
Subject: Patch to fix few possible issues

Posted by [Mindtraveller](#) on Mon, 25 Jan 2021 14:34:29 GMT

[View Forum Message](#) <> [Reply to Message](#)

hi! I've been working on possible security issues with U++ project and came across PVS-Studio as a nice static analysis tool. Since only U++ Core was used, it was processed by PVS-Studio and a number of potential security-related issues was found (collected in xlsx file). So here is a patch for U++ Core to fix possible issues found (looks like very few of those important).

File Attachments

1) [upp-pvs.xlsx](#), downloaded 247 times

2) [upp-core-pvs.patch](#), downloaded 252 times

Subject: Re: Patch to fix few possible issues

Posted by [mirek](#) on Thu, 28 Jan 2021 08:37:41 GMT

[View Forum Message](#) <> [Reply to Message](#)

Only one worth fixing is

Core/Vcont.h 312

Other than that, I am unwilling to change the code just to support a tool with such poor understanding of C++:

Core/Mem.h 316 ? V792 The 'Cmp128' function located to the right of the operator '&' will be called regardless of the value of the left operand. Perhaps, it is better to use '&&'.

Core/Mem.h 330 ? V792 The 'Cmp128' function located to the right of the operator '&' will be called regardless of the value of the left operand. Perhaps, it is better to use '&&'.

Core/Value.h 138 fixed V557 Array overrun is possible. The '2' index is pointing beyond array bound.

Core/Value.h 139 fixed V557 Array overrun is possible. The '2' index is pointing beyond array bound.

Core/z.cpp 243 not a bug V614 Uninitialized buffer 'h' used. Consider checking the first actual argument of the 'Poke32le' function.

On the positive note, I have fixed theide so that goto position now works with Copy of file/line part from the xlsx :)

Subject: Re: Patch to fix few possible issues

Posted by [koldo](#) on Fri, 29 Jan 2021 07:41:38 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi Mindtraveller

Thank you very much for this. Even detecting some or many false positives, if this tool could catch just one problem I would be happy.

I would be grateful if you could test the packages I manage in main U++ and in Bazaar. I promise to review all detected issues. Thank you.

Subject: Re: Patch to fix few possible issues

Posted by [Klugier](#) on Fri, 29 Jan 2021 10:30:06 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hello Koldo,

You could try to use flawfinder tool for scanning security issues in the C/C++ code. Sometimes, it produces false positives (Here is the ticket I raised to this tool), but the output are generally acceptable.

Here is the output for Function4U:

Toggle Spoiler

Flawfinder version 2.0.11, (C) 2001-2019 David A. Wheeler.

Number of rules (primarily dangerous function names) in C/C++ ruleset: 223

Examining Functions4U/GatherTpp.h

Examining Functions4U/Bsdiff/bspatch.cpp

Examining Functions4U/Bsdiff/bsadditional.cpp

Examining Functions4U/Bsdiff/bsdifff.cpp

Examining Functions4U/Functions4U_Gui.cpp

Examining Functions4U/GatherTpp.cpp

Examining Functions4U/bsdifff.h

Examining Functions4U/LocalProcess2.cpp

Examining Functions4U/StaticPlugin.cpp

Examining Functions4U/SvgColors.h

Examining Functions4U/Html/htmlid.h

Examining Functions4U/Html/htmlid.cpp

Examining Functions4U/LocalProcess2.h

Examining Functions4U/SvgColors.cpp

Examining Functions4U/Functions4U_Gui.h

Examining Functions4U/StaticPlugin.h

Examining Functions4U/QtEquation.cpp

Examining Functions4U/Functions4U.cpp

Examining Functions4U/Functions4U.h

FINAL RESULTS:

Functions4U/Functions4U.cpp:380: [5] (race) chmod:

This accepts filename arguments; if an attacker can move those files, a race condition results. (CWE-362). Use fchmod() instead.

Functions4U/Functions4U.cpp:129: [4] (shell) system:

This causes a new program to execute and is difficult to use safely

(CWE-78). try using a library call that implements the same functionality if available.

Functions4U/Functions4U.cpp:131: [4] (shell) system:

This causes a new program to execute and is difficult to use safely (CWE-78). try using a library call that implements the same functionality if available.

Functions4U/Functions4U.cpp:135: [4] (shell) system:

This causes a new program to execute and is difficult to use safely (CWE-78). try using a library call that implements the same functionality if available.

Functions4U/Functions4U.cpp:137: [4] (shell) system:

This causes a new program to execute and is difficult to use safely (CWE-78). try using a library call that implements the same functionality if available.

Functions4U/LocalProcess2.cpp:293: [4] (shell) execl:

This causes a new program to execute and is difficult to use safely (CWE-78). try using a library call that implements the same functionality if available.

Functions4U/LocalProcess2.cpp:331: [4] (shell) execv:

This causes a new program to execute and is difficult to use safely (CWE-78). try using a library call that implements the same functionality if available.

Functions4U/Functions4U.cpp:2577: [3] (misc) LoadLibraryEx:

Ensure that the full path to the library is specified, or current directory may be used (CWE-829, CWE-20). Use registry entry or GetWindowsDirectory to find library path, if you aren't already.

Functions4U/LocalProcess2.cpp:231: [3] (buffer) getenv:

Environment variables are untrustable input if they can be set by an attacker. They can have any content and length, and the same variable can be set more than once (CWE-807, CWE-20). Check environment variables carefully before using them.

Functions4U/Bsdiff/bsdiff.cpp:227: [2] (misc) open:

Check when opening files - can an attacker redirect it (via symlinks), force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents? (CWE-362).

Functions4U/Bsdiff/bsdiff.cpp:254: [2] (misc) open:

Check when opening files - can an attacker redirect it (via symlinks), force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents? (CWE-362).

Functions4U/Bsdiff/bsdiff.cpp:278: [2] (misc) fopen:

Check when opening files - can an attacker redirect it (via symlinks), force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents? (CWE-362).

Functions4U/Bsdiff/bsdiff.cpp:294: [2] (buffer) memcpy:

Does not check for buffer overflows when copying to destination (CWE-120).

Make sure destination can always hold the source data.

Functions4U/Bsdiff/bspatch.cpp:85: [2] (misc) fopen:

Check when opening files - can an attacker redirect it (via symlinks), force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents? (CWE-362).

Functions4U/Bsdiff/bspatch.cpp:128: [2] (misc) fopen:

Check when opening files - can an attacker redirect it (via symlinks), force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents? (CWE-362).

Functions4U/Bsdiff/bspatch.cpp:140: [2] (misc) fopen:

Check when opening files - can an attacker redirect it (via symlinks), force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents? (CWE-362).

Functions4U/Bsdiff/bspatch.cpp:152: [2] (misc) fopen:

Check when opening files - can an attacker redirect it (via symlinks), force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents? (CWE-362).

Functions4U/Bsdiff/bspatch.cpp:165: [2] (misc) open:

Check when opening files - can an attacker redirect it (via symlinks), force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents? (CWE-362).

Functions4U/Bsdiff/bspatch.cpp:237: [2] (misc) open:

Check when opening files - can an attacker redirect it (via symlinks), force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents? (CWE-362).

Functions4U/Functions4U.cpp:254: [2] (buffer) char:

Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119/CWE-120). Perform bounds checking, use functions that limit length, or ensure that the size is larger than the maximum possible length.

Functions4U/Functions4U.cpp:557: [2] (misc) open:

Check when opening files - can an attacker redirect it (via symlinks), force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents? (CWE-362).

Functions4U/Functions4U.cpp:566: [2] (buffer) char:

Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119/CWE-120). Perform bounds checking, use functions that limit length, or ensure that the size is larger than the maximum possible length.

Functions4U/Functions4U.cpp:578: [2] (misc) open:

Check when opening files - can an attacker redirect it (via symlinks),

force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents? (CWE-362).

Functions4U/Functions4U.cpp:638: [2] (buffer) char:

Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use functions that limit length, or ensure that the size is larger than the maximum possible length.

Functions4U/Functions4U.cpp:721: [2] (buffer) char:

Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use functions that limit length, or ensure that the size is larger than the maximum possible length.

Functions4U/Functions4U.cpp:727: [2] (buffer) char:

Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use functions that limit length, or ensure that the size is larger than the maximum possible length.

Functions4U/Functions4U.cpp:1719: [2] (misc) open:

Check when opening files - can an attacker redirect it (via symlinks), force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents? (CWE-362).

Functions4U/Functions4U.cpp:1727: [2] (misc) open:

Check when opening files - can an attacker redirect it (via symlinks), force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents? (CWE-362).

Functions4U/Functions4U.cpp:1769: [2] (misc) fopen:

Check when opening files - can an attacker redirect it (via symlinks), force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents? (CWE-362).

Functions4U/Functions4U.cpp:1981: [2] (misc) fopen:

Check when opening files - can an attacker redirect it (via symlinks), force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents? (CWE-362).

Functions4U/Functions4U.cpp:2024: [2] (misc) fopen:

Check when opening files - can an attacker redirect it (via symlinks), force the opening of special file type (e.g., device files), move things around to create a race condition, control its ancestors, or change its contents? (CWE-362).

Functions4U/Functions4U.cpp:2067: [2] (integer) atoi:

Unless checked, the resulting number can exceed the expected range (CWE-190). If source untrusted, check both minimum and maximum, even if the input had no minus sign (large numbers can roll over into negative number; consider saving to an unsigned value if that is intended).

Functions4U/Functions4U.cpp:2067: [2] (integer) atoi:
 Unless checked, the resulting number can exceed the expected range (CWE-190). If source untrusted, check both minimum and maximum, even if the input had no minus sign (large numbers can roll over into negative number; consider saving to an unsigned value if that is intended).

Functions4U/LocalProcess2.cpp:183: [2] (buffer) memcpy:
 Does not check for buffer overflows when copying to destination (CWE-120).
 Make sure destination can always hold the source data.

Functions4U/LocalProcess2.cpp:188: [2] (buffer) memcpy:
 Does not check for buffer overflows when copying to destination (CWE-120).
 Make sure destination can always hold the source data.

Functions4U/LocalProcess2.cpp:236: [2] (buffer) memcpy:
 Does not check for buffer overflows when copying to destination (CWE-120).
 Make sure destination can always hold the source data.

Functions4U/LocalProcess2.cpp:251: [2] (race) vfork:
 On some old systems, vfork() permits race conditions, and it's very difficult to use correctly (CWE-362). Use fork() instead.

Functions4U/LocalProcess2.cpp:533: [2] (buffer) char:
 Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119/CWE-120). Perform bounds checking, use functions that limit length, or ensure that the size is larger than the maximum possible length.

Functions4U/LocalProcess2.cpp:566: [2] (buffer) char:
 Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119/CWE-120). Perform bounds checking, use functions that limit length, or ensure that the size is larger than the maximum possible length.

Functions4U/Bsdiff/bsdiff.cpp:238: [1] (buffer) read:
 Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20).

Functions4U/Bsdiff/bsdiff.cpp:265: [1] (buffer) read:
 Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20).

Functions4U/Bsdiff/bspatch.cpp:174: [1] (buffer) read:
 Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20).

Functions4U/Functions4U.cpp:567: [1] (buffer) read:
 Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20).

Functions4U/Functions4U.cpp:593: [1] (buffer) read:
 Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20).

Functions4U/Functions4U.cpp:1534: [1] (buffer) strlen:
 Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).

Functions4U/Functions4U.cpp:1536: [1] (buffer) strlen:
 Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).

Functions4U/Functions4U.cpp:1738: [1] (buffer) read:
Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20).

Functions4U/Functions4U.cpp:1739: [1] (buffer) read:
Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20).

Functions4U/Functions4U.cpp:1781: [1] (buffer) fgetc:
Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20).

Functions4U/Functions4U.cpp:1984: [1] (buffer) fgetc:
Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20).

Functions4U/Functions4U.cpp:2031: [1] (buffer) fgetc:
Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20).

Functions4U/Functions4U.cpp:2342: [1] (buffer) equal:
Function does not check the second iterator for over-read conditions (CWE-126). This function is often discouraged by most C++ coding standards in favor of its safer alternatives provided since C++14. Consider using a form of this function that checks the second iterator before potentially overflowing it.

Functions4U/Functions4U.cpp:2698: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).

Functions4U/Functions4U.cpp:2699: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).

Functions4U/Functions4U.cpp:2727: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).

Functions4U/Functions4U.cpp:2728: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).

Functions4U/Functions4U.cpp:2764: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).

Functions4U/Functions4U.cpp:2765: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).

Functions4U/Functions4U.h:371: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).

Functions4U/LocalProcess2.cpp:176: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).

Functions4U/LocalProcess2.cpp:182: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).

Functions4U/LocalProcess2.cpp:193: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).
Functions4U/LocalProcess2.cpp:325: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).
Functions4U/LocalProcess2.cpp:567: [1] (buffer) read:
Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20).
Functions4U/QtfEquation.cpp:266: [1] (buffer) equal:
Function does not check the second iterator for over-read conditions (CWE-126). This function is often discouraged by most C++ coding standards in favor of its safer alternatives provided since C++14. Consider using a form of this function that checks the second iterator before potentially overflowing it.
Functions4U/bsdifff.h:19: [1] (buffer) read:
Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20).

ANALYSIS SUMMARY:

Hits = 66
Lines analyzed = 8458 in approximately 0.10 seconds (88475 lines/second)
Physical Source Lines of Code (SLOC) = 7076
Hits@level = [0] 3 [1] 27 [2] 30 [3] 2 [4] 6 [5] 1
Hits@level+ = [0+] 69 [1+] 66 [2+] 39 [3+] 9 [4+] 7 [5+] 1
Hits/KSLOC@level+ = [0+] 9.75127 [1+] 9.3273 [2+] 5.51159 [3+] 1.27191 [4+] 0.989259 [5+] 0.141323
Minimum risk level = 1
Not every hit is necessarily a security vulnerability.
There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(<https://dwheeler.com/secure-programs>) for more information.

On my distro (Manjaro) you can download flawfinder as a package.

Klugier

Subject: Re: Patch to fix few possible issues
Posted by [Oblivion](#) on Fri, 29 Jan 2021 11:07:06 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hello Koldo,

You can also use cppcheck.

I have attached the csv file of Functions4U.

Best regards,
Oblivion

File Attachments

1) [functions4u.results.csv](#), downloaded 266 times

Subject: Re: Patch to fix few possible issues
Posted by [koldo](#) on Sun, 31 Jan 2021 08:45:20 GMT
[View Forum Message](#) <> [Reply to Message](#)

Thank you all.

Subject: Re: Patch to fix few possible issues
Posted by [koldo](#) on Wed, 03 Feb 2021 08:22:30 GMT
[View Forum Message](#) <> [Reply to Message](#)

After reviewing both:

- flawfinder gives generic advices, e.g., if it finds a system() call, it advices to replace it with the specific OS function. Not bad, but generic
- cppcheck seems to really check the sources in detail, giving very specific advices

So cppcheck, although it sometimes fails, seems more useful.
