Galore: Session Handling

1. Introduction

Reconstructing the session as is while replaying is one of the many challenges in Galore. Many reasons why this is difficult.

- The session variables that were captured by the capture service may not work if we replay them as is. one good example for this could be session ids in cookies for http.
- requests and responses can not be grouped while capturing, as the capture service keep capturing the data in the sequence of time. Pairing the request to a response is not going to be easy.
- Finding the session-begin and session-end from the capture data may not always be easy, specially in the case of state-less protocols like http/imap.
- handling the authentication ???

The document explains the context of session in capture/replay/result handler modules. The names Session Variable and cookie has been used in many places in the document. They essentially mean the same; cookie is a session variable in http context and it has special characteristics. In a way, we can say that cookie is a specialization of session variable.

2. Session Capture

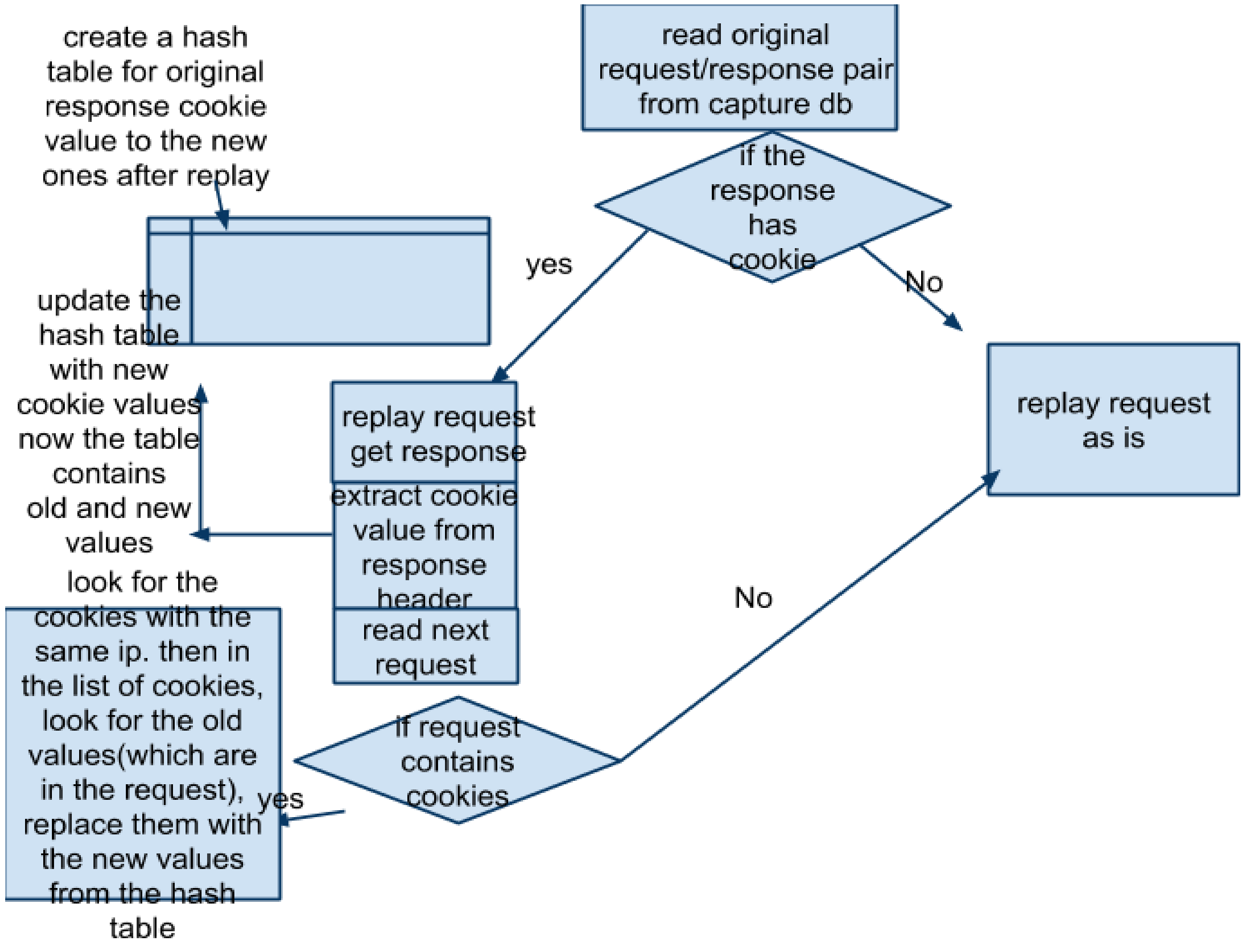
capture agent captures the raw request packets in the order of arrival and the responses in the order of sending. while capturing, we can not do lot of fancy things, however we have to make sure that the we have all the information that is needed to reconstruct the session while replay. For any protocol the following five at any given time would be unique.
(*protocol, source IP, source port, dest IP, dest port*)
ex: (HTTP, 192.168.22.2, 3456, 172.168.4.2, 6663)
This indicates that, for all of the session based protocols, if the capture agent captures all the above five, we should be able to reconstruct the session while replay. However, this would not work for non-session based protocols like http. These protocols have indirect way (in case of http, these are cookies)to maintain the session.
one thing to note in case of non-session based protocols though, is that the cookie/session variable information is embedded in the protocol headers/data itself. capture agent captures all the data/headers anyway. so, there is no need to store additional information. However, there needs to be a protocol specific handler to extract the session variables out from the payload.

3. Session Reconstruction while Replay

Session based Protocols
For session based protocols, we can identify the caplets based on the above mentioned five

parameters. If we can get all the caplets for which the above five parameters are the same, we know that we have all the request/response caplets for that session.

Session-less Protocols
For session-less protocols, this can be tricky. Here is how it can be done in case of http.


create a hash table for original response cookie value to the new ones after replay
read original request/response pair from capture db
if the response has cookie
yes
No
update the hash table with new cookie values now the table contains old and new values
replay request get response
extract cookie value from response header
read next request
replay request as is
No
look for the cookies with the same ip. then in the list of cookies, look for the old values(which are in the request), replace them with the new values from the hash table
if request contains cookies
yes

Each protocol has its own way of handling session variables(http-cookies). As part of Galore installation, a default session variable handler is provided, which provides the following functionality.

- interface to add a session variable
- can be stored in-memory or on-disk
- interface to provide expiry datetime for session variable/cookies

- interface to delete session variable
- interface to search for a session variable value based on its name, IP, timestamp etc.,

Each protocol handler can override the above functionality to implement its specific functionality.

4. Http Specialization

Here are the sequence of steps that can be realized while replaying the requests with cookies. you can find the pictorial representation in the above figure.

1. read the request/response pair from the capture db
2. read the response header to see if there are any cookies returned while capturing.
3. if there are cookies returned as part of the response, create the hash table(for the first time) and store
    a. localIP
    b. website (for which cookie is to be stored)
    c. cookies
        i. name
        ii. value_c(capture time value)
        iii. creation_timestamp_c(capture time timestamp)
        iv. path_c(capture time path)
        v. expiry date_c(capture time expiry)
        vi. value_r(replay time)
        vii. creation_timestamp_r(replay time)
        viii. path_r(replay time)
        ix. expiry_date_r(replay time)
4. replay the request and get the response.
5. read the response headers and extract the cookie values, update the hash table with _r values(corresponding to the replay time cookie values)
6. read next request, response pairs from the capture db.
7. if the response contains more cookies in the response headers, go to step 3 and jump to step 8.
8. extract cookie values if the request contains cookies
9. lookup the hash table with IP, website to get the related cookies.
10. In the cookies list, look up again for the old values(meaning cookie values while capturing _c variables), if you find an entry, replace the old value with the new value.
11. stuff the request with new values.
12. replay the request.
13. This should magically work :-) :-)