




DOS_Readlines_Finding - Findings Report

Oct 26, 2007 7:07:24 PM

Classification	Type	Severity	File	Line	CWE ID	SmartTrace
Vulnerability	AppDOS		Z:\jspwiki\JSPWiki_2_4_104\JSPWiki-src\src\com\ecyrd\jspwiki\diff\ExternalDiffProvider.java	165		
<div> <div>Description:</div> <div>The application contains a variety of different locations where unbound reads may theoretically expose the application to DOS attacks. If an attacker is capable of controlling whether the reads continue he may cause the DOS attack.</div> <div>Recommendation:</div> <div>Ensure that the reads are bound by a certain threshold to prevent DOS potentials.</div> </div> <div> <pre> BufferedReader in = new BufferedReader(new StringReader(diffText)); StringBuffer out = new StringBuffer(); out.append("<table class=\"diff\" border=\"0\" cellspacing=\"0\" cellpadding=\"0\">\n"); while((line = in.readLine()) != null) { stop = CSS_DIFF_CLOSE; if(line.length() > 0) { </pre> </div>						
Vulnerability	AppDOS		Z:\jspwiki\JSPWiki_2_4_104\JSPWiki-src\src\com\ecyrd\jspwiki\providers\RCSFileProvider.java	148		
<div> <div>Description:</div> <div>The application contains a variety of different locations where unbound reads may theoretically expose the application to DOS attacks. If an attacker is capable of controlling whether the reads continue he may cause the DOS attack.</div> <div>Recommendation:</div> <div>Ensure that the reads are bound by a certain threshold to prevent DOS potentials.</div> </div> <div> <pre> Pattern datepattern = compiler.compile(PATTERN_DATE); Pattern notepattern = compiler.compile(PATTERN_CHANGENOTE); boolean found = false; while((line = stdout.readLine()) != null) { if(matcher.contains(line, headpattern)) { MatchResult result = matcher.getMatch(); </pre> </div>						
Vulnerability	AppDOS		Z:\jspwiki\JSPWiki_2_4_104\JSPWiki-src\src\com\ecyrd\jspwiki\SearchMatcher.java	67		
<div> <div>Description:</div> </div>						

The application contains a variety of different locations where unbound reads may theoretically expose the application to DOS attacks. If an attacker is capable of controlling whether the reads continue he may cause the DOS attack.

Recommendation:

Ensure that the reads are bound by a certain threshold to prevent DOS potentials.

```
int scores[] = new int[ m_queries.length ];
BufferedReader in = new BufferedReader( new StringReader( pageText ) );
String line = null;

while( (line = in.readLine()) != null )
{
    line = line.toLowerCase();
    for( int j = 0; j < m_queries.length; j++ )
    {
```

Vulnerability AppDOS



Z:\jspwiki\JSPWiki_2_4_104\JSPWiki-src\src\com\ecyrd\jspwiki\providers\RCSFileProvider.java

471

Description:

The application contains a variety of different locations where unbound reads may theoretically expose the application to DOS attacks. If an attacker is capable of controlling whether the reads continue he may cause the DOS attack.

Recommendation:

Ensure that the reads are bound by a certain threshold to prevent DOS potentials.

```
String line;
WikiPage info = null;
while( (line = stdout.readLine()) != null )
{
    if( matcher.contains( line, revpattern ) )
    {
        info = new WikiPage( m_engine, page );
    }
```

Vulnerability AppDOS



Z:\jspwiki\JSPWiki_2_4_104\JSPWiki-src\src\com\ecyrd\jspwiki\providers\RCSFileProvider.java

278

Description:

The application contains a variety of different locations where unbound reads may theoretically expose the application to DOS attacks. If an attacker is capable of controlling whether the reads continue he may cause the DOS attack.

Recommendation:

Ensure that the reads are bound by a certain threshold to prevent DOS potentials.

```
stderr = new BufferedReader(new InputStreamReader(process.getErrorStream()));
Pattern headpattern = compiler.compile( PATTERN_REVISION );
while( (line = stderr.readLine()) != null )
{
    if( matcher.contains( line, headpattern ) )
    {
        MatchResult mr = matcher.getMatch();
        checkedOutVersion = Integer.parseInt( mr.group(1) );
    }
```

Vulnerability AppDOS



Z:\jspwiki\JSPWiki_2_4_104\JSPWiki-src\src\com\ecyrd\jspwiki\FileUtil.java

114

Description:

The application contains a variety of different locations where unbound reads may theoretically expose the application to DOS attacks. If an attacker is capable of controlling whether the reads continue he may cause the DOS attack.

Recommendation:
Ensure that the reads are bound by a certain threshold to prevent DOS potentials.

```
{
    result.append( line+"\n");
}
StringBuffer error = new StringBuffer();
while( (line = stderr.readLine()) != null )
{
    error.append( line+"\n");
}
if( error.length() > 0 )
```

Vulnerability AppDOS



Z:\jspwiki\JSPWiki_2_4_104\JSPWiki-src\src\com\ecyrd\jspwiki\FileUtil.java

108

Description:

The application contains a variety of different locations where unbound reads may theoretically expose the application to DOS attacks. If an attacker is capable of controlling whether the reads continue he may cause the DOS attack.

Recommendation:

Ensure that the reads are bound by a certain threshold to prevent DOS potentials.

```
stdout = new BufferedReader( new InputStreamReader(process.getInputStream()) );
stderr = new BufferedReader( new InputStreamReader(process.getErrorStream()) );
String line;
while( (line = stdout.readLine()) != null )
{
    result.append( line+"\n");
}
StringBuffer error = new StringBuffer();
```

Vulnerability AppDOS



Z:\jspwiki\JSPWiki_2_4_104\JSPWiki-src\src\com\ecyrd\jspwiki\providers\RCSFileProvider.java

605

Description:

The application contains a variety of different locations where unbound reads may theoretically expose the application to DOS attacks. If an attacker is capable of controlling whether the reads continue he may cause the DOS attack.

Recommendation:

Ensure that the reads are bound by a certain threshold to prevent DOS potentials.

```
// FIXME: Should this use encoding as well?

stderr = new BufferedReader( new InputStreamReader(process.getErrorStream() ) );

while( (line = stderr.readLine()) != null )
{
    log.debug( "LINE="+line );
    if( line.equals("done") )
    {
        success = true;
    }
}
```

Vulnerability AppDOS



Z:\jspwiki\JSPWiki_2_4_104\JSPWiki-src\src\com\ecyrd\jspwiki\filters\SpamFilter.java

224

Description:

The application contains a variety of different locations where unbound reads may theoretically expose the application to DOS attacks. If an attacker is capable of controlling whether the reads continue he may cause the DOS attack.

Recommendation:

Ensure that the reads are bound by a certain threshold to prevent DOS potentials.

```
{
    BufferedReader in = new BufferedReader( new StringReader(list) );

    String line;

    while( (line = in.readLine()) != null )
    {
        line = line.trim();
        if( line.length() == 0 ) continue; // Empty line
        if( line.startsWith("#") ) continue; // It's a comment
    }
}
```

Vulnerability AppDOS



Z:\jspwiki\JSPWiki_2_4_104\JSPWiki-src\src\com\ecyrd\jspwiki\providers\RCSFileProvider.java

212

Description:

The application contains a variety of different locations where unbound reads may theoretically expose the application to DOS attacks. If an attacker is capable of controlling whether the reads continue he may cause the DOS attack.

Recommendation:

Ensure that the reads are bound by a certain threshold to prevent DOS potentials.

```
// Especially with certain versions of RCS on Windows,
// process.waitFor() hangs unless you read all of the
// standard output. So we make sure it's all emptied.
//
while( (line = stdout.readLine()) != null )
{
}
process.waitFor();
```

Vulnerability AppDOS



Z:\jspwiki\JSPWiki_2_4_104\JSPWiki-src\src\com\ecyrd\jspwiki\providers\RCSFileProvider.java

394

Description:

The application contains a variety of different locations where unbound reads may theoretically expose the application to DOS attacks. If an attacker is capable of controlling whether the reads continue he may cause the DOS attack.

Recommendation:

Ensure that the reads are bound by a certain threshold to prevent DOS potentials.

```
//
BufferedReader error = null;
String elines = "";
error = new BufferedReader(new InputStreamReader(process.getErrorStream()));
String line = null;
while ((line = error.readLine()) != null)
{
    elines = elines + line + "\n";
}

log.debug("Done, returned = "+process.exitValue());
```