T-SQL static code analysis: Dynamically executing code is security-sensitive

3 minutes

Executing code dynamically is security sensitive. It has led in the past to the following vulnerabilities:

- CVE-2017-9807
- CVE-2017-9802

Some APIs enable the execution of dynamic code by providing it as strings at runtime. These APIs might be useful in some very specific meta-programming use-cases. However most of the time their use is frowned upon as they also increase the risk of Injected Code. Such attacks can either run on the server or in the client (exemple: XSS attack) and have a huge impact on an application's security.

Both `EXECUTE( ... )` and `EXEC( ... )` execute as a command the string passed as an argument. They are safe only if the argument is composed of constant character string expressions. But if the command string is dynamically built using external parameters, then it is considered very dangerous because executing a random string allows the execution of arbitrary code.

This rule marks for review each occurrence of `EXEC` and

`EXECUTE`. This rule does not detect code injections. It only highlights the use of APIs which should be used sparingly and very carefully. The goal is to guide security code reviews.

Ask Yourself Whether

- the executed code may come from an untrusted source and hasn't been sanitized.
- you really need to run code dynamically.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

The best solution is to not run code provided by an untrusted source. If you really need to build a command string using external parameters, you should use `EXEC sp_executesql` instead.

Do not try to create a blacklist of dangerous code. It is impossible to cover all attacks that way.

Sensitive Code Example

```
CREATE PROCEDURE USER_BY_EMAIL(@email
VARCHAR(255)) AS
BEGIN
  EXEC('USE AuthDB; SELECT id FROM user WHERE email =
''' + @email + ''' ;'); -- Sensitive: could inject code using @email
END
```

Compliant Solution

```
CREATE PROCEDURE USER_BY_EMAIL(@email
VARCHAR(255)) AS
```

```
BEGIN
  EXEC sp_executesql 'USE AuthDB; SELECT id FROM user
WHERE email = @user_email;',
                    '@user_email VARCHAR(255)',
                     @user_email = @email;
END
```

See

- OWASP Top 10 2017 Category A1 - Injection
- MITRE CWE-95 - Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')

Available In: