```
From 08dd37c35ec5e64e26aacb8514437f54708f7fd1 Mon Sep 17 00:00:00 2001
From: Toddr Bot <toddbot@rinaldo.us>
Date: Mon, 16 Mar 2026 22:16:11 +0000
Subject: [PATCH] fix: off-by-one heap buffer overflow in st_serial_stack
 growth check

When st_serial_stackptr == st_serial_stacksize - 1, the old check
(stackptr >= stacksize) would not trigger reallocation. The subsequent
++stackptr then writes at index stacksize, one element past the
allocated buffer.

Fix by checking stackptr + 1 >= stacksize so the buffer is grown
before the pre-increment write.

Add a deep nesting test (600 levels) to exercise this code path.

Fixes #39

Co-Authored-By: Claude Opus 4.6 <noreply@anthropic.com>
---
 Expat/Expat.xs   |  2 +-
 t/deep_nesting.t | 22 ++++++++++++++++++++
 2 files changed, 23 insertions(+), 1 deletion(-)
 create mode 100644 t/deep_nesting.t

diff --git a/Expat/Expat.xs b/Expat/Expat.xs
index 5f9b193..0226a24 100644
--- a/Expat/Expat.xs
+++ b/Expat/Expat.xs
@@ -514,7 +514,7 @@ startElement(void *userData, const char *name, const char **atts)
     }
   }

-  if (cbv->st_serial_stackptr >= cbv->st_serial_stacksize) {
+  if (cbv->st_serial_stackptr + 1 >= cbv->st_serial_stacksize) {
     unsigned int newsize = cbv->st_serial_stacksize + 512;

     Renew(cbv->st_serial_stack, newsize, unsigned int);
diff --git a/t/deep_nesting.t b/t/deep_nesting.t
new file mode 100644
index 0000000..8237b5f
--- /dev/null
+++ b/t/deep_nesting.t
@@ -0,0 +1,22 @@
+BEGIN { print "1..1\n"; }
+
+# Test for deeply nested elements to exercise st_serial_stack reallocation.
+# This catches off-by-one errors in the stack growth check (GH #39).
+
+use XML::Parser;
+
+my $depth = 600;
+
+my $xml = '';
+for my $i (1 .. $depth) {
+    $xml .= "<e$i>";
+}
+for my $i (reverse 1 .. $depth) {
+    $xml .= "</e$i>";
+}
+
+my $p = XML::Parser->new;
+eval { $p->parse($xml) };
+
```

```
+print "not " if $@;
+print "ok 1\n";
```