Testing for Session Hijacking

ID
WSTG-SESS-09

Summary

An attacker who gets access to a honest user's cookies can impersonate her by presenting such cookies: this attack is known as session hijacking.

Even when the web application is entirely deployed over HTTPS, the `Secure` attribute [1] should be used for the session cookies. For example, assume that `www.good.com` is entirely deployed over HTTPS, but does not mark its session cookies as `Secure`:

1. The user sends a request to `http://www.another-site.com`
2. The attacker corrupts the corresponding response so that it triggers a request to `http://www.good.com`
3. The browser now tries to access `http://www.good.com`
4. Though the request fails, the session cookies are leaked in clear.

How to test

Black-Box Testing

We assume to have a testing account on the website for the victim, that we call Alice and a testing account on the website for the attacker, that we call Bob. The intuition behind the testing strategy for session hijacking is to simulate a scenario where the attacker steals all Alice's cookies she might be exposed to.

We assume that the attacker is a network attacker (i.e. an attacker who has access to the same network as the victim), so we could have a cookie leakage in case of either no HSTS adoption and the `Secure` attribute is not set, or partial HSTS adoption[2], the `Secure` attribute is not set and the Domain attribute is set to a parent domain. The attacker may then use these cookies to access Alice's account: if they are enough to act on Alice's behalf, session hijacking is possible.

Even when this is not possible, however, security might still be at risk, because it might be that not all the cookies were disclosed to the attacker and the attempted operation failed because just a subset of the expected cookies was sent to the website.

To account for this case, we also perform a fresh login to the website as the attacker to get a full set of cookies and then restore the cookies stolen from Alice before reattempting the operation, so that all the website cookies (though mixed from two different accounts) are sent as part of a new operation attempt: if the operation succeeds in Alice's account, session hijacking is possible.

Specifically, the testing strategy proceeds as follows:

1. Login to `www.target.com` as Alice and reach the page under test;
2. Find the cookies which satisfy either of the following cookie leakage conditions:

    - no HSTS adoption and the `Secure` attribute is not set and there's
    - partial HSTS[2] adoption, the `Secure` attribute is not set and the Domain attribute is set to a parent domain;
3. Clear all the other cookies from the browser;
4. Perform the operation under test;
5. Check: has the operation been performed? If yes, report as insecure;
6. Clear the cookies from the browser;
7. Login to `www.target.com` as the attacker and reach the page under test;
8. Restore in the browser the cookies previously kept at step 2;
9. Perform again the operation under test;
10. Clear the cookies from the browser and login to `www.target.com` as Alice;
11. Check: has the operation been performed? If yes, report as insecure.

1: A secure cookie is only sent to the server when a request is made with the `https:` scheme.

2: We refer to full HSTS adoption when a host activates HSTS for itself and all its sub-domains, and to partial HSTS adoption when a host activates HSTS just for itself.

Tools

- OWASP ZAP
- JHijack - a numeric session hijacking tool

References

- Calzavara, S., Rabitti, A., Ragazzo, A., Bugliesi, M.: Testing for Integrity Flaws in Web Sessions.