From: "Joyce L Tokar" <tokar@pyrrhusoft.com>
Subject: **comments on the C annex of the Vulnerabilities TR**
Date: 29 September 2011 8:17:14 PM ET
To: "John Benito" <benito@bluepilot.com>, "Jim Moore" <James.W.Moore@ieee.org>
Cc: <brad.moore@shaw.ca>

Comments on C Annex

C.1 This section only speaks of converting between numeric signed integer types of different sizes, it doesn't mention how C implicitly converts between enumeration values, characters, float, addresses. Probably one of the biggest problems in this area relates to how enumeration literals are essentially treated as integers. I object to the C claim that it can be considered to be a strongly typed language, which suggests that it is in the same realm as Ada. You can't for example create two separate floating point types for Celsius and Farenheit, and expect the compiler to reject compilations that assign Celsius values to Farenheit values. Maybe there should be a term, moderatly- typed language if C is more strongly typed than other languages.

C.18 I object to the claim that Sign Extension errors cant happen in C. I'm sure there are millions of C programs that have vulnerabilities due to this error.

C.20 I object to the claim that dead stores cannot occur in C.

C.23 I object to the claim that namespace issues cannot occur in C.
Each C header is essentially a separate name space (even though it is not named)
The following illustrates this vulnerability. It compiles in gcc with no warnings.

```
#include <stdio.h>
#include "nm1.h"
#include "nm2.h"
int main (int argc, char **argv)
{
int status = SUCCESS;
printf ("Status=%d\n", status);
return status;
}
```

nm1.h ----------
```
enum {SUCCESS = 0, FAILURE = -1};
```

nm2.h ----------
```
#define SUCCESS -1
```

C.25 The text implies that C doesn't have any issues with logical operator precedence. Yes the rules are clearly defined, but the rules tend to be poorly understood due to there being a higher number of precedences.