive web pages.

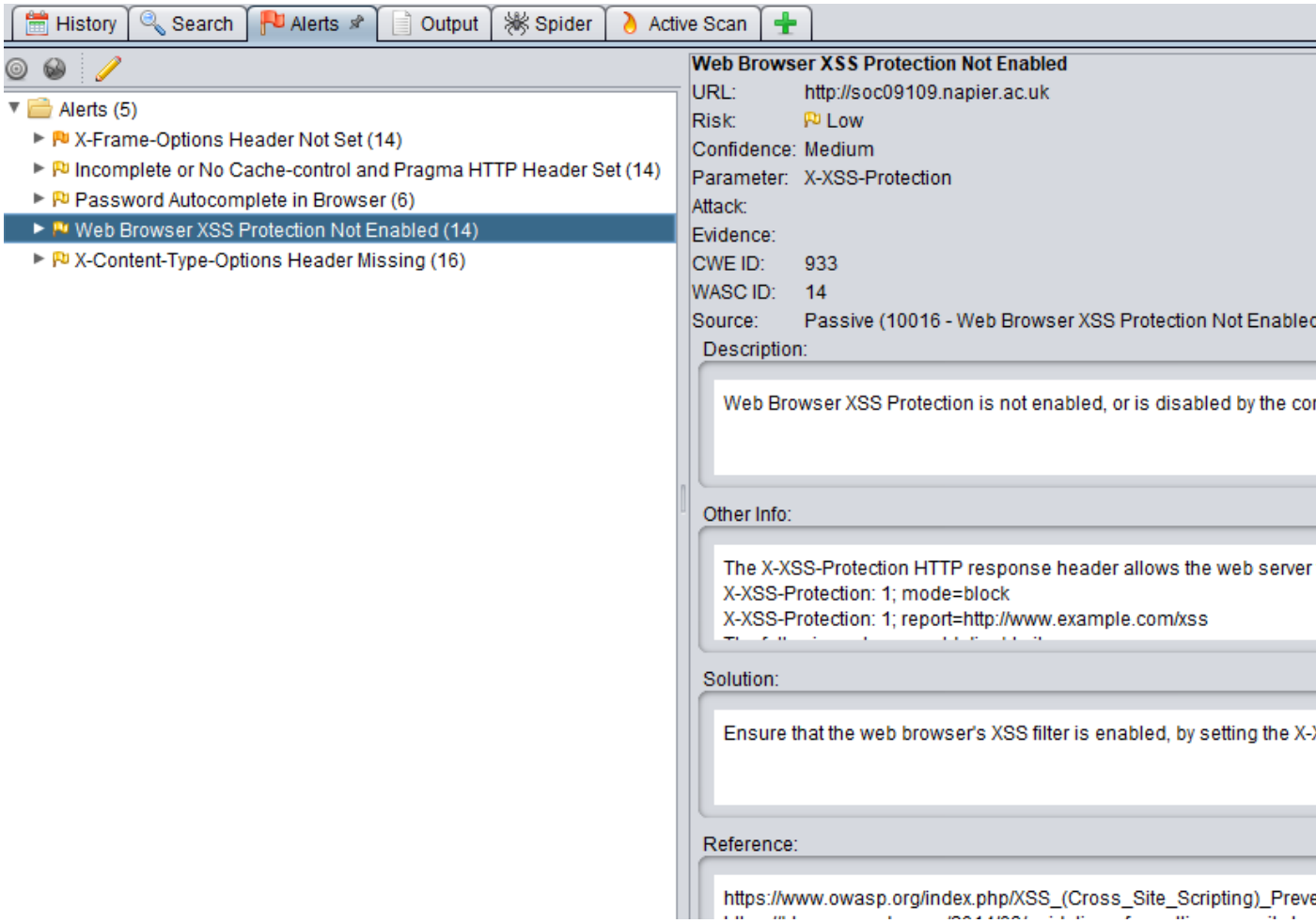
Independently of the cache policy defined by the web application, if caching web application contents is allowed, the session IDs must never be cached, so it is highly recommended to use the “Cache-Control: no-cache="Set-Cookie, Set-Cookie2"” directive, to allow web clients to cache everything except the session ID.



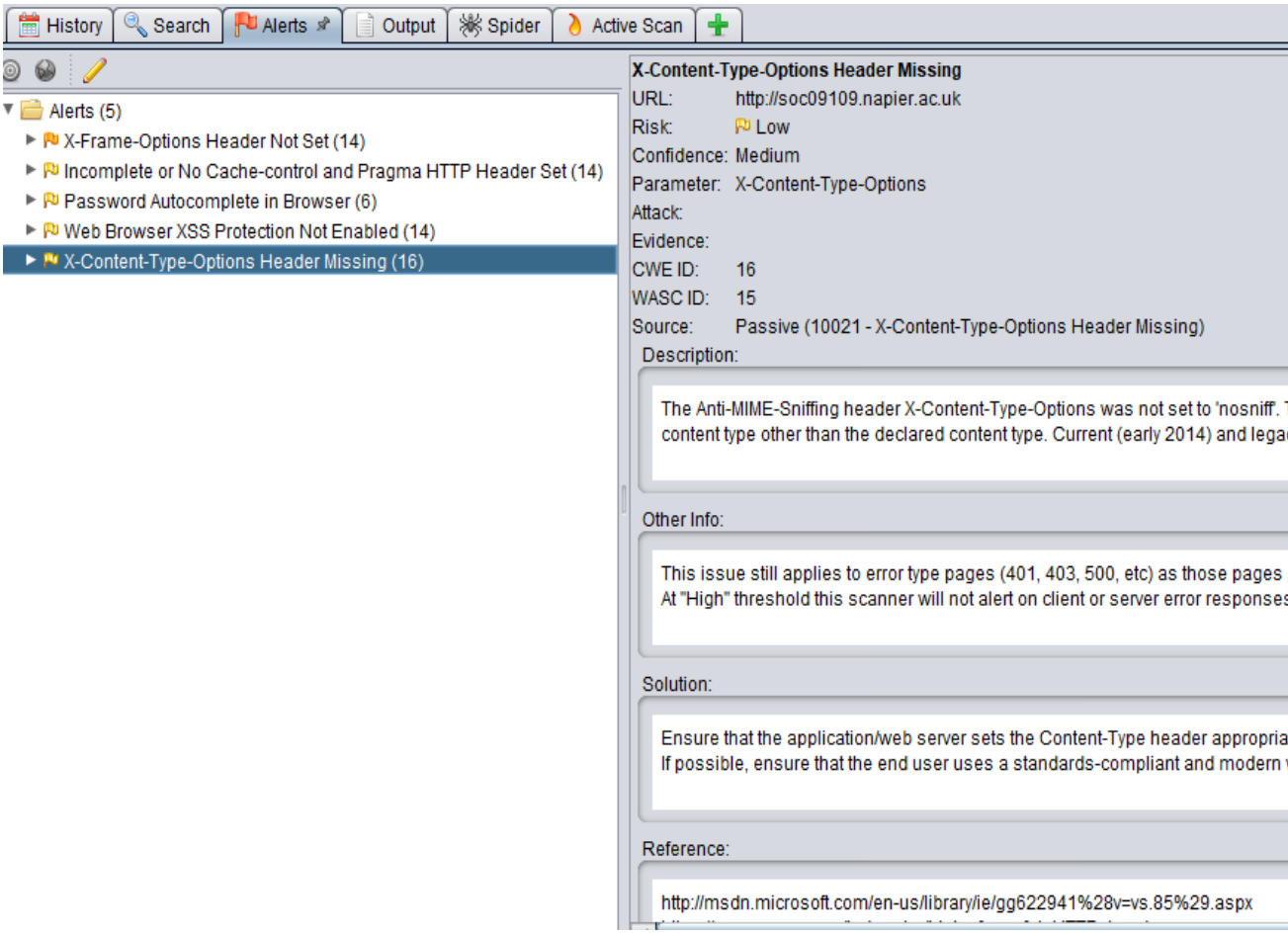

3) Most browsers have a facility to remember user credentials that are entered into HTML forms. This function can be configured by the user and also by applications that employ user credentials. If the function is enabled, then credentials entered by the user are stored on their local computer and retrieved by the browser on future visits to the same application.

The stored credentials can be captured by an attacker who gains control over the user's computer. Further, an attacker who finds a separate application vulnerability such as cross-site scripting may be able to exploit this to retrieve a user's browser-stored credentials.

To prevent browsers from storing credentials entered into HTML forms, include the attribute **autocomplete="off"** within the FORM tag (to protect all form fields) or within the relevant INPUT tags (to protect specific individual fields).




4) Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server. Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. The web browsers XSS filter will need to be enabled to resolve this problem.




5) The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing. MIME sniffing is done for the purpose of determining an asset's file format. This technique is useful in the event that there is not enough metadata information present for a particular asset, thus leaving the possibility that the browser interprets the asset incorrectly. Need to make sure, to avoid potential MIME attacks, that x-content-type is set to 'nosniff'