Accelerating Information Technology Innovation

http://aiti.mit.edu

India Summer 2012
Lecture 11 – App Security

Securing Your Apps!

(or: how to avoid losing your customers trust)

Case Study:
Sony PlayStation Network

- An old unpatched security hole on a Sony server gave access to PSN's user database. [1]
- Hackers gained information on 77 million users and had access to over 10 million credit cards [2]
- Shut down parts of website for 23 days [3]
- Didn't alert users for 6 to 8 days! [2]
- Cost of upwards of $171 million to Sony [3]

Why Security?

- Android is the most targeted mobile platform for security attacks!
- Many passwords/personal info stored on web servers/web applications.

Threat Models

What do you think people will try to attack/steal?

App Security

- Sign Your Apps
- Don't Trust Outside Data
- Don't Prompt for Passwords (Often)
- Only Send/Record What You Must
- Keep the User Informed

Sign Your Apps

- *Threat Model:* Someone releases an "update" to your app that steals users' passwords.
- Sign your app with a digital certificate which identifies that it came from you.
    - Required for submission to many app stores!

KEEP YOUR PRIVATE KEYS SAFE

Sign Your Apps

DON'T SHARE YOUR KEYS!

KEEP THEM IN A SAFE PLACE!

Don’t Trust Outside Data

- *Threat Model:* Someone sends you bad data to crash your program (or steal data!)
- Always check your inputs (from local content providers or the Internet!)
    - Are they null?
- Define your own permissions.

Don’t Prompt for Passwords (Often)

- *Threat Model:* Someone makes a lookalike app that asks for a password to steal one!
- Take a password once and cache a local authentication token (like a cookie).
- Refresh the authentication token often.

Only Send/Record What You Must

- *Threat Model:* Someone uses a flaw in your application/server to steal IMEIs so they know people who use your app!
- Don't identify users by phone numbers
  - Hash, or generate a unique identifier
  - Don't use IMEIs either.
- Don't keep location/payment info for long.

Keep the User Informed

- The user might not trust your app.
- Build trust by being open about what you collect and what you use it for.
- Have a Privacy Policy
  - Make it readily known to your users.
  - Inform them of changes in plain language.
- Android forces use of permissions.

Web Security

- Secure Your Passwords
- Access Control for Sensitive Pages
- Check Your Input Data
- Adding Encryption
- Secure Your Cookies
- Prevent Your Data from Leaking
- Protect Your User

Secure Your Passwords

- *Threat Model:* Attackers may try to steal users' passwords to pretend to be users.
- *ALWAYS* hash *AND* salt passwords
  - Hash keeps passwords from being plaintext.
    - e.g. Yahoo password leak
  - Salt keeps passwords from being easily looked up in "rainbow tables" (reverse lookup of hash)
    - e.g. LinkedIn password leak
  - Django does this too.

Access Control for Sensitive Pages

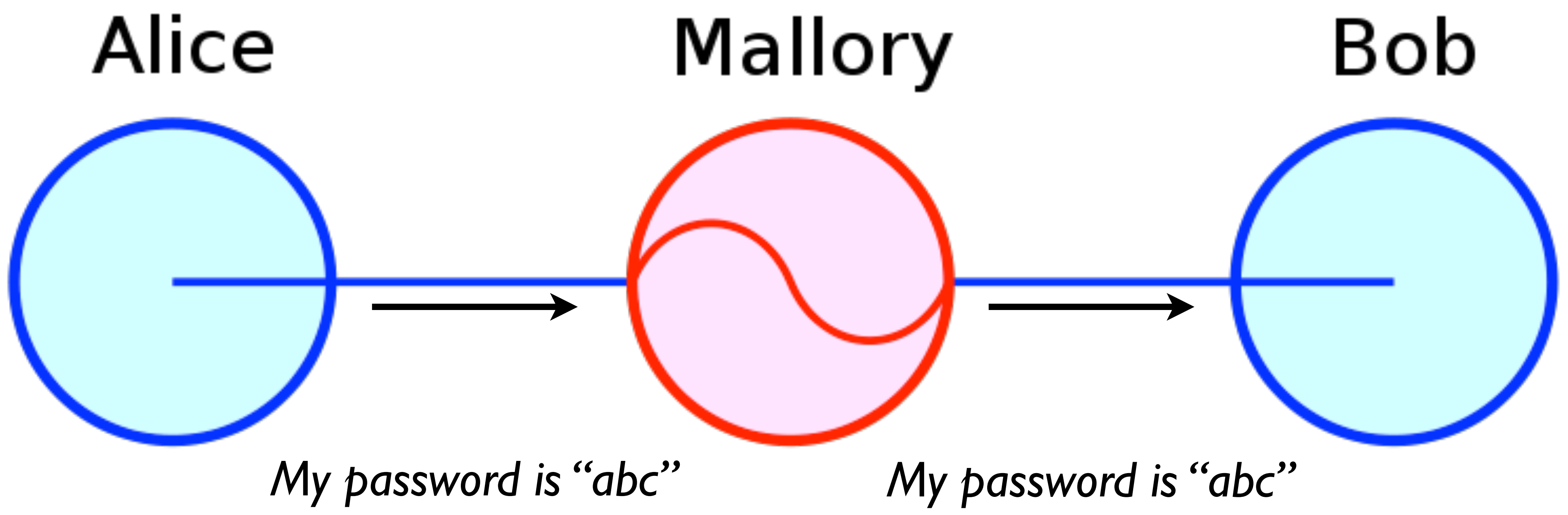
- *Threat Model:* People may guess hidden "delete" or "edit" pages to try to change site data.
- Use access control to restrict who can access a page
    - Authenticate the user and authorize their access
    - Django has access control if you want
- Or just don't implement edit/delete pages!

Check Your Input Data

- *Threat Model:* Attacker might change cost from positive to negative to "pay" negative money (you pay him for his use of service!)
- Never trust your user's data!
  - Validate yourself, don't trust Django, although Forms are better than nothing.
  - Escape data for SQL (prevent SQL injection)

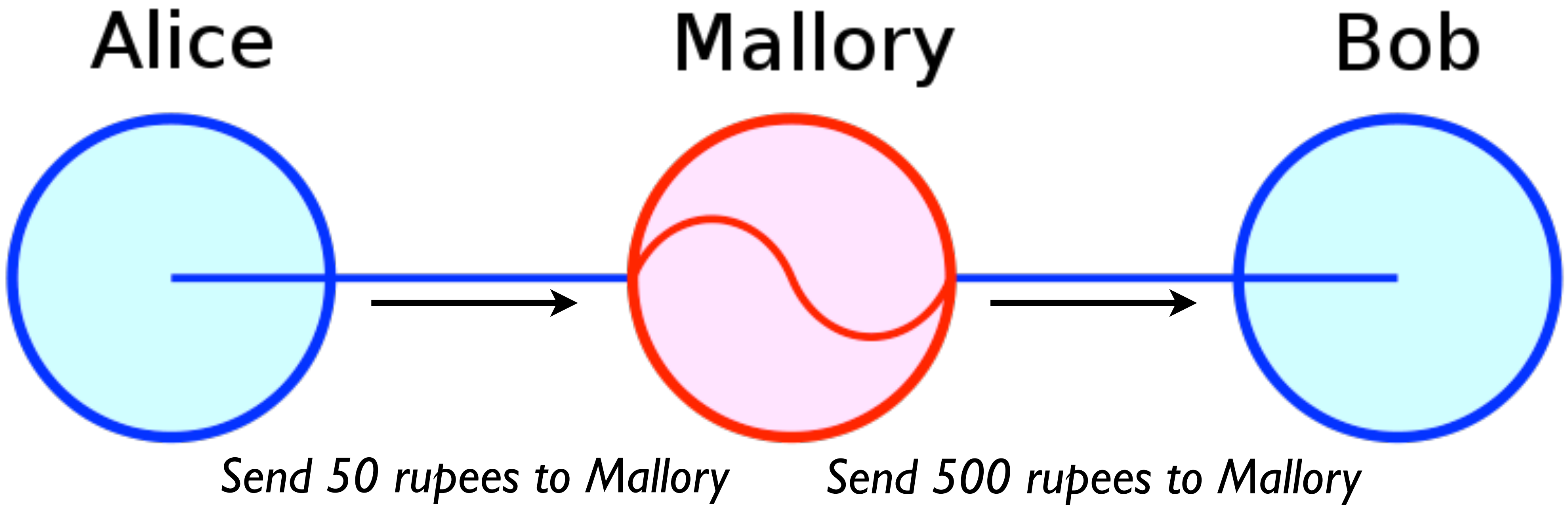
Adding Encryption

- *Threat Model:* Someone might listen to the data between you and your customer.
- Use SSL to encrypt private communication
  - Passwords, payment info, addresses



Adding Encryption

- *Threat Model:* Someone might pretend to be you to alter data from a customer.
- Use SSL to encrypt private communication
  - Passwords, payment info, addresses



Secure Your Cookies

- *Threat Model:*
  - Cookies identify the user and "save" the login.
  - Other websites could force users to do actions without their knowledge via cookies (cross-site request forgery)
- Use a secret key to generate unique CSRF tokens that cannot be forged.
  - Django does this, if you don't share (and randomize) your SECRET_KEY

Prevent Your Data from Leaking

- *Threat Model:* People may use your site against you; run their code from your site!
    - Cross-site Scripting (XSS)
- Always clean and escape the HTML data you show users.
    - Django does by default, but you should check!
- Don't use eval()!

Prevent Your Data from Leaking

- *Threat Model:* People may try to use a hole in your software to get a command-line or system files
  - Root Exploit
- Keep your software/libraries up to date!
  - App Engine should do this, but just in case...

Protect Your User

- *Threat Model:* An attacker fakes a Facebook page to steal their login info (Phishing)
- Build trust with your user.
  - Use their name, remind them that you never request login information by e-mail
- Let user select a custom image to know it's from your site!
  - Much better: Two-factor authentication

References

- Android Developer Site: "Designing for Security" <http://developer.android.com/guide/practices/security.html>
- Android Developer Site: "Permissions" <http://developer.android.com/guide/topics/security/permissions.html>
- "Android Security Overview": <http://source.android.com/tech/security/index.html>
- "Mobile Application Security": <http://www.cio.ca.gov/OIS/Government/events/documents/Mobile_Application_Security.pdf>
- "Google Code University: Web Security" <http://code.google.com/edu/security/index.html>
- "Mobile Web Application Best Practices" from the W3C: <http://www.w3.org/TR/mwabp/>

Credits

- More about the Sony breach can be found via Wikipedia: <http://en.wikipedia.org/wiki/PlayStation_Network_outage>
- The web security segment is made with apologies to Victor Costan, whose presentation “Security for Web Applications” served as a more detailed model: <http://courses.csail.mit.edu/6.857/2012/files/L06-Costan-web-security/html/all.html>
- The image on slides 17 and 18 is “Man in the middle attack” by Miraceti <http://commons.wikimedia.org/wiki/File:Man_in_the_middle_attack.svg> It is licensed under a Creative Commons Attribution-Share Alike 3.0 Unported license.