Security Considerations for Mobile Apps and APIs

John DiLeo, D.Sc. (@gr4ybeard)
OWASP New Zealand - Auckland
August 2019

“Well…how did I get here?”

- Born and raised in northeastern US
- Straight to university from high school
  - B.S.E.E. in Computer Engineering
  - No relevant summer jobs or internships
- Started first job four weeks after graduation
- Swore would never return to school

“Well…how did I get here?”

- Phase 1: Operations Research / Simulation
  - – US military systems (Army/DoD)
    - Other duties: TEMPEST assessments, Unix admin.
  - – US air traffic control (FAA)
  - – Two Master’s degrees, part-time
  - – Doctoral research: FreeSML simulation “language”
- This phase lasted 17 years

“Well…how did I get here?”

- Phase 2: Java/Web App Development
  - Tinkering: Java Applets, CGI Scripts, etc.
  - Studies: Programming Languages course
  - Teaching: Info. Systems, Programming, Maths
  - Building server-side applications
    - US Dept. of Agriculture – Farm subsidy programmes
    - Java, JSP, Struts, Spring, etc.
    - “Heavyweight” Web Services – SOAP, EJBs
  - Shifted to Architect roles – mostly AFK since
- This phase lasted 18 years

“And you may find yourself in another part of the world”

- Phase 3: Application Security
  - Started with teaching…again
  - Cross-trained: GSSP-Java, GSEC, CEH
  - Secure Coding → Software Assurance
  - Moved to New Zealand in 2017
- This phase has lasted six years, so far…
  - BTW…it has NOT been 41 years since graduation
  - There *were* overlaps

OWASP
Open Web Application Security Project

“And you may find yourself in a beautiful house…”

- Joined Orion Health in December 2017
  - I *am* the Application Security Team
  - And usually work out of Orion House, in Auckland
- Orion specialises in healthcare information systems
  - Electronic Medical Records
  - Healthcare Analytics
  - “Precision Medicine” (Machine Learning)
  - PHI protection has to be a high priority
- Customers world-wide
  - District/Regional Health Boards
  - Private Health Insurers
  - Hospitals
  - Health Information Exchanges (HIEs)

Sidebar: Orion Heath is Hiring

- Current headcount: 650+
  - Development teams in Auckland, Christchurch, Canberra, Bangkok, Montreal
  - Solution implementation teams worldwide
- Hiring Intern and Graduate Developers
  - Working in Auckland (for Grads, initially)
  - Apply through: summeroftech.co.nz
  - CV review already ongoing
  - Interviews 17 September, in Auckland
  - Offers out in early October

"Letting the days go by…"

Then…I got involved in OWASP

- OWASP Kansas City Chapter
  - Spoke up at Meetups
  - Invited to join Chapter Steering Committee
- OWASP New Zealand Chapter
  - Attended OWASP NZ Day
  - Filled vacant role as Auckland-area Leader
- OWASP Projects
  - Software Assurance Maturity Model (SAMM) Project
  - Co-Leader, AppSec Curriculum Project

OWASP Activities and Events

- Global AppSec Conferences
  - December 2020: Tokyo (tentative)
- Regional AppSec Conferences
  - AppSec Days, Sydney
    - Training: 28 – 31 October
    - Conference: 1 November
- Meetups - Auckland, Christchurch, Wellington
- Chapter Mailing List

  To join: https://groups.google.com/a/owasp.org/forum/#!forum/new-zealand-chapter/join
- InfoSecNZ Slack (infosecnz.slack.com)

OWASP New Zealand Day

- University of Auckland Business School
  - Training: 19 – 20 February
  - Conference: 21 February – Still FREE!
- Some travel “scholarships” will be available
  - Applications will open 1 December
- Training
  - Fees higher this year
    - Half-day class: $325
    - One-day class: $625
    - Two-day class: $1250
  - But…watch for future news

OWASP New Zealand Day Sponsors to Date

INSOMNIA
SECURITY SPECIALISTS :: REST SECURED

QUANTUM SECURITY

SECURE CODE WARRIOR

LATERAL SECURITY

SEQA™
Information Security

OWASP
Open Web Application Security Project

And now…this

Something, Something, Mobile App Security

OWASP Resources

- Web Site – https://www.owasp.org
- Mobile Security Project
  - Mobile Top Ten
  - Mobile Security Testing Guide (MSTG) (LeanPub)
  - Mobile AppSec Verification Standard (MASVS) (PDF)
  - Mobile Application Security Checklist (GitHub)

OWASP Mobile Top 10 (2016)

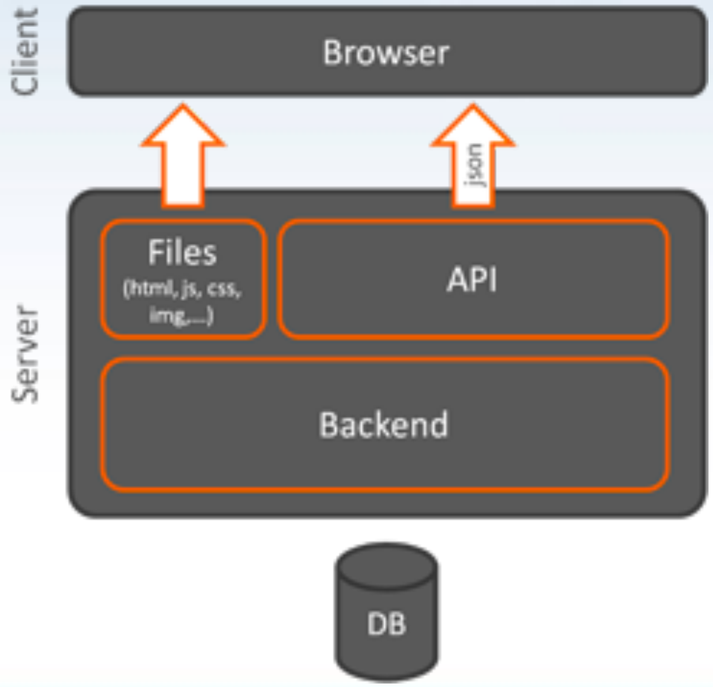
M1 – Improper Platform Usage
M2 – Insecure Data Storage
M3 – Insecure Communication
M4 – Insecure Authentication
M5 – Insufficient Cryptography
M6 – Insecure Authorization
M7 – Client Code Quality
M8 – Code Tampering
M9 - Reverse Engineering
M10 – Extraneous Functionality

Mobile and Client-Side Apps

Mobile apps and client-side applications have a lot in common

- Emphasis on responsive user experience
- Business logic executes in end-user device
- Rely on "back-end" service requests to obtain/persist data

Much of what we'll look at really applies to both



Considerations when Building Apps

- SECURITY and PRIVACY – Design it in from the start!
- User experience and useability
- Performance
- Platform(s) to support
- Testing approach
- Monetization
  - Payment processing
  - Users: The customer or the product?
- Future-proofing
  - Scalability
  - Reliability
  - Updates and patching

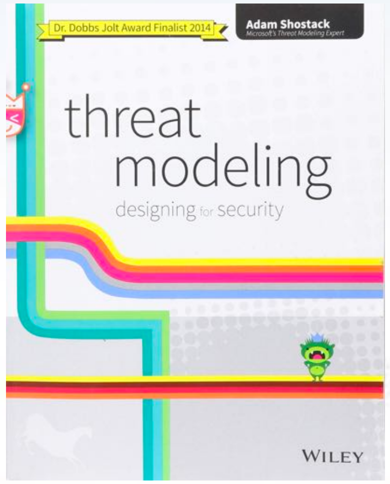
OWASP
Open Web Application Security Project

Security Mobile Apps
What should I worry about?

Well…what's in the **threat model**?

"Four Questions" Approach (Adam Shostack)

1. What are we building?
2. What could go wrong (threat)?
3. What can we do about that (mitigation)?
4. How did we do?
   - Verify mitigations
   - Validate model


Dr. Dobbs Jolt Award Finalist 2014
Adam Shostack
Microsoft's Threat Modeling Expert
threat modeling
designing for security
WILEY

What are we building?

- Mobile app
  - Our source code (usually proprietary)
  - Core platform and build system
  - Third-party libraries
  - Local data storage, including keys/credentials
  - Device function interfaces (camera, GPS, etc.)
- REST APIs
- Data
  - Users
  - Subjects
  - Transactions → Users’ rights/permissions/abilities/swag

What could go wrong? - Our source code

- Insecure code
  - Injection vulnerabilities
  - Home-built encryption or AuthX system
  - Buffer overflows
  - Memory management issues (leaks)
  - Test mode/test code/demo creds included in releases

Mobile Top 10: *M7 – Client Code Quality*
M10 – Extraneous Functionality

Mitigation: Don't do that!

- Developer training and awareness
- Secure coding standards
- Shared libraries/services
- Automated security testing (static and/or dynamic)
- Code reviews

OWASP
Open Web Application Security Project

What could go wrong?
- Our source code

- Malicious modification
  - Source code, in the repository
  - Executable app – in store
  - Executable app – through unauthorized redistribution

Mobile Top 10: *M8 – Code Tampering*

Mitigation:

- Restrict access to source code repositories
- Restrict access to build-publish pipeline
- Separation of duties in release approval process
- Use application signing
- Distribute only through reputable app stores
- Provenance checking – more challenging

What could go wrong?
- Our source code

- Theft
  - Publication
  - Appropriation
  - Zero-day attacks

Mobile Top 10: *M9 – Reverse Engineering*

Mitigation:

- Restrict access to source code repositories
- Data Loss Prevention (DLP)
- Robust Joiners/Movers/Leavers (JML) processes
- Anti-reverse engineering techniques

What could go wrong?
- Our source code

- Corruption / Destruction
  - Entire code base
  - Recent work
  - Expert knowledge

Mitigation:

- Replication and/or backups of code repositories
  - And test them!
- Developer training: Frequent commits
- Never skip documentation "to save time"
- JML processes, again

What could go wrong?
- Core Platform and Build System

- Vulnerabilities in core platform libraries
- Vulnerabilities in build system components

Mobile Top 10: *M7 – Client Code Quality*

Mitigation:

- Pay attention to various “intelligence channels”
  - “Official” sources: US-CERT, CERT NZ, vendors
  - “Informal” channels: Twitter, Blogs, Reddit (usually faster)
  - “News summary” sources: Slashdot, etc. (usually *slower*)
- Install patches/updates, obtained from trusted sources, in a **reliable, timely** manner

What could go wrong? - Third-Party Libraries

- Vulnerabilities in core platform libraries
- Vulnerabilities in build system components

Mobile Top 10: *M7 – Client Code Quality*

Mitigation:

- Pay attention to various “intelligence channels”
- Install patches/updates, obtained from trusted sources, in a **reliable, timely** manner
- Have a complete inventory of dependencies – including dependencies of dependencies
- Use locked, local mirrors for releases

What could go wrong?
- Local Data Storage (on device)

- Sensitive data / credentials stored insecurely

Mobile Top 10: *M2 – Insecure Data Storage*
M5 – Insufficient Cryptography

Mitigation:

- Leverage device support (e.g., Private mode)
- Encrypt all data
  - Incorporate factor known by user (when possible)
  - Use device-provided support for key storage

What could go wrong?
- Device function interfaces

- App has permissions to access and/or update hardware/data it doesn't need

Mobile Top 10: *M1 – Improper Platform Usage*

Mitigation:

- – Request only the minimum set of permissions required
- – Request permission for "high-value" access only if user requests functionality *requires* it
- – Ensure app responds sensibly, if permission for "high-value" access is denied

What could go wrong? - REST APIs

- Unauthenticated client accesses sensitive data
  - Authentication not implemented / enforced
  - Steal valid credentials
  - Fabricate valid credentials
  - Authentication bypass / race conditions

Mobile Top 10: *M4 – Insecure Authentication*

Mitigation:

- Use strong authentication mechanisms
- Delegate to IDaaS provider when possible
- What Kate said: DON'T create your own!

What could go wrong?
- REST APIs

- Access from stolen device

Mobile Top 10: *M4 – Insecure Authentication*

Mitigation:

– Require additional local authentication (e.g., PIN)

– Disable user's access when theft is reported

What could go wrong? - REST APIs

- Authenticated client accesses unauthorized data
  - Access controls not implemented / enforced
  - Access checks not granular enough
  - Authorization bypass / race conditions

Mobile Top 10: *M6 – Insecure Authorization*

Mitigation:

- Assume NOTHING about client's authorization
- Use robust authorization frameworks
- AVOID creating your own
- Deny-by-default strategy
- Thoroughly test access control rules

What could go wrong?
- REST APIs

- Sensitive data “sniffed” in request/response traffic
  - Client-to-server connections not encrypted
  - Known insecure encryption mechanism used
  - Sensitive data in request URLs
  - Machine-in-the-middle intercepts traffic (TLS Stripping)

Mobile Top 10: *M5 – Insufficient Cryptography*

Mitigation:

- Publish your API, take reported issues seriously
- Use TLS 1.2 or 1.3 only
- Remove server support for insecure ciphers
- AVOID responding to HTTP requests (Port 80)

What could go wrong?
- Data on the server

- Users' personal information stolen
- Transaction data stolen/faked/corrupted

Mitigation:

- NEVER collect, store, or share any information you don't need to
- Follow best practices for databases
  - Encryption
  - Access controls
  - Separation of duties

OWASP
Open Web Application Security Project

Privacy

- Obligation to protect customers' data
  - Personally identifiable information (PII)
  - Bank / credit card information
  - Breach penalties vary by country, but are STEEP
- Do you REALLY need it?
  - If you don't collect it, you can't misuse/lose it
  - If you don't store it, it won't be in a data breach
  - If you don't sell/share it, it won't be in *their* data breach

OWASP
Open Web Application Security Project

john.dileo@owasp.org

@gr4ybeard

Coming Soon:

I’ll join the 21st Century and launch a Blog

It’s called “Gr4ybeard’s Treasure” because… why not?

Questions?

OWASP
Open Web Application
Security Project