ForceHTTPS: Protecting High-Security Web Sites from Network Attacks

Collin Jackson and Adam Barth
HTTPS and Network Attackers

• High-security sites employ HTTPS
  – Protects against active network attackers
  – Passwords encrypted
  – “Secure” cookies kept confidential

• Especially important for wireless networks
HTTPS Certificate Errors

• Low-security sites
  – Self-signed certs
  – Passive attackers

• Cert errors common
  – Browser shows warning
  – Users override errors

• Misconfig or attack?
  – Browser doesn’t know
  – User doesn’t know
Strong Threat Model

• Active network attacker
  – Controls the network
  – Has a certificate for attacker.com
  – Does not have a certificate for bank.com

• User click through certificate errors
  – Only type bank password at https://bank.com
  – Second factor in Secure cookie (e.g., BofA SiteKey)

• Realistic: Wireless networks
Related Work: WSKE

- Web Server Key-Enabled Cookies
  - Secure cookies only sent for same TLS key
  - Intended to secure the user’s second-factor cookie
Related Work: Locked SOP

• Locked same-origin policy
  – “Broken” HTTPS page can’t script valid HTTPS page
  – Sites cannot use `<script src="…"`, CSS, SWF, etc

• Importing libraries ignore scripting policy
  – `<script src="https://www.paypalobjects.com/…">`
  – User clicks through cert error for paypalobjects.com
  – Real PayPal imports script from paypalobjects.com
  – Attacker runs script as “unbroken” PayPal
Related Work: Firefox 3

- Firefox 3 – Four clicks
  - User override harder
  - Controversial balance
    - Security
    - Compatibility
  - Low-security sites
    - Harder to use
  - High-security sites
    - User can still override

- How will users react?
Our Proposal: ForceHTTPS

• Site sets a “ForceHTTPS” cookie
  – Opt in to strict error processing
  – Not interested in compatibility
  – Treat errors as an attack, not a misconfiguration

• Specification
  – Non-HTTPS connections redirect to HTTPS
  – HTTPS errors treated as fatal
  – Importing non-HTTPS content (mixed content) fails
Case Study: Gmail

- Login form always over HTTPS
- Mail available over HTTP and HTTPS
- Imperfect web developers
Gmail and SafeBrowsing

- New account, always visited over HTTPS
- Compromised by passive network attacker
Case Study: PayPal

• Entire website over HTTPS
  – HTTP redirects to HTTPS
  – Cert errors on some dark corners…

• Links on home page point to HTTP…
  – Not necessarily a vulnerability
Implementation: ForceHTTPS

• Firefox extension
  – Monitors all network connections
  – Blocks connections with cert errors for sites that opt-in
  – Blocks mixed contents for sites that opt-in

• Useful debugging tool
  – Logs to developer console
  – Found many issues with real sites just by browsing
  – Want to extend to combine with a web app scanner
Trick: Scheme Relative URLs

• Mixed content is hard to eliminate
  – Often host same content over HTTP and HTTPS
  – Only want to pay for HTTPS when needed

• Consider embedding scripts
  – <script src="http://a.com/foo.js"></script>
  – <script src="//a.com/foo.js"></script>

• Works in all browsers
  – Used extensively by Slashdot to save bandwidth
Conclusions

• Browsers trade off security for compatibility
  – High-security sites want more security
  – Browser can be stricter if sites opt-in
  – Simple kind of “content restriction”

• ForceHTTPS
  – “Please enable strict HTTPS error processing”
  – Strong threat model, difficult to get mechanism right
  – More details in the paper
    • Denial of service, error recovery, cookie integrity, privacy, etc