7th USENIX Workshop on Offensive Technologies

WOOT '13

AUGUST 13, 2013
WASHINGTON, D.C.

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

# Truncating TLS Connections to Violate Beliefs in Web Applications

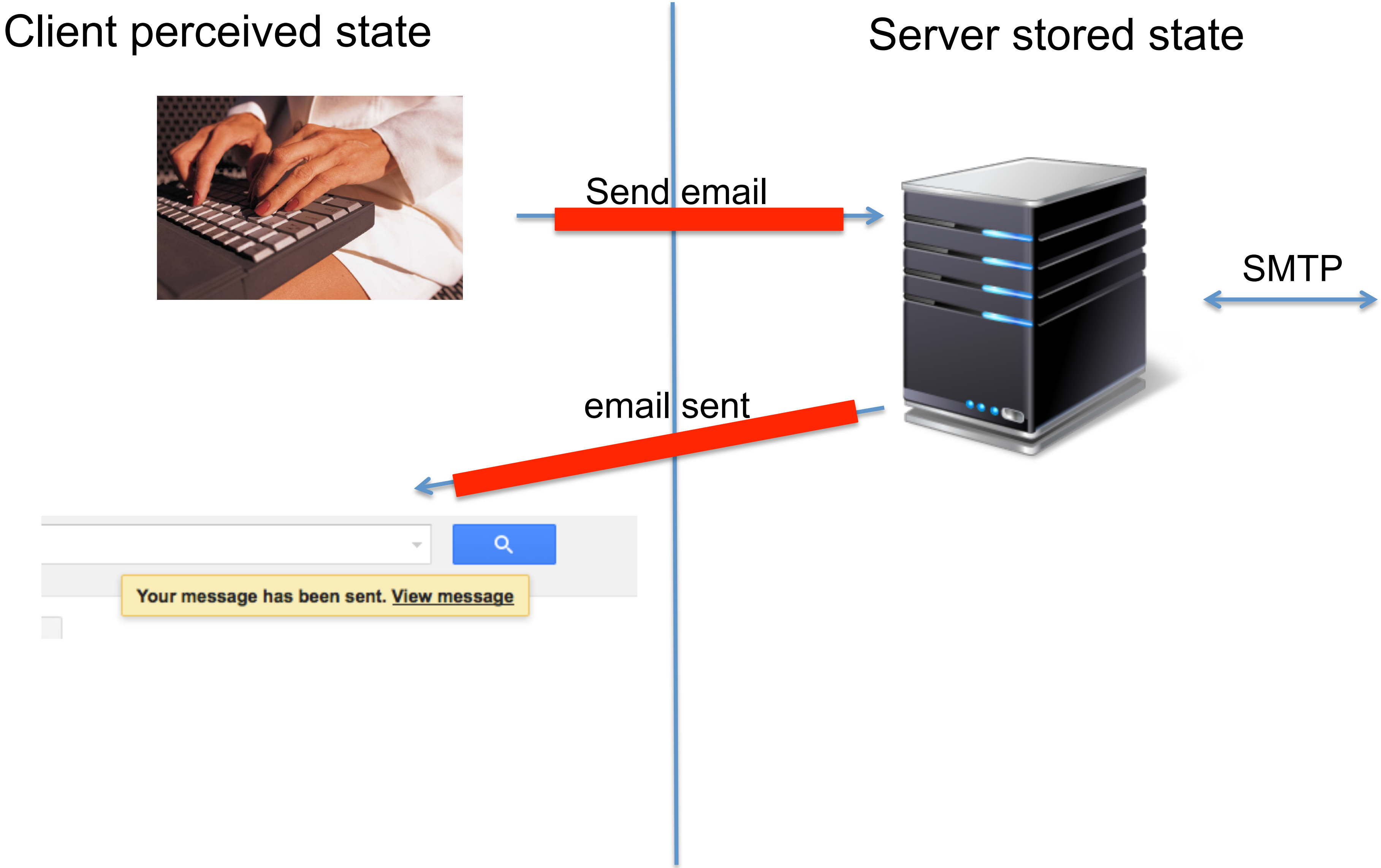
Ben Smyth & Alfredo Pironti

*WOOT*
13 Aug 2013

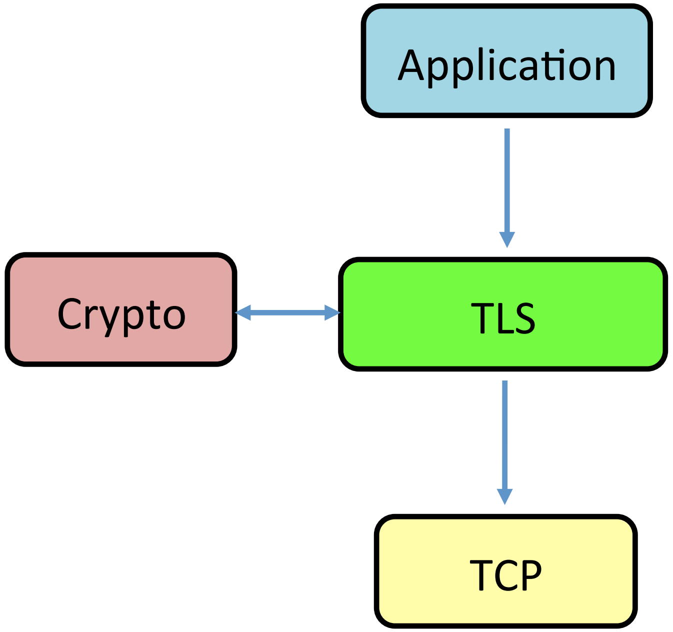
http://www.bensmyth.com
http://alfredo.pironti.eu/research/

Inria
INVENTEURS DU MONDE NUMÉRIQUE

# Web application state




# TLS security




Security:
- Server (and client) authentication
- Confidentiality
- *Integrity: messages received as sent*
  - single connection

Termination modes:
- *Graceful closure*
  - all messages received as sent
- *Fatal closure* e.g. after a corrupt message
  - a prefix of messages received as sent

# Truncating TLS connections

*"failure to properly close a connection no longer requires that a session not be resumed [...] to conform with widespread implementation practice"*

RFC 5246 – TLS specification

Consider a wire transfer to *"Charlie's Angels"*:

POST /wire_transfer.php HTTP/1.1
Host: mybank.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 40
amount=1000&recipient=Charlie%27s_Angels

Suppose the request is fragmented by TLS
1)POST […] recipient=Charlie
2)%27s_Angels

Attack: Drop the 2nd fragment to transfer money to Charlie.

Server ignores:

- termination mode
- Content-Length field

Fix:

- wire transfers upon *graceful closure* only
- check lengths

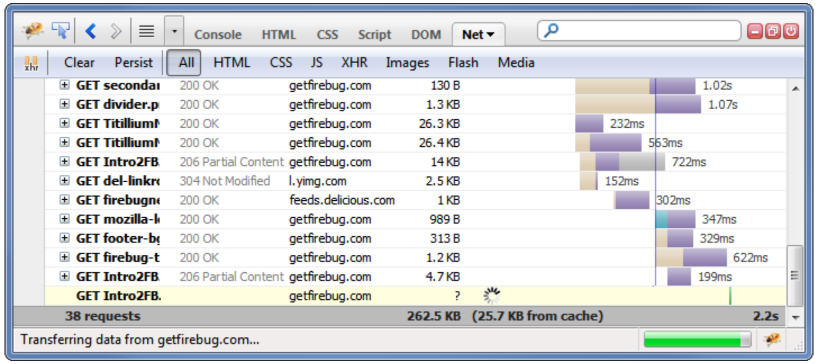
Attack works against Apache

Henceforth, we consider truncation attacks which drop messages, rather than fragments

# Challenges for web applications

Web applications:

- Browsers maintain multiple connections (to load content in parallel, for example)




TLS provides:

- No integrity guarantees across multiple connections
  - hence, ordering issues between connections

Adversary model (standard):

- Adversary has full control of the network
  - i.e. read, delete, and inject messages

# Contribution

Attacks which truncate TLS connections to exploit logical web application flaws, enabling:

- Cast votes [on behalf of honest voters] in Helios elections
- Full control of Microsoft Live accounts
- Temporary access to Google accounts

We suspect our insights will lead to the discovery of further attacks.

# Helios electronic voting system

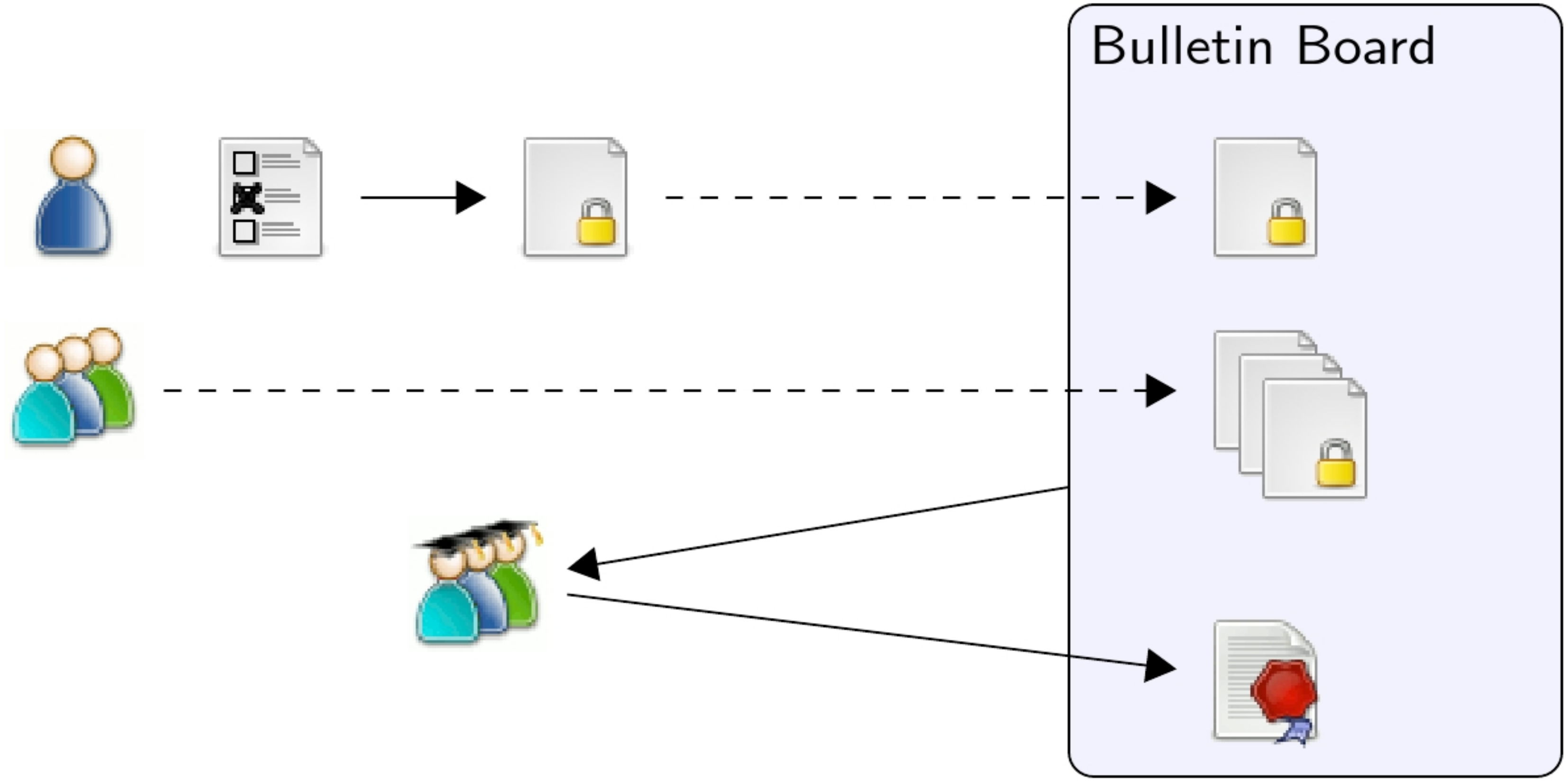
Helios is a verifiable e-voting system

- Catholic University of Louvain 2009 presidential election:
  - ~4000 votes / 25000 voters
- IACR 2011+onwards board election
  - 621 votes / 1484 voters (2011)
- Princeton University 2009+onwards for student government

Cryptographic proofs of security!


*Verifiability enables us to use untrusted voting machines and check afterwards that the claimed result is valid*

# Helios: Overview




Ballot construction and authentication handled by a voting machine
Permits *re-voting*: cast arbitrarily many ballots/count last

# Helios: Ballot casting

```
1)   REQUESTS https://vote.heliosvoting.org/helios/elections/<<id>>/cast_done
     Response: 200 – OK; HTML payload:
       …
       <p><b>For your safety, we have logged you out.</b></p>
       <iframe border="0" src="/auth/logout" frameborder="0" height="0" width="0">
       </iframe>
       …

2)   REQUESTS https://vote.heliosvoting.org/auth/logout
     Response: 302 – Moved Temporarily
     Location[http://vote.heliosvoting.org/]
```

Notification of sign-out *before* voting machine makes the request!

3) Truncate sign-out request
4) Use voting machine to cast a new vote

No TLS protection: sign-out request (2) and adversary (4) use different connections. Fix: (1) & (2) atomic.


A video demonstrating this attack will be available online.

# Microsoft Live accounts

Setting:

- *Shared computer* (e.g., public library, work place, …)
  - Trusted computer, i.e., not tampered with
  - Adversary accesses computer after honest user has finished

## *Video Demo*

(Live demos are too stressful!)

The video will be available online.

# Microsoft Live accounts

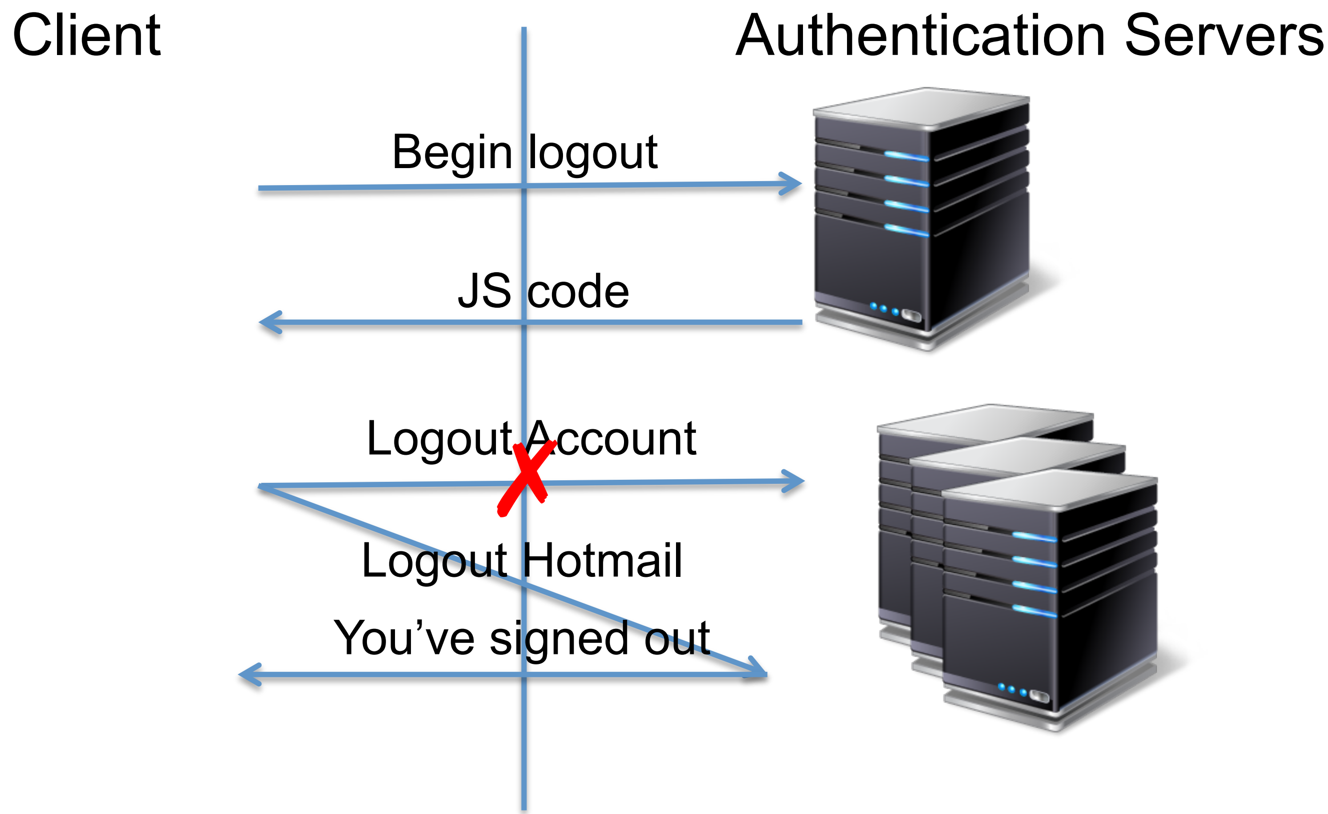
Setting:

- *Shared computer* (e.g., public library, work place, …)
  - Trusted computer, i.e., not tampered with
  - Adversary accesses computer after honest user has finished

Notification of sign-out *before* server receives request (client's belief ≠ server's belief)!

- Truncate sign-out
- Access account on another connection

# Microsoft Live accounts




Fixes:

- *Centralize authentication*; or
- *Chain sign-out requests*

# Google accounts

Setting: *Shared computer* (e.g., public library, work place, …)

```
1) GET https://accounts.google.com/Logout?continue=https://www.google.com/webhp
   Response: 302 - Moved Temporarily,
   Location[http://www.google.com/accounts/Logout2?ilo=1&ils=mail,s.FR&ilc=0&
            continue=https://www.google.com/webhp?zx=1388193849]

2) GET http://www.google.com/accounts/Logout2?ilo=1&ils=mail,s.FR&ilc=0
       &continue=https://www.google.com/webhp?zx=1388193849
   Response: 200 - OK; HTML payload:
    <body onload="doRedirect()">
     <script type="text/javascript">
      function doRedirect() {
       location.replace("http://www.google.fr/accounts/Logout2?ilo=1&ils=s.FR&
           ilc=1&continue=https://www.google.com/webhp?zx=1076119961");
      }
     </script>
     <img width="0" height="0" alt="Sign Out"
          src="https://mail.google.com/mail?logout=img&zx=-2531125006460954395">
    </body>

3) GET https://mail.google.com/mail?logout=img&zx=-2531125006460954395
   Response: 200 - OK; a one pixel gif.

4) ...
```

# Google accounts: Attack

Setting: *Shared computer* (e.g., public library, work place, …)

```
<body onload="doRedirect()">
    <script type="text/javascript">
     function doRedirect() {
      location.replace("http://www.google.fr/accounts/Logout2?ilo=1&ils=s.FR&
          ilc=1&continue=https://www.google.com/webhp?zx=1076119961");
     }
    </script>
    <img width="0" height="0" alt="Sign Out"
         src="https://mail.google.com/mail?logout=img&zx=-2531125006460954395">
   </body>
```

Notification of sign-out *before* server receives request!

- Truncate Gmail sign-out with *TCP reset*
  - (TCP drop hangs the browser)
- Fatal connection closure *ignored*
- Access Gmail on another connection
  - House-keeping terminates (~5mins)

Fixes:

- Handle fatal connection closure; or
- Centralize auth. or chain sign-outs

A video demonstrating this attack will be available online.

# Summary

- We exploit flaws in sign-out procedures to prevent termination of sessions, whilst notifying the user of success.
  - Attacks against Helios, Google & Microsoft

- Consequently, even *trusted* shared computers offer no security!

- Fixes proposed, therefore trusted shared computers offer security.

- All vulnerabilities have been disclosed; but none have been fixed yet.


- De-synchronization of client/server state as attack vector.
  - Further attacks?
  - Better programming practices?

Thank you!

Questions?

http://www.bensmyth.com
http://alfredo.pironti.eu/research/

INVENTEURS DU MONDE NUMÉRIQUE