Session Continuation

Phillip Hallam-Baker
Nico Williams
Existing Work

• Problem Statement and Requirements
  – draft-williams-websec-session-continue-prob

• One or more proposed solutions
  – draft-hallambaker-httpintegrity
  – draft-williams-websec-session-continue proto
  – draft-hammer-oauth-v2-mac-token
Three Types of Authentication:

• Registration
  – Decide she is ‘Alice’ give her a token / password
  – [Out of Scope]

• Presentation [HTML / SAML / OAUTH / ... ]
  – Alice proves she has a token
  – Give her another token
  – [Out of WEBSEC scope – See HTTP-AUTH]

• Continuation [Cookies]
  – Re-authenticate without representing credentials
TLS is not the (full) answer

• TLS Client Authentication is rarely viable
  – Works very well when it works
  – Requires client certs

• Only some traffic moves over TLS

• TLS is not designed to meet threat model
  – Protect bearer tokens from chosen plaintexts generated by Turing complete engine controlled by the attacker. (Aww come on!)
Problems

• HTTP Cookies are bearer tokens
  – Present cookie to gain access
  – Brittle security
  – Cached by intermediaries under Rule 81
  – Remain in shared machines
  – Relies on TLS in unsafe ways (CRIME, BEAST)

• No session closure
  – Cookies typically cached for 2 weeks!
Alternative

• Registration, Presentation as before
• Standard for session continuation
  – MAC Based (like Digest, maybe Digest 2.0)
    • Use big (128+ bit keys) for security
    • Client never passes key en-clair
  – Standard mechanism for replay attack prevention
  – Standard session log out
  – TLS channel binding (if using TLS)
Presentation Implications

• SAML, OpenID, OAUTH, ...
  – Simplifies design
  – Purpose designed capability for function

• HTTP-Auth
  – Take out of design consideration

• Cookie replacement
  – Need mechanism to pass key en-clair to client
Cookie Implications:

• 2 types of cookie
  – Server session state stored on client
    • Use encryption and authentication
  – Bearer token authentication
    • Should GO AWAY
    • Won’t (quickly)
Use Cases

• Web Browsing
  – Has to support legacy
    • Must accept a downgrade attack
  – User interface concerns

• Web Services
  – Can mandate particular mechanism
  – May not have a ‘user’
Requirements

• Permit determination that specified party
  – Sent a message
    • Cookies
  – Sent this message
    • Content binding
  – Sent this message to me
    • Replay attack
    • Man in the Middle Attack
    • TLS Channel binding
Content Binding

• Scope
  – None
    • Just like cookies do today
  – Request / Response line (Method, URI)
    • Often the most important

– Headers
  • Here be dragons

– Message Body
  • Ignore transport encoding (e.g. chunked)
Replay Attack

• Bound to issue time
  – Only prevents replay outside time window
  – Does not require local state
  – Requires trustworthy clock

• Challenge-Response nonce
  – Proves message was sent to me
  – Requires local state to reject duplicates.
TLS Binding

• HTTP and TLS frequently have different extent
  – TLS accelerator gateway
  – MITM Proxy
• TLS Binding allows HTTP endpoints to tell
  – Specify credentials
Realization

• Use Authorization / WWW-Authenticate
  – Headers exist
  – Wrong names

• Use New Header
  – Avoids confusion with legacy
  – Requires new headers

• Bike shed discussion
Next Steps

• Do we want to address this?
• What features do we not need?
  – How do we decide?