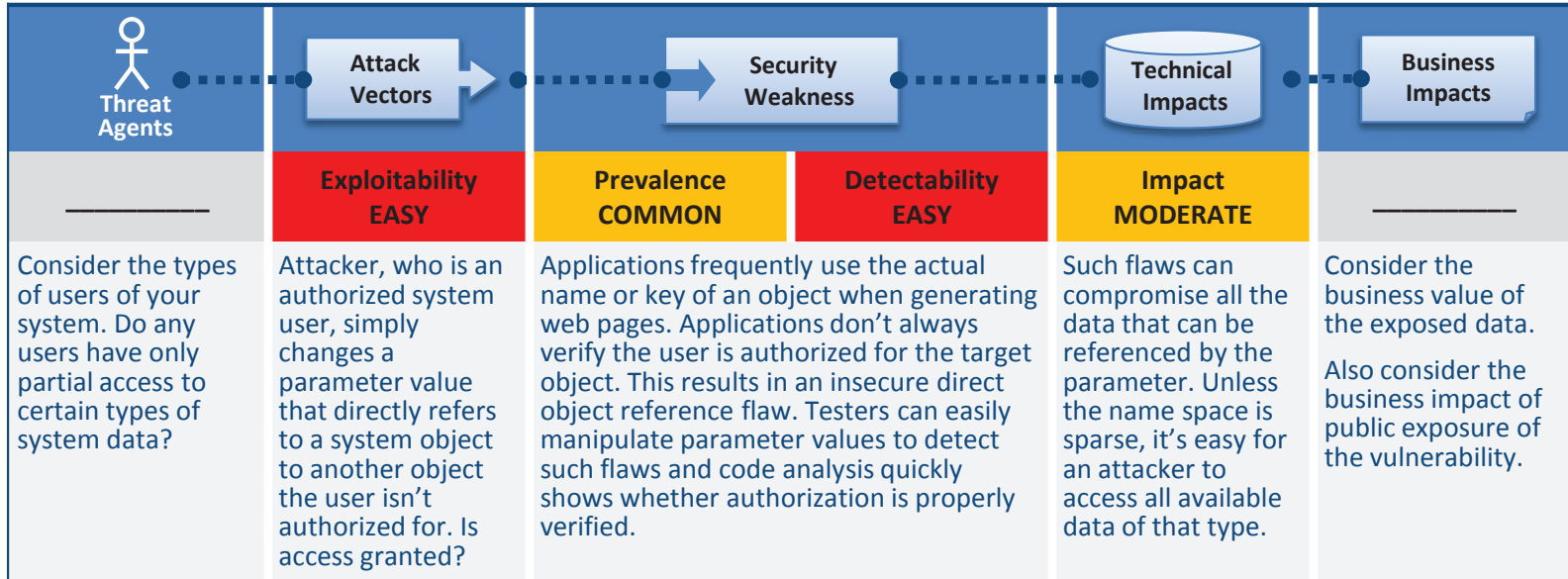


# A4

# Insecure Direct Object References



## Am I Vulnerable?

The best way to find out if an application is vulnerable to insecure direct object references is to verify that all object references have appropriate defenses. To achieve this, consider:

1. For **direct** references to **restricted** resources, the application needs to verify the user is authorized to access the exact resource they have requested.
2. If the reference is an **indirect** reference, the mapping to the direct reference must be limited to values authorized for the current user.

Code review of the application can quickly verify whether either approach is implemented safely. Testing is also effective for identifying direct object references and whether they are safe. Automated tools typically do not look for such flaws because they cannot recognize what requires protection or what is safe or unsafe.

## How Do I Prevent This?

Preventing insecure direct object references requires selecting an approach for protecting each user accessible object (e.g., object number, filename):

1. **Use per user or session indirect object references.** This prevents attackers from directly targeting unauthorized resources. For example, instead of using the resource's database key, a drop down list of six resources authorized for the current user could use the numbers 1 to 6 to indicate which value the user selected. The application has to map the per-user indirect reference back to the actual database key on the server. OWASP's [ESAPI](#) includes both sequential and random access reference maps that developers can use to eliminate direct object references.
2. **Check access.** Each use of a direct object reference from an untrusted source must include an access control check to ensure the user is authorized for the requested object.

## Example Attack Scenario

The application uses unverified data in a SQL call that is accessing account information:

```
String query = "SELECT * FROM accts WHERE account = ?";
PreparedStatement pstmt =
connection.prepareStatement(query, ... );
pstmt.setString( 1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

The attacker simply modifies the 'acct' parameter in their browser to send whatever account number they want. If not verified, the attacker can access any user's account, instead of only the intended customer's account.

<http://example.com/app/accountInfo?acct=notmyacct>

## References

### OWASP

- [OWASP Top 10-2007 on Insecure Dir Object References](#)
- [ESAPI Access Reference Map API](#)
- [ESAPI Access Control API \(See isAuthorizedForData\(\), isAuthorizedForFile\(\), isAuthorizedForFunction\(\)\)](#)

For additional access control requirements, see the [ASVS requirements area for Access Control \(V4\)](#).

### External

- [CWE Entry 639 on Insecure Direct Object References](#)
- [CWE Entry 22 on Path Traversal](#) (which is an example of a Direct Object Reference attack)