Harvesting Developer Credentials in Android Apps

Yajin Zhou, Lei Wu, Zhi Wang, Xuxian Jiang
North Carolina State University
Florida State University
Qihoo 360

Presenter: Yue Chen
Florida State University
Apps are Becoming Popular

- Initial release: 10/2008
- 675,000 apps
- 25 billion downloads
- 800,000 apps
- ~30 billion downloads

Timeline:
- 10/2008
- 09/2012
- 02/2013
Third-party Services Require Authentication
Protecting Developer Credentials is Hard

1  .method public static SendMailInBackground
2   new-instance v3, Lcom/pompeiicity/funpic/Email;
3   const-string v7, "whav*****@gmail.com"
4   const-string v8, "jea*****"
5   invoke-direct {v3, v7, v8}, Lcom/pompeiicity/funpic/Email;->
6       <init>(Ljava/lang/String;Ljava/lang/String;)V
7      ...
8  .end method
Credential Leak is Dangerous

"whav*****@gmail.com"
"jea*****"
CredMiner: Mine Credentials from Apps

- App Repo
- Select Candidate Apps
- Identify Data Sources
- Reconstruct Credentials
- Validate Credentials
Select Candidate Apps

• Apps that use interesting libraries (i.e., libraries that accept plaintext credentials)

    JavaMail Library

    Amazon AWS Library

    ……
Identify Data Sources

String _user, _passwd;

_user = "edcba"; _passwd = "54321";

String user = new StringBuilder(_user).reverse().toString();
String passwd = new StringBuilder(_passwd).reverse().toString();

System.out.println("Authenticating..." );

return new PasswordAuthentication(user, passwd);
Program Slicing Example

String _user, _passwd;

_user = “edcba”; _passwd = “54321”;

String user = new StringBuilder(_user).reverse().toString();
String passwd = new StringBuilder(_passwd).reverse().toString();

System.out.println(“Authenticating...”);

return new PasswordAuthentication(user, passwd);
String _user, _passwd;

_user = "edcba"; _passwd = "54321";

String user = new StringBuilder(_user).reverse().toString();
String passwd = new StringBuilder(_passwd).reverse().toString();

System.out.println("Authenticating...");

return new PasswordAuthentication(user, passwd);
String _user, _passwd;

_user = "edcba"; _passwd = "54321";

String user = new StringBuilder(_user).reverse().toString();
String passwd = new StringBuilder(_passwd).reverse().toString();

System.out.println("Authenticating... ");

return new PasswordAuthentication(user, passwd);
String _user, _passwd;

_user = "edcba"; _passwd = "54321";

String user = new StringBuilder(_user).reverse().toString();
String passwd = new StringBuilder(_passwd).reverse().toString();

System.out.println("Authenticating...");

return new PasswordAuthentication(user, passwd);
Program Slicing Example

String _user, _passwd;

_user = "edcba"; _passwd = "54321";

String user = new StringBuilder(_user).reverse().toString();
String passwd = new StringBuilder(_passwd).reverse().toString();

System.out.println("Authenticating...");

return new PasswordAuthentication(user, passwd);
Program Slicing Example

String _user, _passwd;

_user = “edcba”; _passwd = “54321”;

String user = new StringBuilder(_user).reverse().toString();
String passwd = new StringBuilder(_passwd).reverse().toString();

return new PasswordAuthentication(user, passwd);
Backward Slicing

- Dalvik byte-code is register-based.

Simple Example (backward):

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Tracked Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>v -&gt; Interesting Library</td>
<td>v</td>
</tr>
<tr>
<td>move v2, v</td>
<td>v2</td>
</tr>
<tr>
<td>move v3,v4</td>
<td>(Ignored)</td>
</tr>
<tr>
<td>const-string v2, &quot;abcde&quot;</td>
<td>(Done)</td>
</tr>
</tbody>
</table>

Generated Program Slice:

- const-string v2, "abcde"
- move v2,v
- v -> Interesting Library
Reconstruct Credentials

- Use an execution engine (in Python) to execute (forward) the program slice.
- Create mock objects on demand, to run the program slice.
Reconstruct Credentials

Emulated `java.lang.StringBuilder.append(char):`

```python
def StringBuilder_append_C_Ljava_lang_StringBuilder(arg_regs):
    #get the register value. Register names are in the
    #parameter arg_regs, v1, v2 for instance
    arg1_value = vm.get_reg_value(arg_regs.split("","")[0].strip()()
    arg2_value = vm.get_reg_value(arg_regs.split("","")[1].strip()()
    #check the type
    if type(arg1_value) != str or type(arg2_value) != str:
        print ("[x] [%d] we expect an (string, string) here. But"
        "the actual type is (%s,%s)" % (get_linenumber(),
        str(type(arg1_value)), str(type(arg2_value))))
        return False
    #emulate java.lang.StringBuilder.append(char)
    arg1_value += arg2_value
    vm.put_reg_value(arg1, arg1_value)
    return True
```
Validate Credentials

• Run the app in an Android emulator and monitor its execution.
• Compare the run-time parameters to those recovered by CredMiner.
• Monitor the interaction with remote servers.
## Evaluation

**Distribution of Collected Apps:**

<table>
<thead>
<tr>
<th></th>
<th>Google Play</th>
<th>Alternative Markets</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Apps</td>
<td>21,092</td>
<td>15,479</td>
</tr>
<tr>
<td>Percentage</td>
<td>57.67%</td>
<td>42.33%</td>
</tr>
<tr>
<td># of Total Apps</td>
<td>36,571</td>
<td></td>
</tr>
<tr>
<td># of Distinct Apps</td>
<td>36,561</td>
<td></td>
</tr>
</tbody>
</table>
Evaluation

Overall Result:

- 237 candidate apps use the JavaMail library.
- 196 candidate apps use the Amazon AWS SDK.
- 51.1% (121/237) and 67.3% (132/196) of these candidate apps are vulnerable.

Distribution of Vulnerable Apps:

<table>
<thead>
<tr>
<th>Category of Credentials</th>
<th>Google Play</th>
<th>Other Stores</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email Credentials</td>
<td>65</td>
<td>56</td>
<td>121</td>
</tr>
<tr>
<td>Amazon AWS Credentials</td>
<td>128</td>
<td>4</td>
<td>132</td>
</tr>
</tbody>
</table>
Evaluation – Email Credentials

• 121/237 (51.1%) are vulnerable

• 11 apps having more than 50,000 downloads.

• 8/1404 malware are vulnerable!
• Only 2 of these malware protect their accounts with custom string encoding schemes
Evaluation – Email Credentials

Case Study:
• A popular app with more than 1,000,000 downloads.
• JavaMail is used to send pin recovery email.
• Email credential is obfuscated with AES.
• The encrypted credential is encoded again in Base64.
• The key to decrypt is only encoded in Base64.
• the initial vector for AES is simply a constant string.
• The decrypted password is abc123**.
Evaluation – Email Credentials

- Leaked credentials: Total vs. Valid
Evaluation – Amazon AWS Credentials

Background:
Permanent credential?

(Anonymous) token vending machine (TVM):
Evaluation – Amazon AWS Credentials

Problem:
• Mis-configure the TVM servers.
• Fail to constrain the privilege of the temporary credential.

Result:
• 132/196 (67.3%) are vulnerable
• 24% have more than 50,000 downloads

Case study:
• One app from Google Play, has more than 5,000,000 downloads.
• The temporary credential is not properly confined, the attacker is able to enumerate the UIDs and access other app users’ files. (We created two accounts and conducted this experiment.)
Takeaway

• Never embed developers’ credentials in the app.

• Use secure solutions from service providers **correctly**.

  > Insecure sample code provided by service providers can mislead developers.
  
  – **Use secure samples!**

  > There are no clear instructions about how to use the SDKs securely.
  
  – **Simple, clean, and secure documents!**
  – **A secure default configuration of the services!**
Thank you!

Q&A
Backup Slides
Program Slicing Example

```c
int i;
int sum = 0;
int product = 1;
for(i = 1; i < N; ++i) {
    sum = sum + i;
    product = product * i;
}
write(sum);
write(product);
```
Program Slicing Example

```c
int i;
int sum = 0;
int product = 1;
for(i = 1; i < N; ++i) {
    sum = sum + i;
    product = product * i;
}
write(sum);
write(product);
```
int i;
int sum = 0;

for(i = 1; i < N; ++i) {
    sum = sum + i;
}

write(sum);
Identify Data Sources

Username: abcde
Password: 123456

Interesting Library
Identify Data Sources

Username: edcba
Password: 654321

Reverse() → Interesting Library
Identify Data Sources

Username: edcba
Password: 654321

Reverse()

Interesting Library

1. Locating Sink Methods
2. Backtracking Credentials
3. Finding Source Methods