Cut and Paste Attack

Oauth WG @ IETF95
Nat Sakimura
Nomura Research Institute
Cut and paste attack in essence is:

• Attacker
  • gets “code” or “token” of the victim somehow; and
  • pastes it to his own session.

• It has been known for a long time
  • E.g. Bradley, J. “The problem with OAuth for Authentication”
Why are we talking about it recently again?

• Some new novel ways of stealing “code” are pointed out:
  • Stealing it from the server log;
  • Stealing it by inserting the attacker AS in the authz request;
  • Open redirectors; etc.

• Once the “code” has been stolen, the attacker can paste it in a new authorization response that he requested.
attacker gets the code by inserting itself

Bad IdP gets Good IdP token

Access client
authz redirect to Bla
authz req to Bla
authz redirect to Gla
authz req to Gla
authn+authz
codeV+state
codeV+Bcred
codeV stolen
some page
Attacker gets the code from server log
There are many other ways of stealing the “code”. (= cut)
It will be used by the attacker by …
Pasting it to his own session

**State check succeeds**

s/code/codeV/

**ATTACKER**

**UA**

**CLIENT**

**AS**

**RS**

[Diagram showing a cut and paste attack with attacker, UA, client, AS, and RS nodes, and the states and messages exchanged.]
It succeeds because there is no way to find out that the Authz response has been tampered

- The browser redirect segment (both Authz res and Authz req) are not integrity protected.
- OpenID Connect expected this attack and has c_hash (code hash) in the ID Token that comes back from the authorization endpoint: ID Token acts as the dethatched signature for the “code”.
- Similarly, the authz req tampering was dealt with the request object.
How do we want to deal with it?

1. JWT Dethatched signature
2. Bind the code to the state and pass the state with the code to the token endpoint. (Hacky..)
3. Hmac the state+client_id+authz_ep+token_ep+code using the “client secret” as the key, and send the hash in the response. The client then sends the hash to the token endpoint for the AS to check against the hash that has been stored at the AS. (Reverse PKCE).
4. Do not care.