# CWE Detail – CWE-749

## Description

The product provides an Applications Programming Interface (API) or similar interface for interaction with external actors, but the interface includes a dangerous method or function that is not properly restricted.

## Extended Description

This weakness can lead to a wide variety of resultant weaknesses, depending on the behavior of the exposed method. It can apply to any number of technologies and approaches, such as ActiveX controls, Java functions, IOCTLs, and so on. The exposure can occur in a few different ways: The function/method was never intended to be exposed to outside actors. The function/method was only intended to be accessible to a limited set of actors, such as Internet-based access from a single web site.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2007-6382: arbitrary Java code execution via exposed method

**•** CVE-2007-1112: security tool ActiveX control allows download or upload of files

## Related Attack Patterns (CAPEC)

* CAPEC-500

## Modes of Introduction

**•** Architecture and Design: N/A

**•** Implementation: N/A

## Common Consequences

**•** Impact: Gain Privileges or Assume Identity, Read Application Data, Modify Application Data, Execute Unauthorized Code or Commands, Other — Notes: Exposing critical functionality essentially provides an attacker with the privilege level of the exposed functionality. This could result in the modification or exposure of sensitive data or possibly even execution of arbitrary code.

## Potential Mitigations

**•** Architecture and Design: If you must expose a method, make sure to perform input validation on all arguments, limit access to authorized parties, and protect against all possible vulnerabilities. (Effectiveness: N/A)

**•** Architecture and Design: Identify all exposed functionality. Explicitly list all functionality that must be exposed to some user or set of users. Identify which functionality may be: accessible to all users restricted to a small set of privileged users prevented from being directly accessible at all Ensure that the implemented code follows these expectations. This includes setting the appropriate access modifiers where applicable (public, private, protected, etc.) or not marking ActiveX controls safe-for-scripting. (Effectiveness: N/A)

## Applicable Platforms

**•** None (Class: Not Language-Specific, Prevalence: Undetermined)

## Demonstrative Examples

**•** The method in this example is declared public and therefore is exposed to any class in the application. Deleting a database should be considered a critical operation within an application and access to this potentially dangerous method should be restricted. Within Java this can be accomplished simply by declaring the method private thereby exposing it only to the enclosing class as in the following example.

**•** A call into native code can then be initiated by passing parameters within the URL:

**•** Before Android 4.2 all methods, including inherited ones, are exposed to Javascript when using addJavascriptInterface(). This means that a malicious website loaded within this WebView can use reflection to acquire a reference to arbitrary Java objects. This will allow the website code to perform any action the parent application is authorized to.

**•** This code is not vulnerable to the above attack, but still may expose user info to malicious pages loaded in the WebView. Even malicious iframes loaded within a trusted page may access the exposed interface:

## Notes

**•** Research Gap: Under-reported and under-studied. This weakness could appear in any technology, language, or framework that allows the programmer to provide a functional interface to external parties, but it is not heavily reported. In 2007, CVE began showing a notable increase in reports of exposed method vulnerabilities in ActiveX applications, as well as IOCTL access to OS-level resources. These weaknesses have been documented for Java applications in various secure programming sources, but there are few reports in CVE, which suggests limited awareness in most parts of the vulnerability research community.