University of Tartu

Institute of computer science

**CVE-2013-7259**

Neo4j exploit overview for course “Computer security”

Sergey Korneev

may 2015
Neo4j database

Neo4j is a graph database. It can be run as an embedded server on a jvm or as a standalone server just like postgres or oracle DB.
The database has got many functional features that help DB administrators and developers to maintain its runtime.
Among them is a web interface where users can manage the state of the database using cypher query language (This is neo4j's analogue for sql).
Another feature is a Neo4j shell for running Cypher queries. There is also a rest API meant for GET/POST requests. Neo4j also supports plugins. One of them is a gremlin plugin. This plugin can be used during exploit attack.

Using exploit via gremlin plugin

This exploit was can be executed since Neo4j 1.9.2 with a gremlin plugin. As for now it is not completely closed.
A gremlin plugin lets a user to execute some groovy scripts in order to query graphs. It can be done via REST like this.

curl -X POST http://localhost:7474/db/data/ext/GremlinPlugin/graphdb/execute_script -d '{//groovy script goes here}'

The exploit is based on the idea that user is running neo4j on the localhost. The gremlin's rest API with code execution is a feature and has not been removed by now. When gremlin is installed, an exploit becomes available.

This kind of POST request can be executed from a browser via javascript. (ajax request)

<script>
$(function()
{
    var url = 'http://localhost:7474/db/data/ext/GremlinPlugin/graphdb/execute_script';
    var data = '{"script":"//groovy payload here", "params": {}}';

    $.ajax{{
        type: "POST",
        url: url,
        data: data,
        contentType: "application/json",
        success: success
    }});
});
</script>
Should a user with an access to such localhost be attacked via Cross-site request forgery, an attacker can run any groovy script in neo4j runtime. Going further, the groovy script is not run in any sandbox, meaning the attacker can use java.lang.System.runtime.exec("") to execute any authorized commands on the host OS. He could start a listening netcad process (netcat -l 4444) granting him another vector of attack. Also possible to upload a virus or collect OS information. He can also vector a DDOS attack by executing some cpu heavy groovy script. It is also possible to query restricted db information not only via gremlin API but via neo4 java API. (Any groovy script can execute java code)

Since Neo4j is a jvm based database, the exploit is available for all platforms. It is currently not fixed since the fix would remove an important feature of neo4j.

**Traversal REST endpoints exploit**

Although not part of the cve, it was possible to attack neo4j via traversal REST endpoint. It is possible to run a javascript via Rhino runtime via request. Because in neo4j 1.9.M02 the Rhino became sandboxed, it didn't make to the cve and the exploit can be considered to be closed. This was more dangerous exploit that the previous one since no plugin was needed. The attack could be executed in the same manner as before, only the url would need to be different and javascript would replace groovy.

**Neo4j configuration**

Neo4j has got some out of the box configuration in order to make its study and deployment easier. Because of this server listen ports are pre-defined. This means that the out of the box configuration is more vulnerable to attacks since attacker will try them first. A bright side is a password policy: it is required to change a password before using REST api. (Some administrators don't bother and would default passwords if not for this feature.) Another bright side is that Neo4j Web server answers only requests from the localhost by default. But this does not fix above mentioned bug.

Nowadays mentioned exploit is known to the developers of gremlin and neo4j. Because of this warnings about remote scripting are present in gremlin and neo4j documentation. It is possible to tell that the exploit has been partially eliminated by such actions since users will be more cautious.
Fighting exploit

There are plenty ways to fight the exploit. All of them require user actions.

- The gremlin plugin page suggests to simply switch it off and use conventional cypher-query-lang. A partial solution since sometimes people would like to use gremlin.

- It is possible to use proxy servers with white lists as a front end to the running server. The proxy will simply hold dangerous and suspicious requests. The port for neo4j can be closed via iptables and open only for proxy. This way neo4j can communicate with the internet only via proxy. But this is not a 100% solution since some people would like to add /db/data/ext/GremlinPlugin/graphdb/execute_script to the white list. However, another benefit from this that it is possible to allow requests only from specific IPs via proxy configuration. NB: won’t work when browser, proxy and neo4j are on localhost.

- Another solution would be to use the same proxy with authentication. The proxy won’t pass the request from browser to the server and no harm would be done. NB: won’t work when browser, proxy and neo4j are on localhost.

- The easiest way to fight the exploit on the localhost is to switch the API ports in neo4j configuration. This can be done in conf/neo4j-server.properties:

  org.neo4j.server.webserver.port=${newporthere}

- It is also possible to use java security manager to forbid calling java.lang.System.runtime.exec("`). A SecurityException will be thrown in runtime during such a call. By default the security manager is disabled. It is possible to enable it and upload some policies for it via neo4j’s jvm start arguments

  -Djava.security.manager -Djava.security.policy=${URLToFile}

- Some common practices may also reduce the aftermath of attack. For example running neo4j under root in linux would let the attacker to delete every folder in the OS. Thus run neo4j under user with limited privileges.
• Reading neo4j documentation carefully and understanding applied actions will lower chances of attack

References

Neo4j home page http://neo4j.com/
cve exploit http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-7259
gremlin plugin home page http://neo4j-contrib.github.io/gremlin-plugin/
java security policy overview http://docs.oracle.com/javase/6/docs/technotes/guides/security/PolicyFiles.html
traversal endpoint exploit explanation http://blog.scrt.ch/2014/05/09/neo4j-enter-the-graphdb/