(ISC)²

SECURITY CONGRESS | EMEA 2016

In partnership with MIS TI TRAINING INSTITUTE

ADVANCING SECURITY LEADERS

emeacongress.isc2.org

#ISC2CongressEMEA

Mobile Application Infrastructure Security

Michalis Kamprianis

MBA, MSc, CISSP, CCSK, COBIT, Prince2, ISO27001LA, ITIL

https://www.linkedin.com/in/michaliskamprianis

Mobile application categories

- Stand Alone
  - Picture gallery
  - Camera
  - Clock and alarm
  - Notes, Word processor
- Client – Server
  - No significant user data
    - Transport schedules
  - With significant user data and interaction
    - Twitter
    - LinkedIn
    - Tinder

Mobile application security analogy

» What does it mean to protect information **in the mobile application**?

Mobile app security + infrastructure

- What does it mean to protect information **always**?

What are we talking about?

- Security is as strong as your weakest link

(1) You should ensure the mobile application is secure

(2) If you store data in servers, you should protect the servers too

- Hardening, Secure configuration, Patch and Vulnerability management
- Access control, Least privilege, Segregation of duties
- Logging, Monitoring
- Encryption at rest,

(3) And by the way, since you're transferring the data from the application to the servers, protect it en route

- Encryption in transit

An old trick

- Encryption in transit **usually** means SSL
  - And more often than not, that is HTTPS
- Known problem: man-in-the-middle attack
- Known solution: Certificate pinning
  - OWASP has sample code for iOS and Android
  - No solution for Windows Phone

Let’s make sure we do it right!

- Own certificate authority or not?
  - I opt for own; generate the certificates and shut it down!
  - In 2001, Verisign issued fake MS certificates
  - In 2010 Verisign was hacked – no news about the CA
  - Vasco/DigiNotar got hacked in 2011 having issued hundreds of fraudulent certificates
  - KPN/Getronics got hacked in 2007, identified in 2011
  - Comodo’s reseller was hacked, 9 fraudulent certificates issued
  - None of the drawbacks of self-signed certificates is relevant in this case

What else do we add?

- Android – specific : How about client key pairs / certificates?
  - Have either the reverse proxy or plain web server “authenticate” the .apk
  - Use SSL Verify Depth wisely so that ONLY your CA’s certificates are accepted
  - Use Certificate Revocation Lists so that you can block access to a buggy version altogether
  - Issue different certificates per version / device type
- Of course don’t forget to :
  - Remove any debugging code
  - Obfuscate the apk before release (proguard)
  - Construct passwords on the fly; use tokenization and runtime functions
  - Even better, construct passwords algorithmically from existing application strings
- Important for client key pairs : use the appropriate KeyStore
  - Do not use JKS / BKS; this is only resistant to tampering
  - Use UBER; this is resistant to both tampering AND inspection

Is this security by obscurity?

- Obscurity does not (usually) provide security
  - But even if you're specifically targeted, it **adds** to security
  - This "trick" adds to the setup, does nor replace something
  - If it gets broken, nothing changes in comparison to not having it at all
- Similar cases
  - Do you change your SSH server port?
  - Do you use port-knocking?
  - Do you hide your servers' version information from headers?
  - Do you remove software banners and welcome texts?

Do as Google says…

- Do you use Google Play and In-App billing?
- Google says: Protect your Google Play public key
  - To keep your public key safe from malicious users and hackers, do not embed it in any code as a literal string. Instead, **construct the string at runtime from pieces or use bit manipulation** (for example, XOR with some other string) to hide the actual key. **The key itself is not secret information**, but you do not want to make it easy for a hacker or malicious user to replace the public key with another key

Raise the bar

- There is no perfect security
  - But we should do our best, including delaying the attacker
- If someone targets you, you're in trouble
  - Big names are getting hacked every day
- Make sure you're not the easiest target
  - You avoid as much as possible amateurs / script kiddies
  - You buy yourself some time on the next 0-day vulnerability
- Make intrusion difficult and time consuming
  - A cost benefit analysis should push the attacker away

Remember that sometimes…

Drawbacks

- There is some inherent complexity
  - Complexity is bad for security
  - But you only do it once and the code is very simple
- You may open yourself to new attack vectors
  - SSL – based (although you already have SSL)
  - Potential DOS due to client SSL authentication
- Potential performance overhead

Next steps

- Wouldn’t it be nice
  - If that technique could be extended to generate user-bound certificates on registration?

Get in touch

Michalis Kamprianis

» On LinkedIn:
https://www.linkedin.com/in/michaliskamprianis

» By email:
michalis.kamprianis@gmail.com

» On Twitter:
@kamprianism