Kaynak Kod Güvenliği
Bir Güvensiz API Örneği

Bedirhan Urgun, Ağustos 2010, WGT E-Dergi 6. Sayı

Bu yazıda Tomcat J2EE kısmi uygulama sunucusunda bulunan bir güvenlik açığına, güvenlik probleminin kaynağına ve açıklamasına yer vermeye çalışacağız. CVE-2008-2370 kodlu açıklık Tomcat uygulama sunucusunun 4, 5 ve 6 versiyolarında Stefano Di Paola tarafından bulunmuştur.

Tomcat kaynak kodunda bulunan bu zafiyet ile kullanıcılara (J2EE kod geliştiriciler) güvensiz bir API sağlanmıştır. Bu API'ı kullanan geliştiriciler bilmeden yazdıkları yazılımlarda bir güvenlik problemi oluşturmaktadırlar. Girdi denetimi ile kolaylıkla her iki seviyede de düzeltilebilecek bu açık içeriği ile beraber aynı zamanda kod geliştirme aşamasında güvenli mimarinin de önemini bir kez daha anlatmaktadır.

Yazılan algoritmalar üzerinde saldırgan gibi düşünebilen yazılımcılar bulunması kolay olmayabilir ancak örneklerle eğitilmiş bir yazılımcı kendi geliştirdiği yazılım üzerinde bir güvenlik uzmanından daha etkili incelemelerde bulunacaktır.

Saldırgan Gözü

İlgili Tomcat problemine detaylı göz atmadan saldırgan gözü ile zafiyetin nasıl exploit edilebildiğini görelim. Aşağıdaki yazılımcımız tarafından üretilmiş JSP parçasına bakalım;

```
<%
pageContext.forward("/page2.jsp?somepar=someval&par="+request.getParameter("blah")
);
%>
```

Yapılmak istenen, kullanıcıdan alınan blah isimli parametre değerini kullanarak başka bir URL oluşturmak ve uygulama sunucu üzerinde bulunan bir diğer page2.jsp JSP dosyasını çalıştırıp sonucunu kullanıcıya dönmektir.

Kullanıcı tarafından blah parametresinin değeri someotherval olarak verilirse, pageContext.forward metoduna gönderilecek argüman aşağıdaki gibi olacaktır.

```
/page2.jsp?somepar=someval&par=someotherval
```

Ve page2.jsp JSP dosyası somepar ve par parametreleri ve verilen değerleri ile çalıştırılacaktır. Ancak kullanıcı blah parametresinin değeri olarak /../WEB-INF/web.xml verirse, forward metoduna gönderilecek parametre aşağıdaki gibi üretilecektir.

```
/page2.jsp?somepar=someval&par=/../WEB-INF/web.xml
```

Metodun çalışması ile beraber normalde /WEB-INF/ dizini altındaki ulaşılması yasaklanan dosyalara ulaşılabilecektir. Bu örnekte ulaşılmak istenen dosya J2EE uygulamaları yapılandırma dosyası web.xml'dir.

Geliştirici Gözü

Peki sorun neden kaynaklanmaktadır? Bunu anlamak için pageContext.forward metodu incelenmelidir.

```
/**
* Return a RequestDispatcher instance that acts as a
* wrapper for the resource at the given path. The path must begin

* with a "/" and is interpreted as relative to the current context root.

*

* @param path The path to the desired resource.

*/

public RequestDispatcher getRequestDispatcher(String path) {

  // Validate the path argument

  if (path == null)

    return (null);

  if (!path.startsWith("/"))

    throw new IllegalArgumentException

      (sm.getString

        ("applicationContext.requestDispatcher.iae", path));

  path = normalize(path);

  if (path == null)

    return (null);


  // Retrieve the thread local URI

  MessageBytes uriMB = (MessageBytes) localUriMB.get();

  if (uriMB == null) {
```

```
    uriMB = MessageBytes.newInstance();
    CharChunk uriCC = uriMB.getCharChunk();
    uriCC.setLimit(-1);
    localUriMB.set(uriMB);
  }
  else {
    uriMB.recycle();
  }

...
```

Metod üzerinde kırmızı olarak gösterilmiş normalize metodu path argümanı içerisindeki değeri sistem için anlaşılabilir hale getirecektir (yani .., ., / gibi meta karakterleri anlamları ile düzeltecektir). Yani kötü amaçlı kullanıcı tarafından gönderilmiş ve yazılım tarafından aşağıdaki gibi oluşturulmuş argüman,

```
/page2.jsp?somepar=someval&par=/../WEB-INF/web.xml
```

normalize metodundan geçtikten sonra aşağıdaki gibi bir hal alacaktır.

```
/WEB-INF/web.xml
```

Bu da, WEB-INF dizin altındaki web.xml dosyasının okunmasına ve kullanıcıya dönmesine neden olacaktır;

getRequestDispatcher metodundaki problem, alınan argümandan URL sorgu bölümü çıkarılmadan normalize metodunun çağrılmasıdır. Bu şekilde sorgu bölümünde değişiklik yapma hakkı bulunan kötü amaçlı kullanıcı sistem üzerinde yol belirleyebilmektedir. Kodun Tomcat'in diğer versiyonlarında nasıl düzeltildiğine bakalım.

```
/**
 * Standard implementation of ServletContext that represents
 * a web application's execution environment.An instance of this class is
 * associated with each instance of StandardContext.
 *
 * @author Craig R. McClanahan
```

```
* @author Remy Maucherat
* @version $Revision$ $Date$
*/

public class ApplicationContext
  implements ServletContext {


  ...
  ...


  /**
  * Return a RequestDispatcher instance that acts as a
  * wrapper for the resource at the given path.The path must begin
  * with a "/" and is interpreted as relative to the current context root.
  *
  * @param path The path to the desired resource.
  */
  public RequestDispatcher getRequestDispatcher(String path) {


  // Validate the path argument
  if (path == null)
    return (null);
  if (!path.startsWith("/"))
    throw new IllegalArgumentException
      (sm.getString
        ("applicationContext.requestDispatcher.iae", path));
```

```
    // Get query string String
    queryString = null;
    int pos = path.indexOf('?');
    if (pos >= 0) {
      queryString = path.substring(pos + 1);
      path = path.substring(0, pos);
    }

    path = normalize(path);
    if (path == null)
      return (null);

  ...
```

Kırmızı ile gösterilen kod parçası normalize metodunu kullanmadan önce, argümandaki muhtemel sorgu bölümlerini çıkarmaktadır. Bu da kullanıcıların argümanda değişiklik yapmalarının önüne geçmektedir.

Sonuç

Ancak alınan bu güvenlik önlemi de %100 koruma sağlamaz. Aşağıdaki gibi yazılmış bir kod parçasını korumak sadece ve sadece geliştiriciye kalmış gözükmektedir.

```
<%
 pageContext.forward("/dir/page"+request.getParameter("lang") + ".jsp");
&>
```

veya

```
<%
 pageContext.forward(request.getParameter("lang") + ".jsp");
%>
```

Kullanılan frameworklerin ve API'ların güvenli geliştirme süreçlerinden geçmiş olmaları gerekmektedir. Bu manuel, otomatik denetimlerle gerçekleştirilmelidir. Bunların yanında yazılım geliştiricilerin farkındalıkları da uç yazılımların güvenliğinde büyük önem taşımaktadır.