1.1.1 Depolu Dizin Gezinme (Stored Path Traversal) (CWE-36)

Açıklık Önem Derecesi: Orta

Açıklığın Etkisi: Hassas Dosyaların Çalınması, SSRF saldırısı düzenlenmesi

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Uygulamalar tasarımları gereği bazen "sistem içerisinden" bir dosya yolu bilgisi alabilmektedirler. Bu şekilde sistemin yerel diskindeki dosyalara erişerek kullanabilmektedirler. Saldırganlar bu tarz uygulamalarda bu işleyişleri eğer dosya yolu bilgisi kontrol edilmiyorsa kötü yönde kullanabilirler. Saldırganlar çeşitli farklı zafiyetler yoluyla sisteme başarılı bir şekilde sızdıklarında girdikleri özel hazırlanmış dosya yolları ile uygulamanın zararlı dosya yolu girdisini kullanmasını sağlayabilirler. Bu şekilde saldırganlar mevcut elde ettikleri sızma derinliğinde olmayan, uygulamanın kullanıcılarına hizmet olarak sunduğu uygulama sunucusunun yerel diskindeki dosyalarla birlikte ayrıca uygulamanın kullanıcılarına hizmet olarak sunmadığı uygulama sunucusunun yerel diskindeki dosyalara da erişim imkanı elde edebilirler. Bu ise geliştiricileri ciddi bilgi ifşalarıyla veya teknik problemlerle başbaşa bırakabilir.

Saldırganlar uygulamanın kullanması adına keyfi (rastgele) dosya yolu girdisi girerek;

- Uygulama sunucusunun yerel diskindeki hassas dosyaları çalabilirler (örn; konfigürasyon dosyalarını veya sistem dosyalarını)
- Uygulama sunucusunun yerel diskindeki dosyaların üzerine yazma yapabilirler (örn; program binary'leri, konfigürasyon dosyaları veya sistem dosyaları v.b. üzerine)
- Uygulama sunucusunun yerel diskindeki kritik dosyaları silebilirler ve böylece DoS saldırısı gerçekleştirebilirler.

Uygulamalar kullanıcı girdisi olarak dosya yolu aldıklarında ve bu girdi denetlenmeden / filtrelenmeden uygulamada kullanıldığında buna "Depolu Dizin Gezinme (CWE-36)" açıklığı adı verilir.

Açıklığı somutlaştırmak adına bazı teknolojilerde örnekler verilmiştir:

Java - Güvensiz Kod Bloğu:

```java
// ENG: Stored Path Traversal from DB
// TUR: Veri tabanından Çekilen Dosya
//      Yolu ile Depolu Dizin Gezinme

String query = "SELECT username FROM users WHERE id = ?";
PreparedStatement pstmt = con.prepareStatement(query);
pstmt.setInt(userID, 1);

ResultSet resultSet = pstmt.executeQuery();
resultSet.next();
path = resultSet.getString(1);
con.close();

File file = new File(path);
Scanner scanner = new Scanner(file);
System.out.println(scanner.nextLine());
scanner.close();
```

Java - Güvenli Kod Bloğu:

```java
// TUR: Veri Tabanından Çekilen Dosya Yolu
//      Üzerinde Paths.get() ve getName()
//      Kullanarak Dosya Yolunu Ayrıştırarak
//      Giderilmiş Depolu Dizin Gezinme
// ENG: Stored Path Traversal from DB
//      Mitigated by Utilizing Paths.get()
//      and getName()

String query = "SELECT username FROM users WHERE id = ?";
PreparedStatement pstmt = con.prepareStatement(query);
pstmt.setInt(userID, 1);

ResultSet resultSet = pstmt.executeQuery();
resultSet.next();
path = resultSet.getString(1);
con.close();

File file = new File(path);
String fileName = file.getName();
Path safePath = Paths.get("userFiles", fileName);
file = new File(safePath.toString());

Scanner scanner = new Scanner(file);
System.out.println(scanner.nextLine());
scanner.close();
```

Güvensiz java kod örneğinde veri tabanına bir sorgu yapılmaktadır ve bir dosya yolu path nesnesine aktarılmaktadır. Ardından bu dosya yolu kullanılarak scanner sınıfı ile dosyayı açma işlemi uygulanmaktadır. Bu noktada dosya yolu (path nesnesi) doğrulama / kontrol adımları uygulanmadan dosya açma işleminde kullanıldığından güvensiz kabul edilir.

Güvenli java kod örneğinde veri tabanına bir sorgu yapılmaktadır ve bir dosya yolu path nesnesine aktarılmaktadır. Ardından dosya yolu parse edilmektedir (ayrıştırılmaktadır) ve getName() metoduyla dosya yolundaki "dosya adı" cımbızlanmaktadır. Daha sonra Paths.get() metoduyla cımbızlanan dosya adının başına belirli bir path (userFiles/) eklenmektedir. Son olarak inşa edilen dosya yolu (path nesnesi) Scanner() ile dosya açma işleminde kullanılmaktadır. Bu örnekte dosya yolu parse edildiğinden (ayrıştırıldığından) güvenli kabul edilmektedir.

PHP - Güvensiz Kod Bloğu (1):

```php
<?php

// ENG: Stored Absolute Path Traversal from DB
// TUR: Veri tabanından Çekilen Dosya Yolu
//      ile Depolu Mutlak Dizin Gezinme

...

$pstmt = $con->prepare("SELECT username FROM users WHERE id = ?");
$pstmt -> bind_param("i", $userID);
$pstmt -> execute();
$pstmt -> bind_result($path);
$pstmt -> fetch();

if(is_readable($path)){
  $file = fopen($path, 'rb');
  fpassthru($file);
}
else {
  //return 404
}

?>
```

Güvensiz php kod örneğinde veri tabanına bir sorgu yapılmaktadır ve bir dosya yolu $path nesnesine aktarılmaktadır. Ardından bu dosya yolu kullanılarak fopen() metodu ile dosyayı açma işlemi uygulanmaktadır. Bu noktada dosya yolu (path nesnesi) doğrulama / kontrol adımları uygulanmadan dosya açma işleminde kullanıldığından güvensiz kabul edilir.

PHP - Güvensiz Kod Bloğu (2):

```php
<?php

// ENG: Stored Relative Path Traversal from DB
// TUR: Veri tabanından Çekilen Dosya Yolu
//      ile Depolu Göreceli Dizin Gezinme

...

$pstmt = $con->prepare("SELECT username FROM users WHERE id = ?");
$pstmt -> bind_param("i", $userID);
$pstmt -> execute();
$pstmt -> bind_result($path);
$pstmt -> fetch();

$filename = "public_files/" . $path;

if(is_readable($filename)){
  $file = fopen($filename, 'rb');
  fpassthru($filename);
}
else {
  //return 404
}

?>
```

Güvensiz php kod örneğinde veri tabanına bir sorgu yapılmaktadır ve bir dosya yolu $path nesnesine aktarılmaktadır. Ardından bu dosya yolu public_files/ dosya yolu sonuna eklenmektedir. Son olarak fopen() metodu ile mevcut dosya yolundaki dosyayı açma işlemi uygulanmaktadır. Bu noktada dosya yolu (path nesnesi) doğrulama / kontrol adımları uygulanmadan dosya açma işleminde kullanıldığından güvensiz kabul edilir.

PHP - Güvenli Kod Bloğu:

```php
<?php

// TUR: basename() Kullanarak Giderilmiş
//      Depolu Göreceli Gezinme Zafiyeti
// ENG: Stored Relative Path Traversal
//      from DB Mitigated by Utilizing
//      basename()

...

$pstmt = $con->prepare("SELECT username FROM users WHERE id = ?");
$pstmt -> bind_param("i", $userID);
$pstmt -> execute();
$pstmt -> bind_result($path);
$pstmt -> fetch();

$fileName = "public_files/" . basename($path);

if(is_readable($fileName)){
  $file = fopen($fileName, 'rb');
  fpassthru($file);
}
else {
  //return 404
}

?>
```

Güvenli php kod örneğinde veri tabanına bir sorgu yapılmaktadır ve bir dosya yolu $path nesnesine aktarılmaktadır. Ardından dosya yolu public_files/ dosya yolu sonuna eklenmektedir, fakat eklenirken parse edilmektedir (ayrıştırılmaktadır) ve basename() metoduyla dosya yolundaki “dosya adı” cımbızlanmaktadır. Daha sonra dosya adının başına belirli bir path (public_files/) eklenmektedir. Son olarak inşa edilen dosya yolu ($filename nesnesi) fopen() ile dosya açma işleminde kullanılmaktadır. Bu örnekte dosya yolu parse edildiğinden (ayrıştırıldığından) güvenli kabul edilmektedir.

Not:

Depolu dizin gezinme zafiyetinin var olması için veritabanından dosya yolunun alınması şart değildir. Sadece sistemin kaynaklarından bir yerden gelmesi önemlidir. Örneğin uygulama sunucu sistemindeki environment variable’dan (ortam değişkeninden) gelen dosya yolları ile de ayrıştırma uygulanmıyorsa depolu dizin gezinme zafiyeti vardır denir.

Kurum web uygulamada “Depolu Dizin Gezinme (CWE-36)” açıklığı tespit edilmiştir.

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

Açıklığın Önlemi:

Tavsiyeler şu şekildedir:

- Tüm girdiler kaynağı neresi olursa olsun doğrulamadan geçirilmelidir. Bu doğrulama belirli yapıdaki girdileri reddetme işlemi yerine sadece belirli yapıya uyanların kabul edildiği şeklinde olmalıdır. Yani whitelist (beyaz liste) kullanılmalıdır. Siyah liste (blacklist) önlemi kullanılmamalıdır. Bir girdiyi doğrulamada şunlar kontrol edilebilir:
    - Veri türü (int, string, float, byte, … v.b. tipte mi geliyor kontrolü)
    - Boyutu (x KB, y MB,…v.b. boyutta mı geliyor kontrolü)
    - Aralığı (Veri ölçülebilir bir sayısal değerse aralık sınırlarını aşıyor mu kontrolü)
    - Formatı (çözümlenebilir şifreli, tek yönlü şifreli, açık metin, … v.b. biçimde mi geliyor kontrolü)
    - Beklenen Değerlerden Biri Olup Olmadığı (Whitelist’te tanımlı desenlerden biri mi geliyor kontrolü)
- Dosya yolunun doğrulandığından (canonicalized edildiğinden) emin olunmalıdır.
- Uygulama “uygulamalar binary klasörü”nden ayrı olan tasarlanmış bir klasörü kullanacak şekilde sınırlandırılmalıdır.
- Uygulamanın çalıştığı işletim sistemi kullanıcısının ayrıcalıkları gerekli dosyalar ve klasörler için kısıtlanmalıdır. Uygulama “uygulama binary klasörü”ne yazma yapamıyor olmalıdır ve uygulama dosya ve veri klasörleri dışındaki herhangi bir şeyi okumamalıdır.

Referanslar:

1. https://cwe.mitre.org/data/definitions/36
2. https://www.w3schools.com/php/func_filesystem_basename.asp