1.1.1 İstemcide Kriptografik Olmayan Zayıf PRNG Kullanılması (Client Insecure Randomness) (CWE-330)

Açıklık Önem Derecesi: Düşük

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 işlevinde yetkisizce kullanbilir. 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 zayıf bir sözde rastgele sayı üreteci ile oluşan değerlerle OTP sms kodu gönderiliyorsa OTP sms kodlarının tahmin edilebilir olması nedeniyle

saldırganlar eğer kurbanın kullanıcı adı ve şifresine halihazırda zaten sahiplerse kurbanın telefonunu ele geçirme zahmetine girmeden SMS kod tahmini ile kurbanın hesabına giriş yapabilirler, eğer kurbanın kredi kart bilgilerine (kart numarası, son kullanım tarihi ve CVV koduna) sahiplerse kurbanın telefonunu ele geçirme zahmetine girmeden SMS kod tahmini ile kurbanın kredi kart bilgilerini e-ticaret web sitelerinde kullanabilirler ve e-alışverişler yapabilirler.

- 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.

İstemci tarafta zayıf sözde rastgele sayı üreteci kullanımı açıklığını görsel olarak göstermek adına bir örneğe yer verilmiştir.

Javascript Örneği - Güvensiz Hal:

```
// Değer Üretmek İçin Math.Random() Kullanılması
// MIN ve MAX Arasında Rastgele Bir Değer

var sessionId = Math.floor(Math.Random() * MAX_RANGE - MIN_RANGE + 1) +
MIN_RANGE;
```

Javascript Örneği - Güvenli Hal:

```
// Rastgele Uint32Array Üretmek İçin
// CSPRNG (Cryptographically Secure
// Pseudo Random Number Generator)
// Kullanılması

// Not: window.crypto.getRandomValues'ün
// mevcut olduğu varsayılıyor.

var sessionId = new Uint32Array(10);
window.crypto.getRandomValues(sessionId);
```

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 "istemci tarafta" zayıf sözde rastgele sayı üreteci kullanıldığı tespit edilmiştir. Bu durum Şekil XXX. ABCDEF'de gösterilmiştir.

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

Açıklığın Önlemi:

Bu açıklıktan korunmak için;

- Basit rastgele sayı üreteci metotları yerine kriptografik güvenli sözde rastgele sayı üreteci (CSPRNG) metotları her daim kullanılmalıdır (özellikle güvenlik bağlamındaki aktivitelerde).
- Kriptografik sözde rastgele sayı üretecine (CSPRNG'ye) güvenli bir "seed" değeri eklenmelidir. Zayıf veya rastgele olmayan "seed" değeri eklenmemelidir (çoğu durumda seed değeri kullanılmadığında varsayılan seed güvenli bir rastgele değerdir).
- Yeterince rastgele değer uzunluğu kullanıldığından emin olunmalıdır. Bu sayede sonraki veya önceki rastgele sayı değerlerinin brute-force saldırıları ile tespit edilmesi olanaksız hale gelebilir.

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
17. https://rules.sonarsource.com/typescript/RSPEC-2245