

Protecting Against Denial of Service



Josh Cummings

PRINCIPAL SOFTWARE ENGINEER

@jzheaux blog.jzheaux.io



Denial of Service Attack

An attack that exhausts an application's resources preventing authorized access to the system



```
<?xml version="1.0"?>
<!DOCTYPE request [
  <!ENTITY lol "lol">
  <!ELEMENT lolz (#PCDATA)>
  <!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
  <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
  <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
  <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
  <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
  <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
  <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
  <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
  <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<request>&lol9;</request>
```



[illegible]

Mitigating Denial of Service



Define and Enforce
Limits



Protect Against
Overflow



Clean-up or Pool
Resources



Defining and Enforcing Limits

External Resources

Internal Resources



```
java -Xmx1GB MyClass
```

```
System.setProperty(  
    "jdk.xml.entityExpansionLimit",  
    1000);
```

◀ Java defines and enforces a limit on heap size

25% of available memory, up to 25 GB

◀ Java XML defines and enforces a limit on entity expansion

64000



Defining and Enforcing Limits

External Resources

Internal Resources



```
map.put(key1, value1);  
map.put(key2, value2);  
// ...  
map.put(key1000, value1000);
```

```
https://example.org/index.jsp?aaaa...A  
a=1&aaaa...BB=1&...
```

- ◀ If key1..n all hash to the same value, and key1..n all have the same very long prefix, then the hashtable performs $O(n^2)$ instead of $O(1)$.
- ◀ This vulnerability was exploited in Tomcat in 2011



Java manages memory, but
not resources



```
public void doWithLock(Lock lock, Runnable todo) {  
    try {  
        lock.tryLock();  
        todo.run();  
    } finally {  
        lock.release();  
    }  
}
```

Use Lambdas to Ensure Clean-up

It's easy to forget to clean-up resources

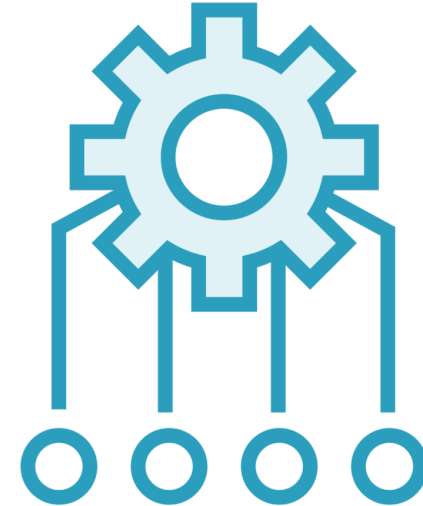
Lambdas are a technique for centralizing the clean-up logic so your application only has one place to do it



Pool Expensive Resources



Connection Pooling



Thread Pooling

```
grant codebase "jar://WEB-INF/lib/myjar.jar" {  
    permission java.io.FilePermission "pictures/-", "read"  
}
```

SecurityManager

Know how to audit existing policy files in use

Use to allowlist risky behaviors like opening resources, using the class loader, and performing reflection



Mitigating Denial of Service



To mitigate denial of service:

- Mitigate inclusion
- Define and enforce limits on external and internal resources
- Account for overflow
- Close or pool resources

When using a security manager:

- Audit policy files using principle of least privilege

You did it!

Now go be secure!

@jzheaux

<https://github.com/jzheaux/pluralsight-secure-coding-java-11>

<https://github.com/spring-projects/spring-security>

