```
line : 24 import android.widget.Button;
line : 25 import android.widget.EditText;
line : 26 import android.widget.ImageView;
line : 56
line : 57 public class LoginActivity extends AppCompatActivity {
line : 58 private static final String AUTHORITY = "https://login.microsoftonline.com/common";
line : 132 this.mConstraintLayout = findViewById(R.id.login_page_parent_layout);
line : 133 this.userNameEditText = (EditText) findViewById(R.id.editTextTextEmailAddress);
line : 134 this.passwordEditText = (EditText) findViewById(R.id.editTextTextPassword);
line : 135 this.loginButton = (Button) findViewById(R.id.login_screen_button);
```

We have detected '(EditText)' in the file workplaceoptions/com/iconnectyou/view/ui/ForgotPasswordSecondScreen.java

```
line : 11 import android.widget.Button;
line : 12 import android.widget.EditText;
line : 13 import android.widget.ImageView;
line : 28
line : 29 public class ForgotPasswordSecondScreen extends AppCompatActivity {
line : 30 private String email;
line : 56 this.forgotConfirmTextView = (Button) findViewById(R.id.forgot_second_send_button);
line : 57 this.passcodeFirstEditText = (EditText) findViewById(R.id.forgot_passcode_one_et);
line : 58 this.passcodeSecondEditText = (EditText) findViewById(R.id.forgot_passcode_two_et);
line : 59 this.passcodeThirdEditText = (EditText) findViewById(R.id.forgot_passcode_three_et);
line : 60 this.passcodeFourthEditText = (EditText) findViewById(R.id.forgot_passcode_four_et);
line : 61 this.mProgressBar = (ProgressBar) findViewById(R.id.common_progress);
```

We have detected '(EditText)' in the file workplaceoptions/com/iconnectyou/view/ui/ForgotPasswordFirstScreen.java

```
line : 11 import android.widget.Button;
line : 12 import android.widget.EditText;
line : 13 import android.widget.ImageView;
line : 28
line : 29 public class ForgotPasswordFirstScreen extends AppCompatActivity {
line : 30 private ConstraintLayout emailIDCL;
line : 53 this.sendPhoneEmailButton = (Button) findViewById(R.id.forgot_first_send_button);
line : 54 this.forgotEmailPhoneEditText = (EditText) findViewById(R.id.forgot_email_phone_et);
line : 55 this.mProgressBar = (ProgressBar) findViewById(R.id.common_progress);
```

We have detected '(EditText)' in the file workplaceoptions/com/iconnectyou/view/fragments/dialogfragments/ProviderStateDialogFragment.java

```
line : 8 import android.view.ViewGroup;
line : 9 import android.widget.EditText;
line : 10 import android.widget.ImageView;
line : 27
line : 28 public class ProviderStateDialogFragment extends DialogFragment {
line : 29 ConstraintLayout backgroundLayout;
line : 162 private void disableCopyPaste() {
line : 163 Utils.disableCopyPasteOperations((EditText) this.searchView.findViewById(2131297254));
line : 164 }
```

We have detected '(EditText)' in the file workplaceoptions/com/iconnectyou/view/fragments/dialogfragments/ProviderCountryDialogFragment.java

```
line : 8 import android.view.ViewGroup;
line : 9 import android.widget.EditText;
line : 10 import android.widget.ImageView;
line : 28
line : 29 public class ProviderCountryDialogFragment extends DialogFragment {
line : 30 ConstraintLayout backgroundLayout;
line : 180 private void disableCopyPaste() {
line : 181 Utils.disableCopyPasteOperations((EditText) this.searchView.findViewById(2131297254));
line : 182 }
```

Severity
Medium

VULNERABILITY

Weak Random Number Generator

OWASP MASVS

3.6 [L1, L2]

Common Weakness Enumeration

CWE-1241

Known Exploits

Multiple vulnerabilities

Common Vulnerability Scoring System

Best Practices:

https://github.com/OWASP/owasp-mstg/blob/1.1.3/Document/0x05e-Testing-Cryptography.md#testing-random-number-generation-mstg-crypto-6

Reference URL[s]:

https://developer.android.com/reference/java/util/Random.html
https://developer.android.com/reference/java/security/SecureRandom.html
https://arstechnica.com/information-technology/2013/08/google-confirms-critical-android-crypto-flaw-used-in-5700-bitcoin-heist/

CVSS BaseScore and Vector

Multiple vulnerabilities

THREAT

Developers generally implement random number generators [RNGs] where the random number is fully determined by the seed knowledge. This is the reason why they are called pseudo-random number generators [PRNGs]

When it comes to cryptography, random numbers play a fundamental role in: key generation nonces one-time pads salts in certain signature schemes

RISK

Using standard PRNGs is a bad practice when implementing security mechanisms, since the attacker may be able to guess the logic behind and predict the generated random numbers. In this case the confidentiality and/or integrity of the vulnerable app might be undermined

FIX

Don't use standard random() method or Pseudo-Random Number Generators [PRNGs] because they will NOT really return non-predictable random numbers

Use cryptographically secure pseudorandom number generators [CSPRN G] such a directly provided by Google. CSPRNG are able to pass the "next-bit" test and to hold up well under serious attack, even when part of their initial or running state becomes available to an attacker

We have detected ' Random()' in the file org/jivesoftware/smack/ReconnectionManager.java

```
line : 4 import java.util.Map;
line : 5 import java.util.Random;
line : 6 import java.util.Set;
line : 23 private final Set<ReconnectionListener> reconnectionListeners = new CopyOnWriteArraySet();
line : 24 private final int randomBase = new Random().nextInt(13) + 2;
line : 25 private volatile int fixedDelay = defaultFixedDelay;
```

We have detected ' Random()' in the file org/jivesoftware/smack/util/RandomUtil.java

```
line : 3 import java.security.SecureRandom;
line : 4 import java.util.Random;
line : 5
line : 17 public Random initialValue() {
line : 18 return new Random();
line : 19 }
```

We have detected ' Random()' in the file org/jivesoftware/smackx/filetransfer/FileTransferNegotiator.java

```
line : 8 import java.util.Map;
line : 9 import java.util.Random;
line : 10 import java.util.WeakHashMap;
line : 37 private static final Map<XMPPConnection, FileTransferNegotiator> INSTANCES = new WeakHashMap();
line : 38 private static final Random randomGenerator = new Random();
line : 39
```

We have detected ' Random()' in the file org/bouncycastle/asn1/ua/DSTU4145PointEncoder.java

```
line : 3 import java.math.BigInteger;
line : 4 import java.util.Random;
line : 5 import org.bouncycastle.math.ec.ECConstants;
line : 55 ECFieldElement fromBigInteger = eCCurve.fromBigInteger(ECConstants.ZERO);
line : 56 Random random = new Random();
line : 57 int fieldSize = eCFieldElement.getFieldSize();
```

We have detected ' Random()' in the file org/bouncycastle/pqc/math/linearalgebra/GF2Polynomial.java

```
line : 3 import java.math.BigInteger;
line : 4 import java.util.Random;
line : 5 import kotlin.UByte;
line : 16 private int[] value;
line : 17 private static Random rand = new Random();
line : 18 private static final boolean[] parity = {false, true, true, false, true, false, false, true, true, false, false, true, false, true, true, false, true, fals
e, false, true, false, true, true, false, false, true, true, false, true, false, false, true, true, false, false, true, false, true, true, false, false, true, true, fa
lse, true, false, false, true, false, true, true, false, true, false, false, true, true, false, false, true, false, true, true, false, true, false, false, true, false,
true, true, false, false, true, true, false, true, false, false, true, false, true, true, false, true, false, false, true, true, false, false, true, false, true, true,
false, false, true, true, false, true, false, false, true, true, false, false, true, false, true, true, false, true, false, false, true, false, true, true, false, fals
e, true, true, false, true, false, false, true, true, false, false, true, false, true, true, false, false, true, true, false, true, false, false, true, false, true, tr
ue, false, true, false, false, true, true, false, false, true, false, true, true, false, false, true, true, false, true, false, false, true, true, false, false, true,
false, true, true, false, true, false, false, true, false, true, true, false, false, true, true, false, true, false, false, true, false, true, true, false, true, false
, false, true, true, false, false, true, false, true, true, false, true, false, false, true, false, true, true, false, false, true, true, false, true, false, false, tru
e, true, false, false, true, false, true, true, false, false, true, true, false, true, false, false, true, false, true, true, false, true, false, false, true, true, fa
lse, false, true, false, true, true, false};
```

We have detected ' Random()' in the file org/bouncycastle/pqc/math/linearalgebra/GF2nONBField.java

```
line : 4 import java.security.SecureRandom;
line : 5 import java.util.Random;
line : 6 import java.util.Vector;
line : 153 int order;
line : 154 Random random = new Random();
line : 155 int i3 = 0;
```

We have detected ' Random()' in the file org/bouncycastle/pqc/math/linearalgebra/GF2nPolynomialElement.java

```
line : 3 import java.math.BigInteger;
line : 4 import java.util.Random;
line : 5 import okhttp3.internal.http2.Http2Connection;
line : 370 do {
line : 371 GF2nPolynomialElement gF2nPolynomialElement2 = new GF2nPolynomialElement((GF2nPolynomialField) this.mField, new Random());
line : 372 ZERO = ZERO((GF2nPolynomialField) this.mField);
```

We have detected ' Random()' in the file org/bouncycastle/math/ec/ECCurve.java

```
line : 4 import java.util.Hashtable;
line : 5 import java.util.Random;
line : 6 import org.bouncycastle.math.ec.ECFieldElement;
line : 142 ECFieldElement fromBigInteger = fromBigInteger(ECConstants.ZERO);
line : 143 Random random = new Random();
line : 144 do {
```

We have detected ' Random()' in the file org/bouncycastle/math/ec/ECFieldElement.java

```
line : 3 import java.math.BigInteger;
line : 4 import java.util.Random;
line : 5 import org.bouncycastle.math.raw.Mod;
line : 556 BigInteger subtract = this.q.subtract(ECConstants.ONE);
line : 557 Random random = new Random();
line : 558 while (true) {
```

We have detected ' Random()' in the file org/bouncycastle/math/raw/Mod.java

```
line : 2
line : 3 import java.util.Random;
line : 4
line : 103 int length = iArr.length;
line : 104 Random random = new Random();
line : 105 int[] create = Nat.create(length);
```

We have detected ' Random()' in the file org/minidns/AbstractDnsClient.java

```
line : 8 import java.util.HashSet;
line : 9 import java.util.Random;
line : 10 import java.util.Set;
line : 90 };
line : 91 this.insecureRandom = new Random();
line : 92 this.dataSource = new NetworkDataSource();
```

Severity
Medium

VULNERABILITY

Weak Hashing Algorithms

OWASP MASVS	**Common Weakness Enumeration**
3.4 [L1, L2]	CWE-326
Known Exploits	**Common Vulnerability Scoring System**
CVE-2018-14992	CVE-2018-14992-CVSS 3.0 Score 5.5
CVE-2017-15999	CVE-2017-15999-CVSS 3.0 Score 9.8

Best Practices:

https://github.com/OWASP/owasp-mstg/blob/1.1.3/Document/0x04g-Testing-Cryptography.md#identifying-insecure-andor-deprecated-cryptographic-algorithms-mstg-crypto-4

Reference URL[s]:

https://developer.android.com/reference/java/security/MessageDigest.html
https://valerieaurora.org/hash.html
https://www.computerworld.com/article/3173616/the-sha1-hash-function-is-now-completely-unsafe.html

CVSS BaseScore and Vector

CVE-2018-14992
Version & Base Score : CVSS 3.0 Score 5.5
CVSS Scoring Vector : CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:H/A:N

CVE-2017-15999
Version & Base Score : CVSS 3.0 Score 9.8
CVSS Scoring Vector :
CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

THREAT

The mobile application leverages weak hashing algorithms. Weak hashing algorithms [e.g. MD2, MD4, MD5, SHA-0 or SHA-1] can be vulnerable to hash collisions, so they should not be used when reliable data hashing is required

RISK

Hashing algorithms are widely used to validate credit card transactions, electronic documents, email PGP/GPG signatures, open-source software repositories, backups and software updates

If a weakness is found in a hash function that allows for two files to have the same digest, the function is considered cryptographically broken, because digital fingerprints generated with it can be forged and cannot be trusted. Attackers could, for example, create a rogue software update that would be accepted and executed by an update mechanism that validates updates by checking digital signatures

FIX

Please be sure to put in place a strong hashing algorithm, like SHA-2 [the SHA-256 function is a very popular choice] or - better - SHA-3

We have detected 'MessageDigest.getInstance("SHA-1")' in the file org/jivesoftware/smackx/caps/EntityCapsManager.java

```
line : 78 try {
line : 79 hashMap.put("SHA-1", MessageDigest.getInstance("SHA-1"));
line : 80 } catch (NoSuchAlgorithmException unused) {
```

Severity
Low

VULNERABILITY

App allowed to run in an emulator

OWASP MASVS

8.5 R

Common Weakness Enumeration

-

Known Exploits

-

Common Vulnerability Scoring System

-

Best Practices:

https://github.com/OWASP/owasp-mstg/blob/1.1.3/Document/0x05j-Testing-Resiliency-Against-Reverse-Engineering.md#testing-emulator-detection-mstg-resilience-5
https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05c-Reverse-Engineering-and-Tampering.md

Reference URL[s]:

https://developer.android.com/tools/help/emulator.html

CVSS BaseScore and Vector

-

THREAT

An emulator activity outside the development process is a good indication that someone other than you is trying to analyse the app

RISK

When an app runs in an emulator an attacker can dynamically analyse it, access its sensitive data and steal its intellectual property

FIX