1.1.1 Null İşaretçi Değeri Kullanımını Saptamak İçin NullPointerException Catch'i Kullanılması (Use of NullPointerException Catch to Detect Null Pointer Dereference) (CWE-395)

Açıklık Önem Derecesi: Düşük

Açıklığın Etkisi: Kaynak Tüketimi, Servis Dışı Kalma

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

"Null İşaretçi Değerini Kullanma" piyasada oldukça yaygın bir şekilde uygulamalarda bulunan bir açıklık olarak öne çıkmaktadır. Popüler birçok dilde (örn; C, C++, Java, JavaScript, C#, Python, Ruby, PHP, Perl, Go, Objective-C, Swift, ...) var olan bir durumdur. "Null İşaretçi Değerini Kullanma", bir uygulama geçerli bir değer göstermesi beklenen bir pointer'ı (işaretçiyi) veya nesneyi değerini gösterecek şekilde kullandığında pointer'ın (işaretçinin) veya nesnenin geçerli bir adres yerine NULL adres içermesi sonucu gelen hata durumuna denir. Örneğin C programlama dilinde bu durum şu şekilde örneklenebilir:

```
int *x = NULL;          // x değişkeni bir null işaretçi olur.

int y = *x;             // Process Bu Satırda Crash Olur (Çöker):
                        // Değerini okumayı deneyerek x değişkeni
                        // değerini gösterme (dereference'e etme).

*x = 0;                 // Process Bu Satırda Crash Olur (Çöker):
                        // Yazma deneyerek x değişkeni değerini
                        // gösterme (dereference'e etme).
```

Yukarıdaki c örneklemesinde gösterildiği gibi NULL adres içeren bir x işaretçi değişkeni önce y = *x ile gösterdiği değer nedir okumasına tabi tutuluyor ve y değişkenine bu değerin atanması yapılıyor, ardından *x = 0 ile gösterdiği değere veri yazma işlemine tabi tutuluyor. x işaretçi değişkeni geçerli bir adres içermediğinden ne değer gösterebilir ne de gösterebileceği bir değer için atama alabilir. Bu nedenle bu iki kullanım da programın çöküşüne sebep olur. Bu programlama hatasının kötü yönde kullanılabilir olması bu hatayı bir güvenlik açıklığı yapmaktadır.

“Null İşaretçi Değerini Kullanma” şeklinde adlandırılan açıklığa göre bu açıklık daima uygulamaların crash olmasına (çökmesine) sebep olur. Ayrıca bu açıklık bazen NULL ifadesi 0x0 bellek adresine denk geldiği durumlarda ve kodların bu adrese erişimi mümkün olduğu durumlarda belleğe veri yazmanın ve bellekten veri okumanın mümkün hale girmesine sebep olur ki bu durum uygulamada veya uygulama sunucusunda kod çalıştırmaya yol açar.

Null işaretçi değerini kullanma açıklığını görsel anlamda göstermek adına bazı örneklere yer verilmiştir:

C Örneği:

```c
void host_lookup(char *user_supplied_addr){
   struct hostent *hp;
   in_addr_t *addr;
   char hostname[64];
   in_addr_t inet_addr(const char *cp);

   validate_addr_form(user_supplied_addr);
   addr = inet_addr(user_supplied_addr);
   hp = gethostbyaddr( addr, sizeof(struct in_addr), AF_INET);
   strcpy(hostname, hp->h_name);
}
```

C örneğine bakılacak olursa; kullanıcı taraflı girilen ve user_supplied_addr işaretçisinin getirdiği ip adres girdisi eğer düzgün formattaysa, ancak hostname’i çözümlenemez durumdaysa - ki bazı ip adreslerin hostname’i çözümlenemeyebilir - bu durumda gethostbyaddr() fonksiyonu hp işaretçisine NULL bir değer döndürecektir. Çünkü hostname çözümlenemeyecektir. Akabinde hp işaretçisi null mı değil mi kontrolüne tabi tutulmadan bir aşağıda yer alan strcp() fonksiyonunda hp->h_name ile kullanıma sokulduğunda hp işaretçisi null değerde olduğundan h_name üyesi gösterilemeyecektir. Bu noktada NullPointerException adı verilen bir hata meydana gelecektir. Bu hata bir açıklık teşkil eder. Saldırganlar bu açığı fark ettiklerinde kullanıcı girdisi yolunu suistimal ederek son kullanıcı bilgisayarlarındaki uygulamada veya uygulama sunucusunda servis dışı bırakma saldırıları uygulayabilirler veya gerekli şartlar mevcutsa kullanıcı girdisi yolunu suistimal ederek son kullanıcı bilgisayarlarındaki uygulamada veya uygulama sunucusunda kod çalıştırma saldırıları uygulayabilirler.

Java Örneği:

```
String cmd = System.getProperty("cmd");
cmd = cmd.trim();
```

Java örneğine bakılacak olursa; yazılımcı bu uygulamada daima tanımlı bir cmd adlı property (özellik) varmış varsayımını yapmaktadır. Fakat örneğin son kullanıcı bilgisayarındaki işletim sistemi linux olabilir. Bu durumda son kullanıcı bilgisayarındaki uygulamada cmd nesnesi null değer alacaktır ve akabinde aşağısındaki satırda yer alan trim() metot çağırımı null'dan bir metot çağırımı olacağından çağırılamayacaktır. Bu noktada NullPointerException hatası meydana gelecektir. Bu hata bir açıklık teşkil eder. Saldırganlar bu açıklığı fark ettiklerinde bu yoldan giderek program ortamından hareketle linux son kullanıcı bilgisayarlarında cmd.trim() satırındaki tanımsız olacak cmd özelliğinin trim() metodunu çağırması işlevini tetikleyebilirler. Bu yolla linux son kullanıcı bilgisayarındaki uygulamada servis dışı bırakma veya gerekli şartlar mevcutsa linux son kullanıcı bilgisayarlarındaki uygulamada kod çalıştırma saldırıları uygulayabilirler.

Not:

Bu java örneğinde uygulama sunucusu mevcutsa sunucu bu kodlamayı barındırdığında risk altında olmaz, çünkü uygulama sunucusunun ortamı (linux / windows) değişken değildir. Burada uygulama çevrimdışı bir uygulamaysa uygulamanın yüklü olduğu ve ortamın değişken olduğu son kullanıcı bilgisayarları risk altındadır.

Android Örneği:

```
...
IntentFilter filter = new IntentFilter("com.example.URLHandler.openURL");
MyReceiver receiver = new MyReceiver();
registerReceiver(receiver, filter);
...

public class UrlHandlerReceiver extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
      if("com.example.URLHandler.openURL".equals(intent.getAction())) {
         String URL = intent.getStringExtra("URLToOpen");
         int length = URL.length();
         ...
      }
   }
}
```

Android örneğine bakılacak olursa; android uygulamasında bir URL adres almak üzere bir intent tanımlaması yapılmıştır. Uygulama URL adresin arayüzdeki ekrandan daima dahil olacağını varsaymaktadır. Fakat URL adresi mevcut olmadığı durumda kodlama satırındaki getStringExtra() metodunu çağırmak NULL değer döndürecektir. Akabinde null değer almış bir URL adlı string'ten lenght() metodu çağırmak null'dan bir metot çağırımı olacağından çağırılamayacaktır. Bu noktada NullPointerException hatası meydana gelecektir. Bu hata bir açıklık teşkil eder. Saldırganlar bu açıklığı fark ettiklerinde URL adresi girmeden ekranı onaylama ile son kullanıcıların mobil cihazlarındaki android uygulamada veya android uygulama sunucusunda servis dışı bırakma saldırıları ve gerekli şartlar mevcut olduğunda son kullanıcıların mobil cihazlarındaki android uygulamada veya android uygulama sunucusunda kod çalıştırma saldırıları uygulayabilirler.

"Null İşaretçi Değerini Kullanma" açıklığı genel itibariyle şu durumlarda meydana gelebilir:

- Null bir işaretçi değişkenin değerinin kullanılmasına çalışılırken (örn; * ile dereference edilirken),
- Null bir nesneden metot çağırılırken,
- Null bir nesnenin özelliğine/alanına (property'sine/field'ına) erişilmeye çalışılırken veya null bir nesnenin özelliği / alanı modifiye edilmeye çalışılırken,
- Bir diziymişçesine null'ın uzunluğu bilgisi alınırken,
- Bir diziymişçesine null nesnenin indisindeki değerlere erişmeye çalışılırken veya indisindeki değerlerin modifiye edilmesine çalışılırken,

- Fırlatılabilir (Throwable) bir değermişçesine null'ı fırlatırken
- Null bir nesne üzerinden çoklu thread kullanımlarında senkronize etmeyi denerken
- v.b.

Saldırganlar uygulamalardaki "Null İşaretçi Değerini Kullanma" (CWE-476) açıklığından yararlanarak web uygulama sunucularına, masaüstü uygulama sunucularına, mobil uygulama sunucularına paket gönderimleri ile veya çevrimdışı olan masaüstü uygulamalara ve çevrimdışı mobil uygulamalara arkaplanda çalışacak zararlı uygulamalar aracılığıyla servis dışı bırakma saldırıları düzenleyebilirler. Ayrıca komut çalıştırma saldırıları ile web uygulama sunucularına, masaüstü uygulama sunucularına ve mobil uygulama sunucularına sızma faaliyeti gerçekleştirilebilirler veya komut çalıştırma saldırıları ile çevrimdışı bir masaüstü uygulamasında veya çevrimdışı bir mobil uygulamada uygulamanın yeteneklerini istismar ederek zararlı (hacking) faaliyetler yürütebilirler.

"Null İşaretçi Değerini Kullanma" (CWE-476) açıklığına karşın uygulama geliştiricileri NullPointerException istisnası ve catch yapısı kullanabilmekteler. Bu istisna ve catch ile istisnayı yakalama yine açıklık oluşturmaktadır. Çünkü Null İşaretçi Değerini Kullanma durumunda catch ile hata yakalama ve işlemler uygulama kaynak tüketimine ve performans problemlerine yol açmaktadır. Null işaretçi değerini kullanma ile catch'e girildiğinde esasında null işaretçi değeri kullanımı önlenmemektedir. Null işaretçi değeri kullanımı yine uygulanmaktadır, fakat catch ile dallanma catch'e yönelip sadece ilave aksiyonlar alınması ve programın geri kalanının devam etmemesi sağlanmaktadır. Null işaretçi değeri kullanımının gerçekleşmesi ile de saldırganlar yeterince ısrarlı olduğu durumda uygulama sunucusunu servis dışı bırakabilirler.

"Null İşaretçi Değeri Kullanımı" (CWE-476) açıklığını kapatmak için NullPointerException istisnası ile catch yapısını kullanmak güvensizdir. Bu güvensiz kullanıma şu şekilde bir örnek verilebilir:

```
public int uzunlukArttirma(String kelime) {

    int len = 2;

    try {
        len += kelime.length();
    }
    catch (NullPointerException npe) {
        log.info("Arguman null'dır.");
    }

    return len;
}
```

Bu örnekte metoda gelen “kelime” argümanı null mı değil mi kontrolü yapılmadan try bloğunda kullanılmaktadır ve null değer içerdiğinde fırlatacağı NullPointerException istisnası ile program catch bloğuna kaydırılmaktadır. Fakat bu işlem kaynak tüketimine ve performans kaybına yol açmaktadır.

Güvensiz kodun güvenli hale dönüştürülmüş haline ise şu şekilde örnek verilebilir:

```
public int uzunlukArttirma(String kelime) {
    int len = 2;

    if (kelime != null) {
        len += kelime.length();
    }
    else {
        log.info("Arguman null'dır.");
    }
    return len;
}
```

Bu örnekte ise metoda gelen “kelime” argümanı null mı değil mi kontrolü yapılarak if bloğu içerisinde kullanılmaktadır ve null değerse program else koşuluna kaydırılmaktadır. Bu işlem kaynak tüketimine ve performans kaybına yol açmamaktadır. Güvensiz kodlamadaki NullPointerException istisnasının ve catch yapısının kullanılmasına “Null İşaretçi Değeri Kullanımını Saptamak İçin NullPointerException Catch’i Kullanılması” (CWE-395) açıklığı denir.

Kurum uygulamasında “Null İşaretçi Değeri Kullanımını Saptamak İçin NullPointerException Catch’i Kullanılması” (CWE-395) açıklığı tespit edilmiştir:

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

Açıklığın Önlemi:

NullPointerException istisnasının catch ile yakalanması durumundan sakınılması gerekmektedir. Bu istisnanın catch ile yakalanması kolaylıkla programlamatik null kontrolüne dönüştürülebilir ve catch bloğundaki aksiyonlar kolaylıkla null mı kontrolü yapan if koşulu içerisinde uygulanabilir. Dolayısıyla null kontrolünü null istisnası yakalayan catch ile yapmak yerine if koşuluyla yapmak gerekir.

Referanslar:

1. https://rules.sonarsource.com/java/RSPEC-1696
2. https://cwe.mitre.org/data/definitions/395.html
3. https://owasp.org/www-community/vulnerabilities/Catch_NullPointerException
4. https://stackoverflow.com/questions/4007268/what-exactly-is-meant-by-de-referencing-a-null-pointer
5. https://cwe.mitre.org/data/definitions/476.html
6. https://owasp.org/www-community/vulnerabilities/Null_Dereference
7. https://www.geeksforgeeks.org/null-pointer-exception-in-java/
8. http://dobegin.com/npe-hell/
9. https://owasp.org/www-community/vulnerabilities/Catch_NullPointerException
10. https://wiki.sei.cmu.edu/confluence/display/java/ERR08-J.+Do+not+catch+NullPointerException+or+any+of+its+ancestors
11. https://www.quora.com/Is-it-bad-practice-to-catch-null-pointer-exceptions-in-Java
12. https://howtodoinjava.com/java/exception-handling/how-to-effectively-handle-nullpointerexception-in-java/
13. https://stackoverflow.com/questions/17302256/best-way-to-check-for-null-values-in-java
14. https://stackoverflow.com/questions/6417902/checking-if-an-object-is-null-in-c-sharp
15. https://stackoverflow.com/questions/13818175/way-to-check-for-null-value-in-if-condition
16. https://www.programiz.com/csharp-programming/ternary-operator
17. https://www.codegrepper.com/code-examples/csharp/ternary+operator+c%23+check+null
18.