1.1.1 Yapılandırma Dosyasında Açık Metin Halinde Parola Bulundurulması (Password in Configuration File) (CWE-260) (CWE-313)

Açıklık Önem Derecesi: Düşük

Açıklığın Etkisi: Hassas Bilgilere Yetkisiz Erişim, Bilgi İfşası, Sızma girişimlerinde saldırının boyutunun artması

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Uygulama geliştiricileri uygulamalarını daha modüler ve güvenli hale getirebilmek için parola gibi hassas verileri yapılandırma dosyalarına koyabilmektedirler. Yapılandırma dosyalarındaki bu hassas veriler uygulama tarafından ihtiyaç duyulan noktalarda kullanılırlar. Güvenliğe derinlikli defans (depth of defence) perspektifinden yaklaşacak olursak yapılandırma dosyalarında bu tarz hassas verilerin açık metin halinde (yani şifrelenmeden) yer alması güvenlik riski teşkil etmektedir.

Parola v.b. hassas veriler uygulamanın bir yapılandırma dosyasında (veya diskte bir dosyada) açık metin halinde depolanmamalıdır. Çünkü uygulamanın ilgili dosyasına erişimi olan / sızan veya depolama ortamına fiziksel ya da yönetici erişimi olan bir saldırgan girdiği sistemde bu hassas verileri okuyabilir. Bu ise saldırganın girdiği sistemde yapabileceği zararın ölçüsünü arttırabilir. Dolayısıyla uygulamanın yapılandırma dosyasında (veya diskte bir dosyada) açık metin halinde hassas veri depolama bir güvenlik açığı olarak ele alınır. Saldırganların girdikleri sistemde vereceği zararı minimize etmek için hassas veriler açık metin halinde depolanmamalıdır.

Uygulama güvenliğinde temel esas saldırganların içeriye hiçbir şekilde girememesidir. Fakat bazen güvenlik ne kadar üst seviyede olursa olsun saldırganın içeri girmesi devasa bir yüzeyde sadece bir boşluk yakalamasına bakabilmektedir. Dolayısıyla güvenliği daha garanti hale getirmenin yolu güvenliğe katman stratejisiyle yaklaşmaktan geçer. Bu ise örneğin uygulamanıza veya ağınıza gelecekte olası bir sızma girişimi yapıldığında şayet girişim başarılı olursa saldırganın içerideyken verebileceği zararı minimize edecek, belki de tam manasıyla sıfırlayacak iç katmanınızdaki güvenliğinizle mümkündür. Uygulama sunucunuza sızıldığı senaryoyu ele alacak olursak saldırgan örneğin uygulama yapılandırma dosyasını okuyarak ldap kullanıcı adı ve şifre bilgilerini veya smtp kullanıcı adı ve şifre bilgilerini, v.b.

ele geçirebilir. Bu sayede belki sadece web sunucu içerisinde uygulamanın kök dizininde sıkışıp kalacakken şimdi edindiği bilgi doğrultusunda uygulamanın bulunduğu domain ağına atlama yapabilir veya eposta sunucusuna erişebilir. Uygulama üzerinden yetki ölçüsünde bu şekilde daha da ilerleyebilir ve uzanabildiği ölçüde atlamalar yapabilir.

Uygulama parolaları açık metin olarak yapılandırma dosyalarında (veya diskteki bir dosyada) yer aldığında "Password in Configuration File (CWE-260)" veya daha genel tabirle "Cleartext Storage in a File or on Disk (CWE-313)" açıklığı olarak işaretlenirler. Uygulamalardaki bu açıklığa şöyle bir örnek verilebilir:

Java: // .properties yapılandırma dosyası

```
…
webapp.ldap.username = ismail.hakki
webapp.ldap.password = Qwerty123*
…
```

Burada java yapılandırma dosyasında uygulamanın ldap bağlantılarında kullanılmak üzere hesap bilgilerine yer verildiği görülmektedir. Bu veriler açık metin halinde yer aldığından güvensizdirler.

ASP.NET: // web.config yapılandırma dosyası

```
…
<connectionStrings>
<add name="ud_DEV" connectionString="connectDB=uDB; uid=admin; pwd=password;
dbalias=uDB;" providerName="System.Data.Odbc" />
</connectionStrings>
…
```

Bu örnekte ise asp.net yapılandırma dosyasında uygulamanın veritabanı bağlantılarında kullanılmak üzere veritabanı bağlantı cümleciğinin yer aldığı görülmektedir. Bu cümlecikteki hesap bilgileri verileri açık metin halinde yer aldığından güvensizdirler.

Kurum uygulama yapılandırma dosyasının açık metin halinde parola bilgisi barındırdığı tespit edilmiştir.

::::::: BULGU :::::::

Açıklığın Önlemi:

Parolalar yapılandırma dosyalarında şifreli halde tutulmalıdırlar. Şifrelemeyi çözmek için bir deşifreleme anahtarı kullanılmalıdır. Uygulama kaynak kodları bu yapılandırma dosyalarından parolaları deşifreleme ile çekerek işlemlerini yürütmelidir.

Referanslar:

1. https://cwe.mitre.org/data/definitions/260.html
2. https://cwe.mitre.org/data/definitions/313.html
3. https://www.owasp.org/index.php/Preventing_LDAP_Injection_in_Java
4. https://www.webguvenligi.org/wp-content/uploads/2007/11/authenticationtrk.pdf
5. https://www.owasp.org/index.php/Use_of_hard-coded_password
6. https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/connection-strings-and-configuration-files
7. https://stackoverflow.com/questions/4227921/how-can-we-have-two-connection-strings-in-web-config-and-switch-betweeen-them-in
8. https://stackoverflow.com/questions/5642474/setting-up-connection-string-in-asp-net-to-sql-server
9. https://www.c-sharpcorner.com/UploadFile/7d3362/various-ways-to-specify-connection-string-in-Asp-Net-web-app/
10. https://www.youtube.com/watch?v=v1EWUAYNA1g
11. https://www.youtube.com/watch?v=Gf3kFBAovXw
12. https://stackoverflow.com/questions/15365210/failed-to-encrypt-the-section-connectionstrings-using-provider-rsaprotectedco
13. https://superuser.com/questions/401495/equivalent-of-unix-find-command-on-windows
14. https://docs.microsoft.com/en-us/previous-versions/aspnet/hh8x3tas(v%3Dvs.100)
15. https://www.youtube.com/watch?v=lzm_uTW9Ug8
16. http://www.sortedset.com/how-to-encrypt-decrypt-a-password-stored-in-a-properties-file-with-java-jasypt-apache-commons-configuration/
17. https://github.com/sortedset/jasyptblog
18. https://commons.apache.org/proper/commons-configuration/userguide/upgradeto2_0.html
19. https://commons.apache.org/proper/commons-configuration/apidocs/org/apache/commons/configuration2/ex/ConfigurationException.html
20. https://commons.apache.org/proper/commons-configuration/apidocs/org/apache/commons/configuration2/PropertiesConfiguration.html
21. https://www.programcreek.com/java-api-examples/?class=org.apache.commons.configuration2.builder.FileBasedConfigurationBuilder&method=save
22. https://stackoverflow.com/questions/22541455/warning-the-method-assertequals-from-the-type-assert-is-deprecated
23. https://www.tutorialspoint.com/junit/junit_suite_test.htm
24. http://www.jasypt.org/dependencies.html
25. https://www.youtube.com/watch?v=2xA82u3Q84M
26. http://docs.vizrt.com/viz-content-pilot-guide/5.6/configuration_database_service_names_and_sid.html
27. https://docs.oracle.com/database/121/ODPNT/featConnecting.htm#ODPNT199
28. https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/oracle-and-adonet
29. https://www.oreilly.com/library/view/adonet-cookbook/0596004397/ch01s09.html

30. https://docs.microsoft.com/tr-tr/dotnet/framework/data/adonet/connection-string-syntax#sql-server-authentication-with-sqlclient
31. https://www.c-sharpcorner.com/UploadFile/nipuntomar/connection-strings-for-oracle/
32. https://www.oracle.com/webfolder/technetwork/tutorials/obe/db/12c/r1/appdev/dotnet/Web_version_Fully_Managed_ODPnet_OBE/odpnetmngdrv.html
33. https://stackoverflow.com/questions/46408085/oracle-connection-in-c-sharp-connection-string
34. https://stackoverflow.com/questions/8202999/encrypt-a-connection-string-how
35. https://stackoverflow.com/questions/8195099/java-how-to-store-password-used-in-application/8195195#8195195
36. https://www.appmarq.com/public/tqi,1020820,Avoid-hardcoded-passwords-TypeScript
37. https://docs.snyk.io/scan-application-code/snyk-code/security-rules-used-by-snyk-code/javascript-and-typescript
38. https://rules.sonarsource.com/typescript/RSPEC-2068
39. https://rules.sonarsource.com/php/RSPEC-2068
40. https://offensive360.com/how-to-prevent-hardcoded-passwords/