Exposing Private Information by Timing Web Applications

Andrew Bortz
Dan Boneh
Palash Nandy
Privacy on the Web

✓ We create more private data every day
✓ We actually care about our privacy
✓ Almost every week we hear news of a new leak of private information
✓ And of course, there is always phishing
Context-Aware and Spear Phishing

✓ Phishing is equally social and technical
✓ The more knowledgeable the attacker is, the more successful the attack is
  ✓ Target the right users for the right attacks
  ✓ Parrot back “secrets” that establish user trust
Existing Timing Attacks

✓ In general, timing is a well-known attack vector:
  ✓ Hard to solve easily and efficiently

✓ On the web, timing is not well known:
  ✓ Exceptions include Felten et al. timing attacks on browser cache

✓ So, is a web application actually vulnerable to timing?
Web Application Architecture

Internal traffic

Misc
Apache
SQL

Web Application

Request
Response

Web Browser

Time

Direct Timing Attack

Web Site

Custom Request

Timing Program

Real Attack: Valid User Profiling

✓ Most sites recognize it’s a bad idea to reveal if an account exists or not
✓ Use timing data to distinguish valid from invalid
Experimental Data

- Two successful attacks:
  - Take the smallest of 10 timing samples per account tested
  - >95% accuracy in distinguishing
  - More samples = better accuracy
Irony: Forget Your Password?

✓ Not even necessary to time sites that give this information away
✓ Some sites try to protect with CAPTCHAs
✓ Definitely not the right solution
Other Direct Timing Attacks

- Other hidden booleans:
  - Hidden profile data, etc.
- Can we do better than boolean quantities?
  - Count hidden blog entries in public blogs
  - Count hidden photos in photo galleries*
Cross-Site Timing Attack

Attacker’s Timing Page

#1 - Visit

#2 - Time

Target Site

Reference Page

Test Page

User’s Browser

#2 - Time
Cross-Site Methods

✓ With JavaScript:
  ✓ In-browser timing using frames
    ✓ Most obvious, not very accurate
  ✓ In-browser timing using images
    ✓ Less obvious, very accurate

✓ Without JavaScript:
  ✓ Server-based timing using special tags
    ✓ Least obvious, somewhat accurate
Sample Attack Code

```html
<html><body><img id="test" style="display: none">
<script>
    var test = document.getElementById('test');
    var start = new Date();
    test.onerror = function() {
        var end = new Date();
        alert("Total time: " + (end - start));
    }
    test.src = "http://www.example.com/page.html";
</script>
</body></html>
```
Real Attack: Visitor Profiling

- Decide which sites the user:
  - Visits or doesn’t visit
  - Is logged into or not
  - Possibly other information contained in the cookie
- Previous techniques use browser history
  - Can be stopped by fixing the browser (ie. SafeHistory)
Experimental Data

2 samples per page
Real Attack: List Counting!

✓ Count how many:

✓ Emails in a web email account
✓ Transactions in a bank account
✓ Auctions at an auction site
✓ Cart items at an online retailer
✓ Search results anywhere
✓ Etc, etc. (Use your imagination)
Experimental Data

- This attack counts the number of items in a shopping cart
- 10 samples per page
- Exact up to 10 items, beyond that it becomes approximate
Other Cross-Site Attacks

- Combination attacks with cross-site request forgery:
  - Use forged request to modify data, time to determine if the data changed, then another forged request to restore data
  - Example: Decide if a user has a specific item in their shopping cart!
Defenses

✓ Mostly a server-side problem:
  ✓ Adding random noise never good enough

✓ Server-side solution
  ✓ Make everything take the same time
  ✓ Manually recode, or use mod_timepad*

✓ Client-side solution
  ✓ Stop all cross-site timing methods
Recap

✓ Two new timing attacks on the web
  ✓ Widespread vulnerabilities
  ✓ Practical and valuable for:
    ✓ Spammers, phishers, and less-than-reputable advertisers
  ✓ Huge potential for additional attacks
✓ Some preliminary defenses
  ✓ Not wholly satisfactory, but a step forward