Dos and Don’ts of Client Authentication on the Web

Kevin Fu, Emil Sit, Kendra Smith, Nick Feamster

Presented: Jesus F. Morales

Overview

- Web client authentication
  - Limitations
  - Requirements
  - Security models
- Interrogative Adversary
- Hints for designing a secure client authentication scheme
- Analysis of the scheme

Introduction

- Client authentication: a common requirement
- Many schemes are very weak
  - Home-made
  - Careless implementation
  - Misunderstanding of how different tools work
  - Balance between usability and security
- Lack of a client authentication infrastructure
- Lack of control over user interfaces

Client Authentication and Limitations

- Client authentication: The problem
  - Client side
  - Server side
- For this paper:
  - Client authentication: “proving the identity of a client (or user) to a server on the Web”.
- Sources of confusion
  - Authentication vs. confidentiality

Practical Limitations

- Deployability
  - Technology must be widely deployed
  - HTTP is stateless and sessionles
    - Client must provide authentication token
  - Useful but high overhead
    - Javascript, Flash, Schockwave…
- User Acceptability
- Performance
  - SSL: computational cost of initial handshaking

Types of Breaks

- Breaks
  - Existential Forgery
    - Forge authenticator for at least one user
    - Example: subscription services
  - Selective Forgery
    - Forge authenticator for a particular user
    - Must construct a new authenticator
  - Total Break
    - Most serious
    - Recovery of a key used to mint authenticators

Types of Adversaries

- Interrogative Adversary
  - Can make queries of a Web server
  - Adaptively choose next query
  - Adaptive chosen message attack
- Eavesdropping Adversary
  - Can sniff the network
  - Replay authenticators
- Active Adversary
  - Can see and modify traffic between client and server
  - Man-in-the-middle attack

Hints for Web Client Authentication

- Use Cryptography Appropriately
- Protect Passwords
- Handle Authenticators Carefully

Use cryptography appropriately

- Appropriate amount of security
  - Keep It Simple, Stupid
- Do not be inventive
  - Designers should be security experts
- Do not rely on the secrecy of a protocol
  - Vulnerable to exposure
- Understand the properties of cryptographic tools
  - Example: Crypt()
- Do not compose security schemes
  - Hard to foresee the effects

Crypt()

username	`crypt()` output	authentication cookie
bitdiddle	MaRdw2J1h6Lfc	bitdiddleMaRdw2J1h6Lfc
bitdiddler	MaRdw2J1h6Lfc	bitdiddlerMaRdw2J1h6Lfc

Protect Passwords

- Limit exposure
  - Don’t send it back to the user (much less in the clear)
  - Authenticate using SSL vs. HTTP
- Prohibit guessable passwords
  - No dictionary passwords
- Reauthenticate before changing passwords
  - Avoid replay attack

Handle authenticators carefully

- Make authenticators unforgeable
  - highschoolalumni.com
  - If using keys as session identifier: should be cryptographically random
  - Protect from tampering (MAC)
- Protect authenticators that must be secret
  - Authenticator as cookie
    - Sent by SSL
    - Don’t forget the flag! (SprintPCS)
  - Authenticator as part of URL

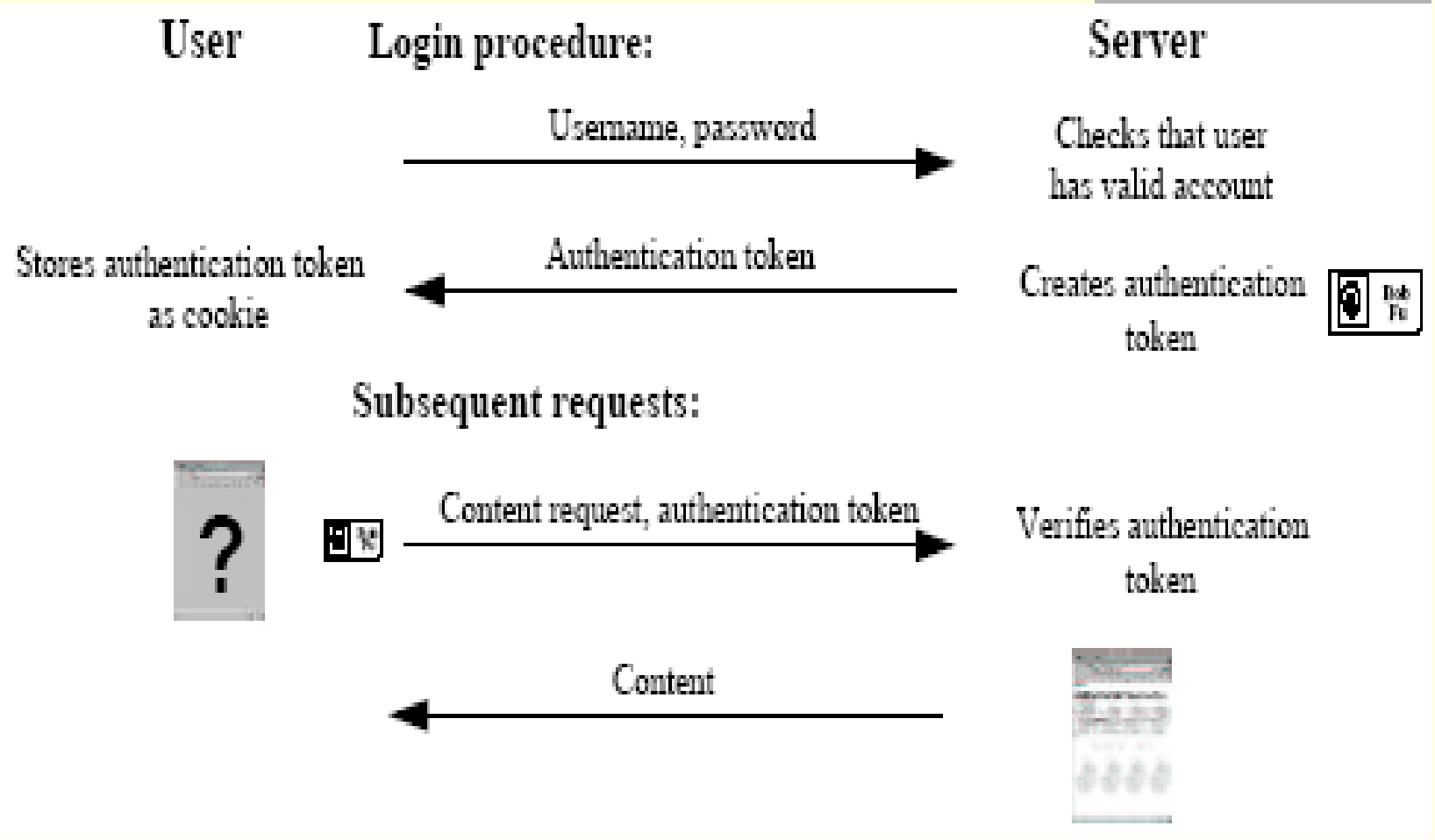
Handle authenticators carefully (cont.)

- Avoid using persistent cookies
  - Persistent vs. ephemeral cookies
  - Cookie files on the web
- Limit the lifetime of authenticators
  - Encrypt the timestamp
  - Secure binding limits the damage from stolen authenticators
- Bind authenticators to specific network addresses
  - Increases the difficulty of a replay attack

Their Design

- Provides request and content authentication
- Stateless
- Secure against interrogative adversary
- On top of SSL: secure against an active adversary

Their Design (cont.)



Their Design (cont.)

- Cookie Recipe

  $exp=t\&data=s\&digest=MAC_k(exp=t\&data=s)$

  - Requires non-malleable MAC
    - HMAC-MD5
    - HMAC-SHA1
  - Timestamp tradeoffs

Authentication and Revocation

- Authentication

  →retrieve cookie's timestamp. If valid,

  →recalculate the MAC in the digest

- Revocation
  - Relies expiration timestamp
  - Revoke all authenticators
    - rotate server key

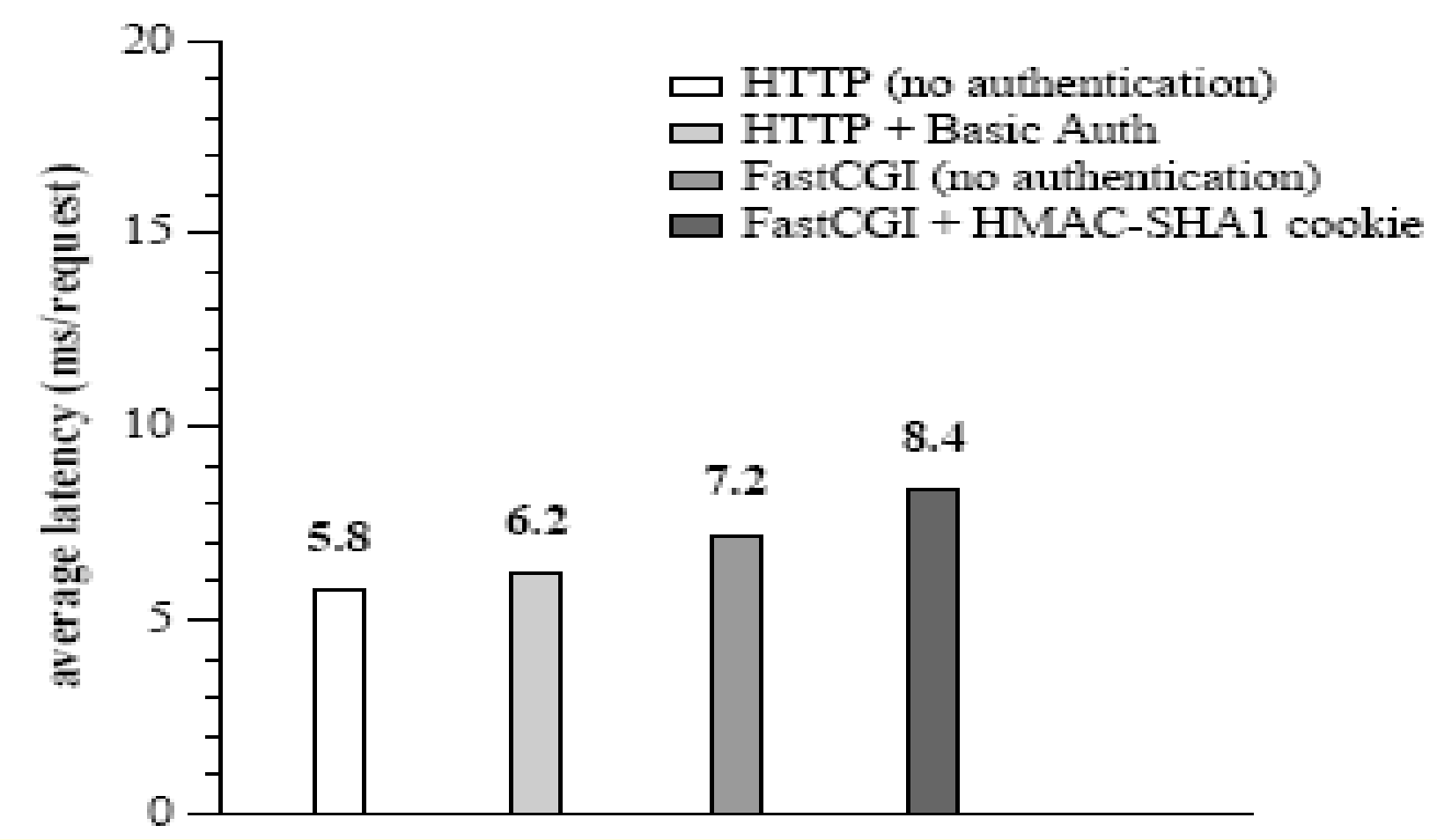
Security Analysis

- Forging Authenticators
  - Adversary tries to forge a new authenticator
  - Adversary tries to extend authenticator capabilities
    - Modify expiration
    - Modify data string
  - Adversary fails
    - Used non-malleable MAC
    - Verifier cannot be calculated by adversary without the key

Security Analysis (cont.)

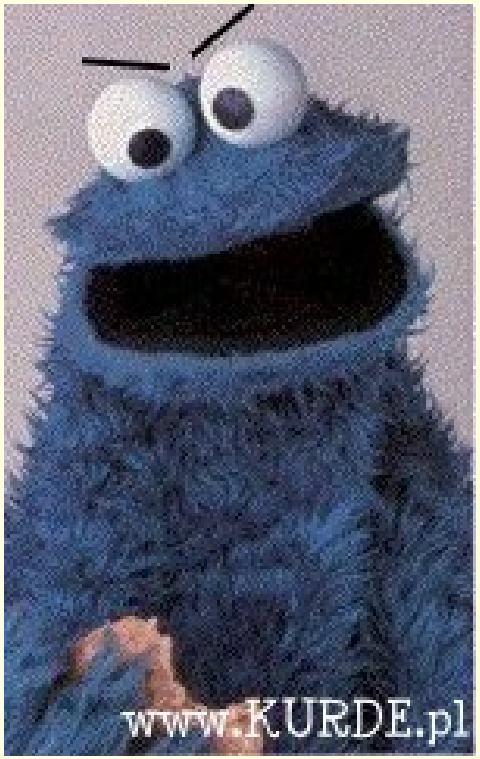
- Authenticator hijacking
  - Eavesdropping adversary can perform a replay attack
  - Limited duration attack
    - As long as the expiration
  - SSL can provide confidentiality
    - Eavesdropper fails
- Brute force
  - Cannot get the key to hash function from the cyphertext
  - Rotate the key

Implementation Performance



Conclusion

- Client authentication is commonly required
- Many schemes are weak
- Authors propose a set of simple hints
  - Appropriate use of cryptography
  - Passwords must be protected
  - Authenticators must be protected
- Authors' design secure against interrogator adversary
  - Also against active adversary if on top of SSL



Any questions?