# PCRE SOS Fund Audit Fix Log

This document lists all of the issues found by the Cure53 audit of PCRE2, and the steps taken to fix them. The fixes were made by Philip Hazel and Zoltán Herczeg of PCRE, and validated by Cure53. All fixes were verified against the respective references in the document and re-verified against the latest available revision from the repository, which was 522 at the time. Any commentary is from the PCRE team except where noted. They are presented in the same order as the audit report.

All of the fixed issues except PCRE-01-028 are fixed in PCRE2 version 10.21, the latest stable release.

## Identified Vulnerabilities

#### PCRE-01-003 Mutex Allocation in allocator_grab_lock() is racy (Medium)

Fix: The JIT maintainer is not sure how to fix this without abort()ing, which the library maintainers feel is inappropriate for a library.

**NOT VERIFIED**

Cure53 validation comment: *"abort()ing is not the issue here, the race condition and error delegation are. We strongly suggest using the implementation from the original report and finding a way to delegate the error condition to the respective caller. We do acknowledge that delegating the error condition up the call stack may be a rather difficult goal to achieve."*

#### PCRE-01-005 Buffer Overflow on Stack during Length Estimation Pass (Critical)

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_compile.c?r1=393&r2=395

**VERIFIED**

#### PCRE-01-008 Buffer Overflow when Handling large Offsets in regerror() (Medium)

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2posix.c?r1=386&r2=399

**VERIFIED**

Not only was there the possibility of a buffer overflow, but also it was not yielding the correct result, which should be the size of buffer needed to hold the full error message. I didn't take up the suggestion of using a short message; instead I just let snprintf() truncate the message as this seems to fit in with the specification of regerror(). [Historical note: PCRE originally stuck to C90 - as it was written in 1997 - which is why snprintf() was not previously used. Nowadays I'm happy to use C99 features.]

**PCRE-01-009 Incorrect Replacement Length Handling in pcre2_substitute() (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_substitute.c?r1=388&r2=400

**VERIFIED**

Set replace length before UTF check.

**PCRE-01-010 pcre2_substitute() has quadratic Runtime in UTF Mode (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_substitute.c?r1=400&r2=401

**VERIFIED**

Use PCRE2_NO_UTF_CHECK in pcre2_substitute().

**PCRE-01-012 Uninit Stack Read in pcre2_substitute() (Medium)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_substitute.c?r1=401&r2=402

**VERIFIED**

Check pcre2_substitute() group number.

**PCRE-01-013 Out-of-bounds Read behind replacement in pcre2_substitute() (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_substitute.c?r1=402&r2=405

**VERIFIED**

**PCRE-01-015 Unsafe out-of-bounds Pointer UTF-tested in pcre2_match() (Medium)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_match.c?r1=376&r2=406

**VERIFIED**

**PCRE-01-017 2nd find_fixedlength Result not stored as max_lookbehind (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_compile.c?r1=404&r2=407

**VERIFIED**

**PCRE-01-018 Problematic Truncation of max_lookbehind (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_compile.c?r1=407&r2=408

**PARTIALLY VERIFIED**

Cure53 validation comment: *"While the actual code has been fixed, the respective explanatory return code comments are slightly wrong as well as partially incomplete."*

**PCRE-01-023 Match Start after End causes pcre2_substitute to repeat Input (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_substitute.c?r1=405&r2=409

**VERIFIED**

**PCRE-01-024 Simultaneous Freeing of unserialized Patterns is racy (Medium)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/doc/pcre2serialize.3?r1=185&r2=410

**VERIFIED**

Documented pcre2_code_free() race issue in threads. I don't want to tangle with threads or locking code in the PCRE2 library, which restricts itself to Standard C library functions so that it will run easily in many different environments.

**PCRE-01-025 Invalid UTF in Substitution Output through int Overflow (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_compile.c?r1=408&r2=416

**VERIFIED**

There was a suggestion in the discussion of this issue that a maximum length should be imposed on pattern strings. I have implemented a new parameter that the application can specify for pcre2_compile() that provides such a limit. This gives flexibility for different applications, and avoids having to choose some arbitrary value for everybody. The default, however, is limited only by what PCRE2_SIZE (usually size_t) can hold.

**PCRE-01-026 Exponential Pattern Compilation Time using Subroutine Calls (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_compile.c?r1=416&r2=417
Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_compile.c?r1=420&r2=422

**VERIFIED**

Adding caching of information avoids exponential time. The caching of group information works well, but cannot be used for patterns that contain (?| because then multiple groups with the same number may have different characteristics. To stop runaway computation in this case, counters have been introduced to the functions that check for groups that can match the empty string and that find the fixed length of a group. In other words, I have implemented both suggestions, but for different cases.

**PCRE-01-028 Call to open_dev_zero() is racy (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/sljit/sljitUtils.c?r1=514&r2=513

**VERIFIED**

This one was fixed much more recently than the others.

## Miscellaneous Issues

**PCRE-01-001 Outdated Comments in pcre2_compile() (Info)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_compile.c?r1=418&r2=419

**VERIFIED**

**PCRE-01-002 Brittle Buffer Overflow Check in scan_for_captures() (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_compile.c?r1=419&r2=420

**VERIFIED**

The suggested fix had sizeof(PCRE2_SPTR) where sizeof(PCRE2_UCHAR) should have been used. I coded it in a slightly different way.

**PCRE-01-004 Lower bound in find_minlength() can be too high (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_study.c?r1=385&r2=421

**VERIFIED**

Cap minlength at 65535 and check for integer overflow.

**PCRE-01-006 Configuration of 16/32bit EBCDIC is not prevented (Info)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/configure.ac?r1=386&r2=425

**VERIFIED**

Lock out configuring for EBCDIC with non-8-bit libraries.

**PCRE-01-007 Inconsistent Error Handling in regerror() (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2posix.c?r1=399&r2=426

**VERIFIED**

Check error code is > 0 in regerror().

**PCRE-01-011 Return Value of pcre2_substitute() is ambiguous (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_substitute.c?r1=409&r2=427

**VERIFIED**

Limit replacements in pcre2_substitute() to INT_MAX. The suggested fix was just to document and maybe add a new function. I have implemented an error return if there are more than INT_MAX replacements. Even in an environment where ints are 16-bits wide, I would have thought that 32767 replacements would be enough, and surely with 32-bit ints (which I would guess most are these days) more than 2 billion replacements are plenty.

**PCRE-01-014 Dead Code causes use of wrong memctl in pcre2_match() (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_match.c?r1=372&r2=376

**VERIFIED**

This had already been fixed in version 10.20.

**PCRE-01-016 Out-of-bounds Pointer in non-UTF OP_REVERSE Handling (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_match.c?r1=406&r2=428

**VERIFIED**

**PCRE-01-019 Hardening: Abort on Memory Safety Error (Low)**

Fix: None made.

**NOT VERIFIED**

I take the points made, but I'm a bit cautious about including an explicit abort in a library. Deliberately killing the process when you have no idea what kind of process it is seems to me to be a bit drastic. Furthermore, most of these errors have never occurred. However, there have been bugs that provoked compile-time error 23, and the explicit message that it gives made it straightforward to figure out what was going on. A core dump (if one had happened) with no other information would have been much harder to debug (I haven't looked at a core dump in decades, not since I was an IBM Assembler programmer about 30 years ago). So for the moment at least, I have not done anything about this issue.

Cure53 validation comment: *"We acknowledge that abort()ing the process may be inadequate and unexpected behaviour for a library. We still suggest finding a way to delegate the error condition to the respective caller. We do acknowledge that delegating the error condition up the call stack may be a rather difficult goal to achieve."*

**PCRE-01-020 Missing NULL check in regcomp() (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2posix.c?r1=426&r2=429

**VERIFIED**

Check malloc failure in regcomp().

**PCRE-01-021 Unsafe Pointer Subtraction in DFA OP_REVERSE Matching (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_dfa_match.c?r1=372&r2=428

**VERIFIED**

**PCRE-01-022 Empty substitution match before CRLF handled weirdly (Info)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2_substitute.c?r1=427&r2=430

**VERIFIED**

The suggestion was better documentation if the behaviour was to be kept, but I have chosen not to keep this behaviour. Nevertheless, I have also updated the documentation.

**PCRE-01-027 Runtime Complexity Increase through global Substitution (Low)**

Fix: None implemented.

**NOT VERIFIED**

If any one match hits the bound, pcre2_substitute() gives an error response and gives up. I understand how matches that do not quite hit the bound could mount up, but I'm not sure if this is a very serious issue. Since pcre2-10.20 I have implemented a new feature after a discussion with users who wanted to track progress when searching large strings. You can now set an offset limit, which means that an error is given when no match can be found starting before that limit in the subject. This limit also applies when calling pcre2_substitute(), so it might be useful here. Apart from that, I don't know whether there are users who are concerned about time limits on substitutions - indeed, I don't know if any users are even using pcre2_substitute(). So I've left this as "something to think about".

**PCRE-01-029 Potential out-of-bounds array read in regcomp() (Low)**

Fix: http://vcs.pcre.org/pcre2/code/trunk/src/pcre2posix.c?r1=429&r2=432

**VERIFIED**

This check is really paranoid, but I found a neat way to code a fix.