

# M183

## Insecure Direct Object Reference

Timo Bonomelli, Patrick Günthard

February 19, 2016

# Table of Contents

## Vulnerability

Threats

Example

## How to prevent an Attack

Session Based

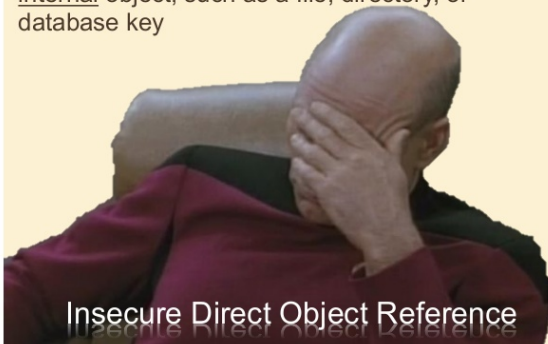
Check Authorization on every access

Solutions and Problems

## Summary

## What is *Insecure Direct Object References*

When a developer exposes a reference to an internal object, such as a file, directory, or database key



Insecure Direct Object Reference

# Threats

- ▶ **Threat Agents:** Any user who has only partial access to certain type of system data
- ▶ **Attacker's Approach:** Attacker, an authorized system user, simply changes a parameter value that directly refers to a system object to another object the user isn't authorized to use
- ▶ **Security Weakness:** Applications don't always verify the user is authorized for target objects

## Example: Code

### *Example Website:*

...

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

...

2010 OWASP - CC-BY-SA

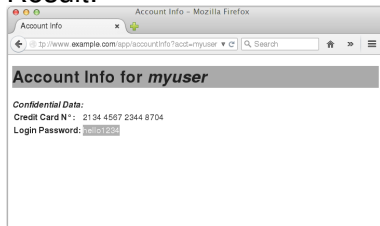
## Example: Attack

### Normal behavior

Example URL:

`http://example.com/app/  
accountInfo?acct=myacct`

Result:

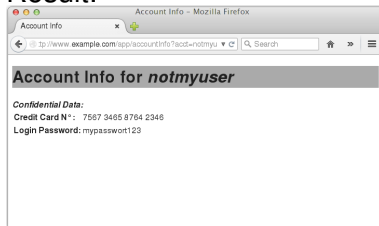


### Attack behavior

Example URL:

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

Result:



## Example: Attack

This URL:

`app/accountInfo?acct=myacct'; DROP accts; (1)`

# Table of Contents

## Vulnerability

Threats

Example

## How to prevent an Attack

Session Based

Check Authorization on every access

Solutions and Problems

## Summary



# Session

- ▶ No *Direct Object Reference* has to be sent to the client, the references can be saved on the session
- ▶ In the case references are needed, they can differ from the server side data (i.e. database) and can be remapped on the server

# Authorization

- ▶ Every access is checked if the user is authorized to do that. Example: A random token can be created for each user which then is checked every time the user accesses the page

# Solutions and Problems

|                      | Advantage   | Disadvantage  |
|----------------------|---|---|
| <b>Session Based</b> | Only one authorization has to be done, access data for Database etc. is saved on the server and is not accessible by the attacker | A session uses a lot of memory for each user. For applications with a high number of users, a session for each client is not possible i.e. a non-session solution has to be implemented |
| <b>Authorization</b> | No Session is needed i.e. less memory is used and more users can access the application   | Authorization is needed every time the user accesses data which is more complex to implement  |

# Table of Contents

## Vulnerability

Threats

Example

## How to prevent an Attack

Session Based

Check Authorization on every access

Solutions and Problems

## Summary

# Summary

- ▶ Serious Issue
- ▶ Easily preventable
- ▶ Several fixing solutions

# End

## Sources:

- ▶ OWASP: 2010-A4-Insecure Direct Object References
- ▶ Wikipedia: HDIV

