Action 36-09 JSF AV 138 propose a placement for this issue, if it is a vulnerability

JSF AV 138 is:

> Identifiers **shall not** simultaneously have both internal and external linkage in the same translation unit.
>
> **Rationale:** Avoid variable-name hiding which can be confusing.

Expanded in the annex to:

> The C++ Standard [**Error! Reference source not found.**] defines linkage in the following way:
>
> - When a name has external linkage, the entity it denotes can be referred to by names from scopes of other translation units or from other scopes of the same translation unit.
> - When a name has internal linkage, the entity it denotes can be referred to by names from other scopes in the same translation unit.
>
> Hence, having names with both internal and external linkage can be confusing since the objects to which they actually refer may not be obvious. Consider the following example where the *i* declared on line 1 has internal linkage while the *i* on line 2 has external linkage. Which entity is referenced by the *i* on line 3?

```
{  static int32 i=1;            // line 1
   {                            // Bad: the i with external linkage hides the i
                                //        with internal linkage.
      extern int32 i;           // line 2
      ...
      a[i] = 10;                // line 3: Confusing: which i?
   }
}
```

Proposal

This should be a vulnerability, because it is confusing and may lead to program's silently misbehaving. The likely candidate positions for it in our issues are:

- Choice of Clear Names [NAI] The issue essentially comes about because the user has chosen to use the same name for both an object local to one translation unit and accessed in another. However, this is really more of a special case of [YOW] – name reuse
- Identifier Name Reuse [YOW] The confusion comes about because of name reuse
- Namespace Issues [BJL] You could consider the internal and externally linkable spaces as separate namespaces, but then you'd be able to refer to both i's in the same scope – which you can't – so this is probably inappropriate
- Obscure Language Features [BRS] You might try to argue that this is an obscure language feature – but it would appear pretty clear that the 'i' on line 3 is the external reference (this is certainly how Visual Studio interprets it)

The answer would appear to be   [YOW] Identifier Name Reuse