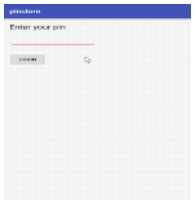
PIN STORE

Tuesday, July 30, 2019 9:40 AM



I have no any idea to get secret . So I decompile the apk and also convert to reabale java code by dex2jar.
Now I opened the code in jd-gui:

Taking user input and storing in variable **str2**

~: I check onclick method , what operation Is perfomed on clicking on login button:

```
public void onClick(View paramAnonymousView)
{
  String str2 = MainActivity.this.pinEditText.getText().toString();
  paramAnonymousView = null;
  Object localObject = null;
  try
  {
    str1 = new DatabaseUtilities(MainActivity.this.getApplicationContext()).fetchPin();
    paramAnonymousView = str1;
```

From DatabaseUtilities class fetchpin method is called and ouput stored in **str1**

Assing str1 value to paramAonymousview

```
public String fetchPin()
  throws IOException
{
  openDB();
  Cursor localCursor = this.db.rawQuery("SELECT pin FROM pinDB", null);
  String str = "";
  if (localCursor.moveToFirst()) {
    str = localCursor.getString(0);
  }
  localCursor.close();
  return str;
}
```

fetchpin simple fetch pin from pinDB database

AFTER FETCHING PIN AND ASSIGING STR1 VALUE TO paramAnonymous

```
try
{
  str1 = CryptoUtilities.getHash(str2);
  localObject = str1;
```

From cryptoUtilities class getHash method is called using str2 and store n variable str1
And its value assign to localobject.

```
public static String getHash(String paramString)
  throws NoSuchAlgorithmException, UnsupportedEncodingException
{
  byte[] arrayOfByte = paramString.getBytes();
  paramString = null;
  try
  {
    MessageDigest localMessageDigest = MessageDigest.getInstance("SHA-1");
    paramString = localMessageDigest;
  }
```

From this method it is clear , it return sha-1 hash

AFTER GETHASH FUNCTION CALLED

Comparing paramAnonymousView (sha-1 hash) with (sha-1 hash of str2)

```
}
if (paramAnonymousView.equalsIgnoreCase((String)localObject))
{
  paramAnonymousView = new Intent(MainActivity.this, SecretDisplay.class);
  paramAnonymousView.putExtra("pin", str2);
  MainActivity.this.startActivity(paramAnonymousView);
```

If equal intent is called **secretdisplay** is called

```
paramAnonymousView = new Intent(MainActivity.this, SecretDisplay.class);
paramAnonymousView.putExtra("pin", str2);
MainActivity.this.startActivity(paramAnonymousView);
```

If equal intent is called **secretdisplay** is called

```
public class SecretDisplay
  extends AppCompatActivity
{
  protected void onCreate(Bundle paramBundle)
  {
    super.onCreate(paramBundle);
    setContentView(2130968602);
    paramBundle = getApplicationContext();
    TextView localTextView = (TextView)findViewById(2131492951);
    String str = getIntent().getStringExtra("pin");
    try
    {
      DatabaseUtilities localDatabaseUtilities = new DatabaseUtilities(getApplicationContext());
      localTextView.setText(new CryptoUtilities("v1", str).decrypt(localDatabaseUtilities.fetchSecret()));
      Toast.makeText(paramBundle, str, 1);
      return;
```

From CryptoUtilities fetchSecret function is called and also decrypt method is called.

```
public String decrypt(String paramString)
  throws Exception
{
  paramString = Base64.decode(paramString.getBytes(), 2);
  Log.d("Status", paramString.toString());
  this.cipher.init(2, this.key);
  return new String(this.cipher.doFinal(paramString), "UTF-8");
}
```

```
public SecretKeySpec getKey(String paramString)
  throws Exception
{
  if (paramString.equalsIgnoreCase("v1"))
  {
    Log.d("Version", paramString);
    paramString = "t0ps3kr3tk3y".getBytes("UTF-8");
    return new SecretKeySpec(Arrays.copyOf(MessageDigest.getInstance("SHA-1").digest(paramString), 16), "AES");
  }
  Log.d("Version", paramString);
  paramString = "SampleSalt".getBytes();
  char[] arrayOfChar = this.pin.toCharArray();
  return new SecretKeySpec(SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1").generateSecret(new PBEKeySpec(arrayOfChar, paramStr
  }
}
```

It is clear that using this code secret is decreypted. Now use this code to decrypt the secret

a. First go to application package and their database and get pin :

```
sqlite> select * from pinDB;
1|d8531a519b3d4dfebece0259f90b466a23efc57b
sqlite>
```

From above code it is clear pin is sha-1 hashed so decrypt this using sha-1:

Pin: 7498

b. Now fetch encrypted secret from database:

```
sqlite> select * from secretsDBv2;
1|Bi528nDlNBcX9BcCC+ZqGQo1Oz8i+GOWSmvxRj7jg1g=
```

Put the secret in code to decrypt this. Code Is given below:

```
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
```

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import java.util.Arrays;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;
import java.util.*;


public class PINSTORE {

	public static void main(String[] args) throws InvalidKeySpecException, NoSuchAlgorithmException,
	NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException, BadPaddingException {
		// TODO Auto-generated method stub
		String SECRET = "Bi528nDlNBcX9BcCC+ZqGQo1Oz01+GOWSmvxRj7jg1g=";

		  Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
		  byte[] salt = "SampleSalt".getBytes();
		  String pin = "7498";
		  SecretKeySpec key = new
		SecretKeySpec(SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1").generateSecret(new
		PBEKeySpec(pin.toCharArray(), salt, 1000, 128)).getEncoded(), "AES");
		  cipher.init(2,key);
		  System.out.println(new String(cipher.doFinal(Base64.getDecoder().decode(SECRET))));
}


}
```