1.1.1 Kriptografik Olmayan Zayıf PRNG Kullanılması (Use of Non-Cryptographic Random) (CWE-330)

Açıklık Önem Derecesi: Orta

Açıklığın Etkisi: Yetkisiz Erişim

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Uygulama dünyasında oturum jetonları (session token'lar), eposta doğrulama linkleri, captcha gibi birçok alanda sözde rastgele sayı üreteçleri (yani PRNG - Pseudo Random Number Generator) kullanılmaktadır. Sözde rastgele sayı üreteçleri zayıf bir şekilde, yani sample (örnek) boyutu düşük bir şekilde sayı üretiyorsa ve ürettiği sayılar kümesi istatistiksel olarak tekdüze dağılım sergiliyorsa bu sözde rastgele sayı üreteci neredeyse deterministiktir denebilir. Bu durumda sözde rastgele sayı üretecinden birkaç adet oluşturulan değer toplandığında saldırganın sözde rastgele sayı üretecindeki önceki değerleri ve sonraki değerleri hesaplayabilmesi mümkün olur. Yani saldırganlar uygulamanın hangi işlevi bu sözde rastgele sayı üretecini kullanıyorsa o işlevini yetkisizce kullanabilir. Uygulamalar geliştirildiği dile bağlı olarak "güvensiz" rastgele sayı üreteçleri yerine "güvenli" rastgele sayı üreteçleri kullanmalıdırlar.

- Örneğin bir uygulamada zayıf bir sözde rastgele sayı üreteci ile oluşan değerlerle belirli bir kaynağa erişim kısıtlaması sağlanıyorsa bu durumda erişimi kısıtlı olan kaynağa üretilen rastgele değerin tahmin edilmesiyle yetkisizce erişilebilir.

- Örneğin bir uygulamada zayıf bir sözde rastgele sayı üreteci ile oluşan değerlerle CSRF jetonu oluşturuluyorsa jetonun tahmin edilebilir olması nedeniyle saldırganlar jetonun değerini tahmin edip CSRF saldırıları düzenleyebilirler.

- Örneğin bir uygulamada zayıf bir sözde rastgele sayı üreteci ile oluşan değerlerle eposta üzerinden gönderilen parola sıfırlama jetonlu link oluşturuluyorsa jetonun tahmin edilebilir olması nedeniyle saldırganlar bir hesabı ele geçirebilirler. Çünkü saldırganlar parola değiştirme form'unun linkini tahmin edebilir haldedirler.

- Örneğin bir uygulamada kimlik doğrulama (authentication) veya yetkilendirme (authorization) mekanizmaları bir fonksiyonelliğe erişimi kısıtlamak adına kullanılıyorsa ve bu mekanizmalarda zayıf sözde rastgele sayı üreteci ile oluşan

rastgele değerler (örn; session ID) kullanılıyorsa bu durumda saldırgan rastgele değerleri (örn; session ID'yi) tahmin ederek erişimi kısıtlı fonksiyonelliğe yetkisizce erişebilir.

Zayıf sözde rastgele sayı üreteci kullanımı açıklığını görsel olarak göstermek adına bazı örneklere yer verilmiştir.

PHP Güvensiz Örnek:

```php
function generateSessionID($userID){
   srand($userID);              // Seed (Tohum) Değeri Verilir.
   return rand();               // Rastgele Sayı Hesaplanır.
}
```

Bu örnekte kullanıcı session (oturum) ID'sini unique (benzersiz) olacak şekilde rastgele oluşturmak için bir metot yer almaktadır. Bu metotta sözde rastgele sayı üretecinin (PRNG'nin) seed (tohum) değeri kullanıcı id'si şeklinde belirlenmiştir. Sözde rastgele sayı üretecinin seed (tohum) değeri her zaman aynı olduğundan üretilen kullanıcı session (oturum) ID'leri her zaman aynı üretilecektir. Saldırgan bu rastgele değer üretim şeklini fark ettiğinde herhangi bir kullanıcının kullanıcı adından yola çıkarak session ID'sini (oturum id'sini) tahmin edebilir. Bu şekilde kullanıcının oturumunu çalabilir ve kullanıcı adına uygulamada oturum açabilir. Dolayısıyla bu sözde rastgele sayı üreteci kullanımı zayıftır ve güvensizdir.

Java Güvensiz Örnek:

```java
String GenerateReceiptURL(String baseUrl) {

      Random ranGen = new Random();
      ranGen.setSeed((new Date()).getTime());

      return(baseUrl + ranGen.nextInt(400000000) + ".html");

}
```

Java diliyle gösterilen bu örnekte bir satın alım sonrası belirli süre aktif kalacak bir fatura url adresi sözde rastgele sayı üreteci ile oluşturulmaktadır. Burada fatura url adresini unique (benzersiz) bir id olarak oluşturmak için Random.nextInt() fonksiyonu kullanılmaktadır. Random.nextInt() fonksiyonu kriptografik bir sözde rastgele sayı üreteci olmadığından, yani tahmin edilebilir bir sözde rastgele sayı üreteci olduğundan saldırganlar bu fonksiyonunun oluşturduğu string'i tahmin edebilirler. Kodlamaya göre

faturalandırma sistemi hatalı tasarlanmış durumda olsa da tahmin edilebilir sözde rastgele sayı üreteci kullanmak yerine kriptografik sözde rastgele sayı üreteci kullanmak sistemi daha güvenli yapacaktır. Mevcut bu kullanım ile uygulama kullanıcılarının faturaları kötü niyetli kişilere ifşa olabilir ve kötü niyetli kişiler bu faturaları herkese ifşa edebilir. Dolayısıyla bu sözde rastgele sayı üreteci tahmin edilebilir olduğundan zayıftır ve güvensizdir.

Sözde rastgele sayı üreteçleri rastlantısallığı sağlamak açısından işletim sistemi üzerinden çeşitli bilinen değerleri alıp rastlantısallık hesaplamalarında kullanabilirler. Örneğin işletim sistemindeki saati kullanmak gibi. İşletim sisteminde saati kullanmak rastgele değer üretirken dar bir entropi oluşturmaktadır. Bu nedenle kullanılmaması gerekir. Ayrıca uygulamada kullanılan bu v.b. diğer değerler işletim sistemi tarafından sağlandığından işletim sistemi üzerinde belli düzeyde tersine mühendislik çalışmaları yapıldığında rastgele değer üreteçlerinde ne tür değerlerin nasıl kullanıldığı anlaşılabilir. Bu v.b. nedenlerle önemli bir uygulamada rastgelelik daha güçlü yöntemlerle sağlanmalıdır. Aksi takdirde üretilen değerlerin kötü niyetli kişiler tarafından tahmin edilebilmesine olanak sağlanmış olur.

Kurum uygulamasında zayıf sözde rastgele sayı üreteci kullanıldığı tespit edilmiştir. Bu durum Şekil XXX. ABCDEF'de gösterilmiştir.

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

Açıklığın Önlemi:

Sadece rastgele sayı üretmek ve kullanmak için kullanılan algoritmalarda hazır rastgele sayı üreteçleri kullanılması olağan olsa da önemli uygulamalar için güçlü ve donanımsal rastgele sayı üreteçleri kullanılmalıdır.

Örnek olarak Windows sistemler için "RtlGenRandom", eski Windows sistemler için "CyptGenRandom", Linux sistemler için de "hw_random()" kullanılabilir.

Uygulamalarda zayıf sözde rastgele sayı üreteçleri kullanımları yerine kriptografik olarak güvenli sözde rastgele sayı üreteçleri kullanılmalıdır. Örn;

- Java projelerde "java.security.SecureRandom" kütüphanesi,

Örn;

```
// GÜVENSİZ KOD

// ENG: Use of Cryptographically Insecure PRNG
// TR : Güvensiz Kriptografik PRNG Kullanılması

…
    Random random = new Random();
    Long sessNum = random.nextLong();
    String sessionId = sessNum.toString();
…
```

```
// GÜVENLİ KOD

// ENG: Use of Cryptographically Secure PRNG
// TR : Güvenli Kriptografik PRNG Kullanılması
…
    SecureRandom random = new SecureRandom();
    byte sessBytes[] = new byte[32];
    random.nextBytes(sessBytes);
    String sessionId = new String(sessBytes);
…
```

- C# projelerde “System.Security.Cryptography.RandomNumberGenerator” kütüphanesi,

- C ve C++ projelerde “libsodium” kütüphanesi ve randombytes_buf() metodu,

- PHP projelerde yerleşik kütüphanelerde yer alan random_bytes() ve random_int() metotları

kullanılabilir.

Referanslar:

1. https://cwe.mitre.org/data/definitions/330.html
2. https://rules.sonarsource.com/java/tag/cwe/RSPEC-4347
3. https://www.php.net/manual/tr/function.srand.php
4. https://cwe.mitre.org/data/definitions/338.html
5. https://programmerall.com/article/78802055655/
6. https://en.wikipedia.org/wiki/CryptGenRandom
7. https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.randomnumbergenerator?view=net-6.0
8. https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.randomnumbergenerator?view=net-6.0

9. https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.randomnumbergenerator.getbytes?view=net-6.0#System_Security_Cryptography_RandomNumberGenerator_GetBytes_System_Span_System_Byte__
10. https://www.csharpcodi.com/csharp-examples/System.Security.Cryptography.RandomNumberGenerator.Create()/
11. https://docs.microsoft.com/en-us/windows/win32/api/ntsecapi/nf-ntsecapi-rtlgenrandom
12. https://cpp.hotexamples.com/examples/-/-/RtlGenRandom/cpp-rtlgenrandom-function-examples.html
13. https://www.geeksforgeeks.org/random-vs-secure-random-numbers-java/
14. https://paragonie.com/blog/2016/05/how-generate-secure-random-numbers-in-various-programming-languages
15. https://find-sec-bugs.github.io/bugs.htm
16. https://jazzy.id.au/2010/09/20/cracking_random_number_generators_part_1.html