모바일 앱 취약점 분석 기초

[19반] 정민석_7000

요약

과제의 목표는 https://mobilehacking.kr의 SolidApp 문제를 해결하는 것입니다. com.ctf.solidapp.ui.login.LoginActivity, AuthLogic 클래스를 통하여 해답을 찾았습니다. 입력한 id가 decodeBase64("cmVkdGVhbQ==")이며, pw가 gttcurct이라는 각각의 문자를 6이라는 key로 XOR연산을 실행한 결과와 같으면 로그인에 성공합니다. 즉, ID: redteam, PW: arrester로 로그인에 성공하였습니다.

flag{redteam_arrester}

풀이

해당 앱은 기능 테스트를 위해 일부 기능만 포함된 버전이기 때문에, MainActivity가 아닌 다른 파일을 참조해야했습니다. 이때 AndroidManifest.xml의 com.ctf.solidapp.ui.login.LoginActivity에 주목하였습니다.

해당 action은 AuthLogic이라는 클래스를 통하여 Username, Password의 유효성 여부를 판단합니다.

```
// 생략
/* JADX INFO: Access modifiers changed from: private */
public static final void onCreate$lambda$5$lambda$4{ProgressBar loading, LoginActivity this$0, EditText username, EditText password, View view) {
       Intrinsics.checkNotNullParameter(loading, "$loading");
       Intrinsics.checkNotNullParameter(this$0, "this$0");
       Intrinsics.checkNotNullParameter(username, "$username");
      Intrinsics.checkNotNullParameter(password, "$password");
       loading.setVisibility(0);
       LoginViewModel loginViewModel = this$0.loginViewModel;
       if (loginViewModel == null) {
             Intrinsics.throwUninitializedPropertyAccessException("loginViewModel");
             loginViewModel = null;
       loginViewModel.login(username.getText().toString(), password.getText().toString());\\
       AuthLogic authLogic = new AuthLogic();
       if (auth Logic. is Password Valid (password.get Text(). to String()) \& auth Logic. is Username Valid (username.get Text(). to String())) \\ \{ (auth Logic. is Username Valid (username.get Text(). to String())) \\ \{ (auth Logic. is Username Valid (username.get Text(). to String())) \\ \{ (auth Logic. is Username Valid (username.get Text(). to String())) \\ \{ (auth Logic. is Username Valid (username.get Text(). to String())) \\ \{ (auth Logic. is Username Valid (username.get Text(). to String())) \\ \{ (auth Logic. is Username Valid (username.get Text(). to String())) \\ \{ (auth Logic. is Username Valid (username.get Text(). to String())) \\ \{ (auth Logic. is Username Valid (username.get Text(). to String())) \\ \{ (auth Logic. is Username Valid (username.get Text(). to String())) \\ \{ (auth Logic. is Username Valid (username.get Text(). to String())) \\ \{ (auth Logic. is Username Valid (username.get Text())) \\ \{ (auth Logic. is Username Valid (username.get Text())) \\ \{ (auth Logic. is Username Valid (username.get Text())) \\ \{ (auth Logic. is Username Valid (username.get Text())) \\ \{ (auth Logic. is Username Valid (username.get Text())) \\ \{ (auth Logic. is Username Valid (username.get Text())) \\ \{ (auth Logic. is Username.get Text()) \\ \{ (auth Logic. is Username Valid (username.get Text())) \\ \{ (auth Logic. is Username Valid (username.get Text())) \\ \{ (auth Logic. is Username Valid (username.get Text())) \\ \{ (auth Logic. is Username Valid (username.get Text())) \\ \{ (auth Logic. is Username Valid (username.get Text())) \\ \{ (auth Logic. is Username Valid (username.get Text())) \\ \{ (auth Logic. is Username Valid (username.get Text())) \\ \{ (auth Logic. is Username.get Text()) \\ \{ (auth Logic. is Username.get 
             Intent intent = new Intent(this$0, (Class<?>) ItemDetailHostActivity.class);
             Toast.makeText(this$0.getApplicationContext(), "Login Success!", 0).show();
             this$0.startActivity(intent);
      } else {
             Toast.makeText(this$0.getApplicationContext(), "Login Fail!", 0).show();
             this$0.finish();
// 생략
```

이에 AuthLogic 클래스의 소스코드를 살펴보았습니다. 해당 코드를 분석한 결과 decodeBase64("cmVkdGVhbQ==")과 입력한 ID가 동일하고, gttcurct이라는 각각의 문자를 6이라는 key로 XOR연산을 실행한 결과가 입력한 PW와 같으면 로그인에 성공합니다.

```
public final class AuthLogic {
    private final String decodedUsername = decodeBase64("cmVkdGVhbQ==");
    private final String staticString = "gttcurct";
    private final int key = 6;

private final String decodeBase64(String encoded) {
    byte[] decodedBytes = Base64.decode(encoded, 0);
    Intrinsics.checkNotNullExpressionValue(decodedBytes, "decodedBytes");
    return new String(decodedBytes, Charsets.UTF_8);
}
```

모바일 앱 취약점 분석 기초

```
private final String xorString(String input, int key) {
   StringBuilder sb = new StringBuilder();
   int length = input.length();
   for (int i = 0; i < length; i++) {
        sb.append((char) (input.charAt(i) ^ key));
   }
   String sb2 = sb.toString();
   Intrinsics.checkNotNullExpressionValue(sb2, "xorResult.toString()");
   return sb2;
}

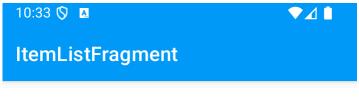
public final boolean isUsernameValid(String inputUsername) {
   Intrinsics.checkNotNullParameter(inputUsername, "inputUsername");
   return Intrinsics.areEqual(inputUsername, this.decodedUsername);
}

public final boolean isPasswordValid(String inputPassword) {
   Intrinsics.checkNotNullParameter(inputPassword, "inputPassword");
   return Intrinsics.areEqual(inputPassword, "inputPassword");
   return Intrinsics.areEqual(inputPassword, xorString(this.staticString, this.key));
}</pre>
```

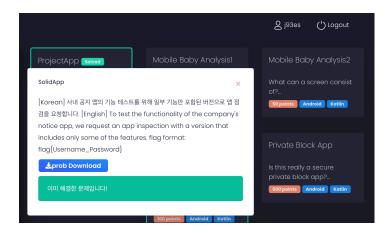
결과

ID: redteam, PW: arrester 를 확인할 수 있으며, 로그인에 성공하였습니다.

모바일 앱 취약점 분석 기초



- 1 Item 1
- 2 Item 2
- 3 Item 3
- 4 Item 4
- 5 Item 5
- 6 Item 6
- 7 Item 7
- 8 Item 8
- 9 Item 9
- 10 Item 10



모바일 앱 취약점 분석 기초 3