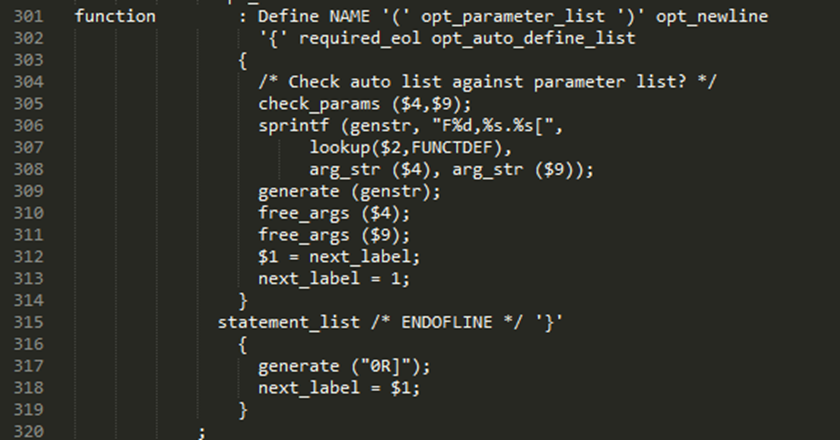
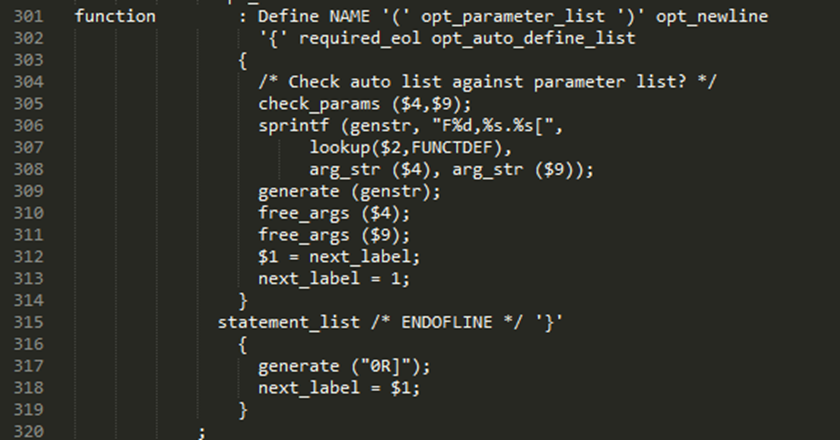
NO.1 buggy version

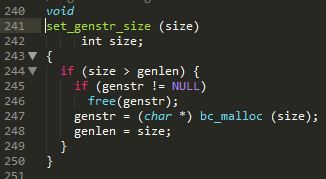


NO.1 fixed version

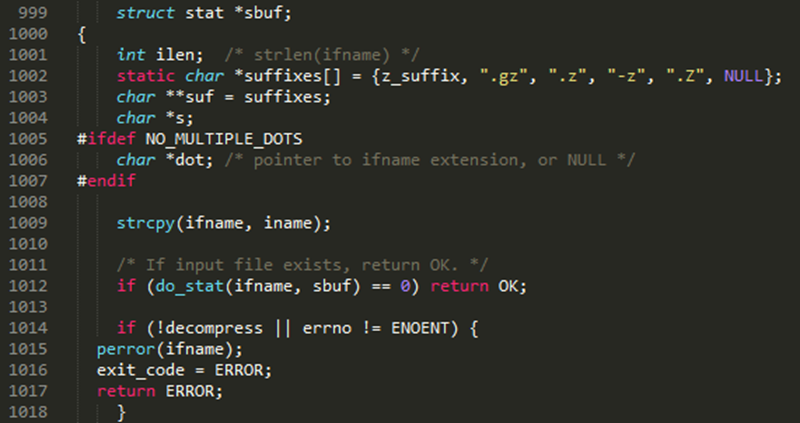


The first parameter of sprintf is not checked. Use function set\_genstr\_size to larger the buffer size.

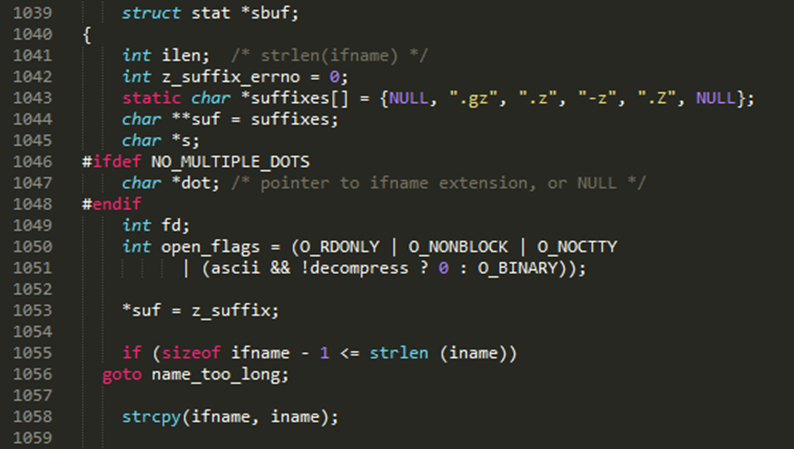
Here is the code of set\_genstr\_size

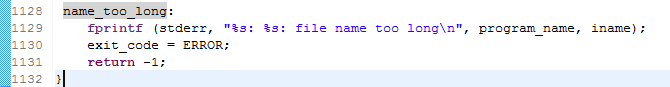


NO.2 buggy version



NO.2 fixed version





Buffer overflow takes place in strcpy(ifname,iname). ifname is defined as “char ifname[MAX\_PATH\_LEN]”, and iname is from function get\_istat, which is called in function treat\_file

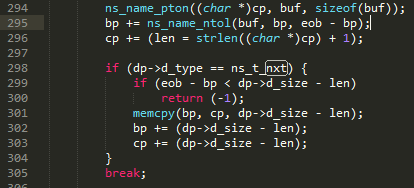


Finally, we get the place where treat\_file is called:

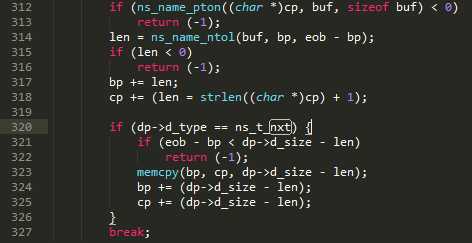


argv is from function main.

NO.3 buggy version

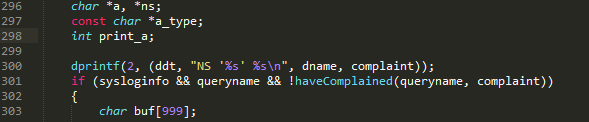


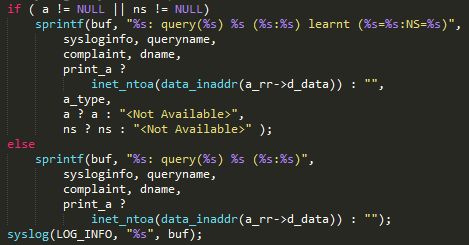
NO.3 fixed version



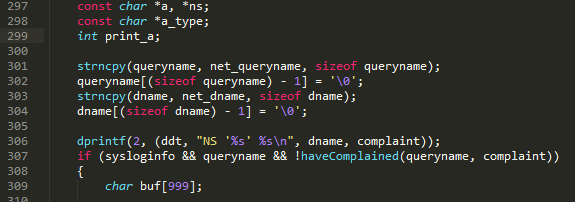
memcpy(bp, cp, dp->d\_size - len) leads to a buffer overflow. When ns\_name\_ntol reports an error, the return value is -1, so bp+=-1. memcpy goes wrong.

NO.5 buggy version



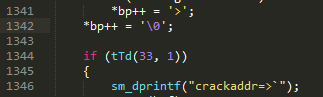


NO.5 fixed version

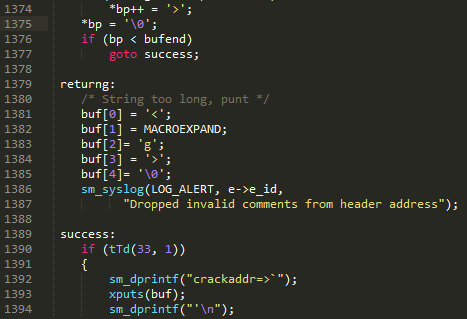


sprintf write queryname into buf, which is defined as “char buf[999]”. A large queryname will cause buffer overflow.

NO.6 buggy version



NO.6 fixed version

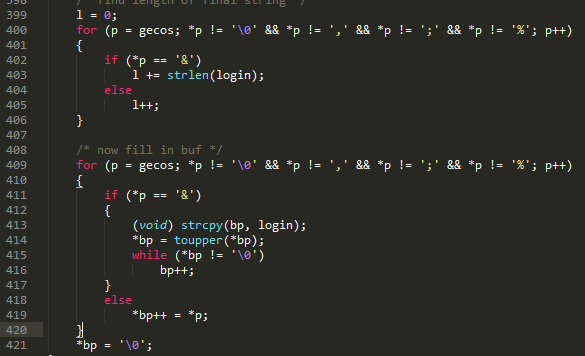


No boundary check for pointer bp causes a buffer overflow.

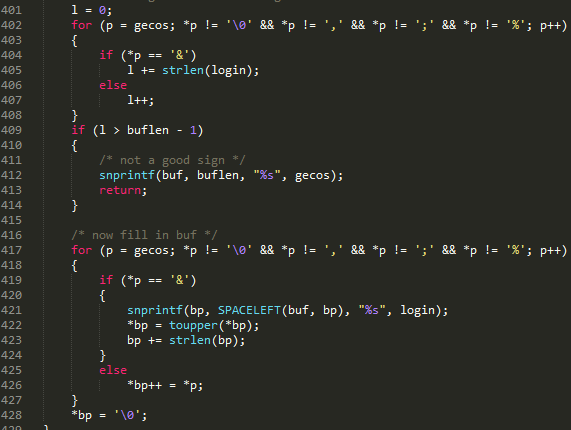


bp writes into buf once a bit. When bp exceeds bufend, a buffer overflow will happen.

NO.7 buggy version

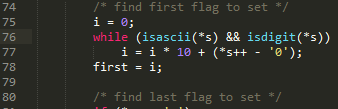


NO.7 fixed version



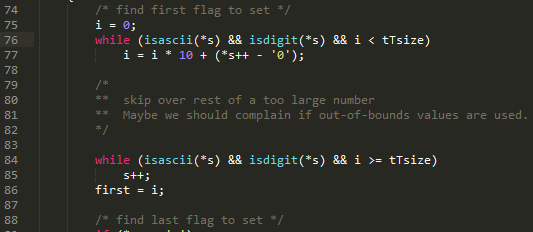
Buffer overflow happens in strcpy, which is caused by a large login.

NO.10 buggy version



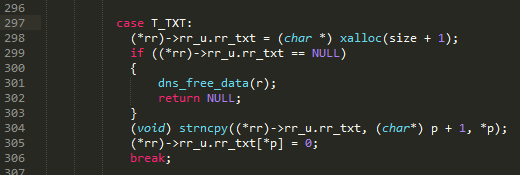


NO.10 fixed version

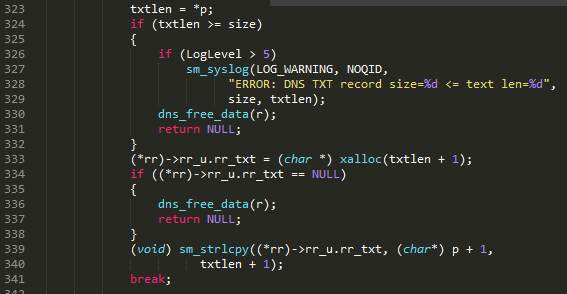


In the buggy version, first can become large due to the while loop. In the fixed version, a boundary check is added.

NO.11 buggy version

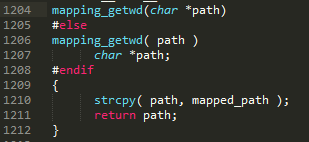


NO.11 fixed version



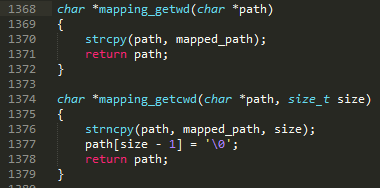
A buffer overflow happens in strncpy((\*rr)->rr\_u.rr\_txt, (char\*) p + 1, \*p). \*p may exceed the size of the first parameter. A boundary check is added in the fixed version.

NO.12 buggy version





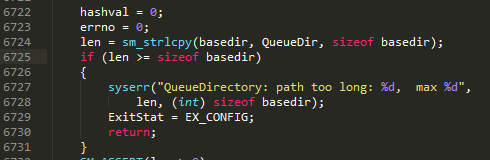
NO.12 fixed version





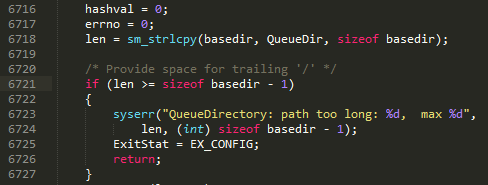
In the buggy version, getcwd is defined as mapping\_getwd, which is implemented with strcpy. In the fixed version, it’s changed to strncpy.

NO.14 buggy version



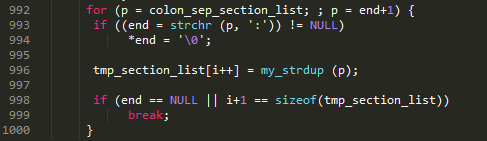


NO.14 fixed version

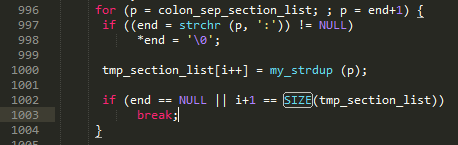


Off-by-one error in boundary check.

NO.15 buggy version



NO.15 fixed version





The array tmp\_section\_list leads to a buffer overflow, which is defined as:

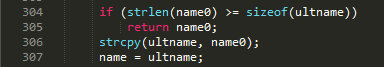


In the for loop, the condition for break is wrong.

NO.16 buggy version



NO.16 fixed version

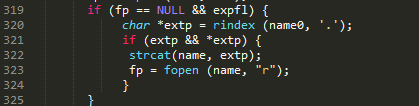


Buffer overflow happens in strcpy(ultname,name0). ultname is defined as:

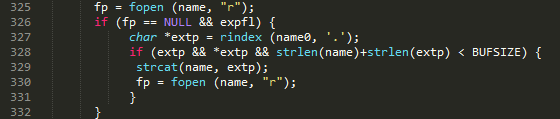


name0 is an input from user. A large name0 will exceed the size of ultname.

NO.17 buggy version



NO.17 fixed version



Buffer overflow happens in strcat(name,extp), name is defined as:



extp is defined as:



The function rindex returns a substring of name0 from the last ‘.’ to the end.

No boundary check leads to a overflow.

NO.18 buggy version



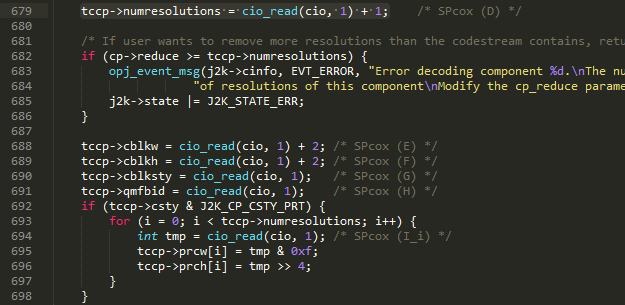
NO.18 fixed version



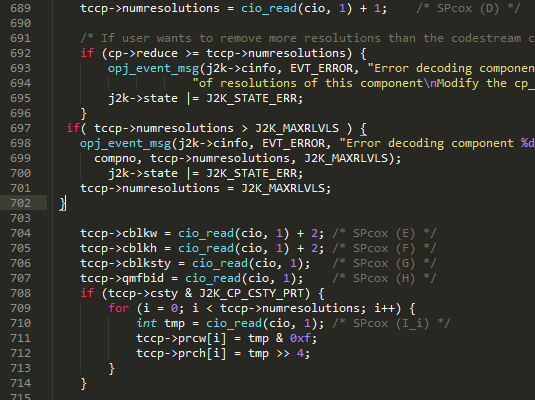


The char array wstr may not end with ‘\0’, so a buffer overflow may happen in sscanf.

NO.19 buggy version



NO.19 fixed version



The array tccp->prcw may overflow. In the for loop, the upper bound of i is tccp->numresolutions-1, which get its value from:



The array prcw is defined as



When tccp->numresolutions is greater than J2K\_MAXRLVLS, the array will overflow.

NO.20 buggy version

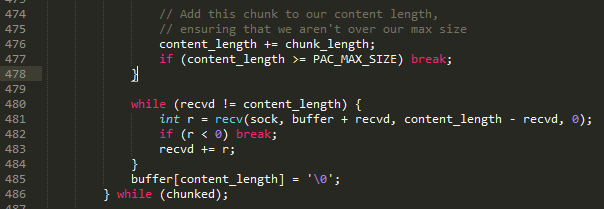


NO.20 fixed version

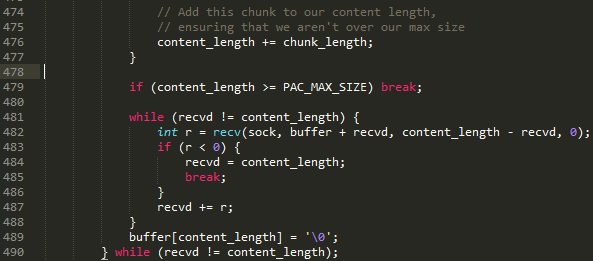


The array  will overflow due to off-by-one error in the boundary check.

NO.21 buggy version



NO.21 fixed version



This line: r = recv(sock, buffer + recvd, content\_length - recvd, 0) will overflow. The position of boundary check is not right.

NO.27 buggy version



NO.27 fixed version

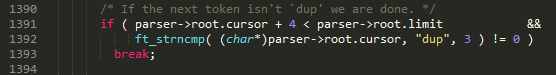


sp is a variable of type char\*. When \*sp=0, i.e., it reaches the string end, the sp++ should not be executed.

NO.30 buggy version

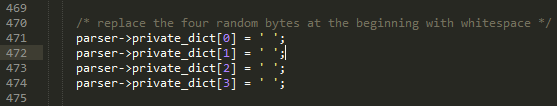


NO.30 fixed version

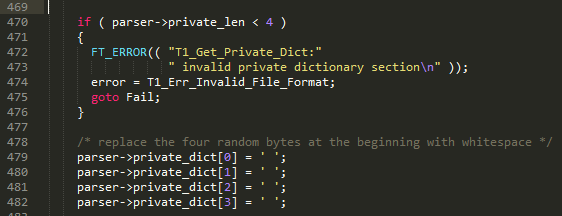


ft\_strncmp is the same as strncmp, which is used to compare two strings. Here n equals 3, so the parser->root.cursor will read 3 more bits, which may cause a buffer overflow.

NO.31 buggy version

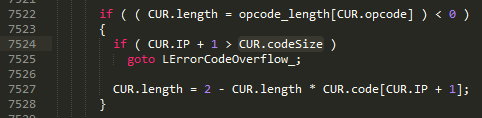


NO.31 fixed version

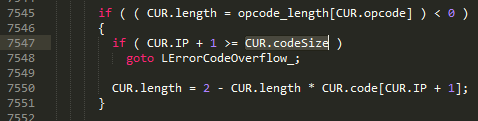


The array private\_dict’s size needs to be at least 4.

NO.32 buggy version

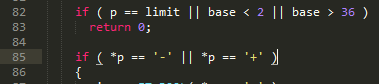


NO.32 fixed version

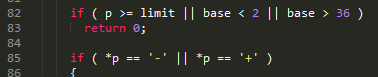


An off-by-one error in boundary check.

NO.33 buggy version

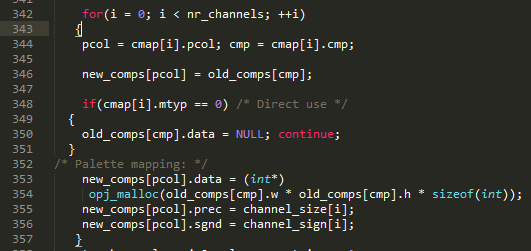


NO.33 fixed version

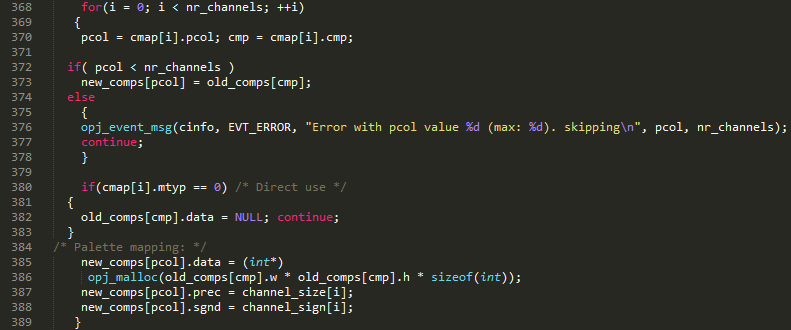


The boundary check in buggy version uses p==limit. However, p gets its value from outside and it may be greater than limit in the first place.

NO.34 buggy version



NO.34 fixed version

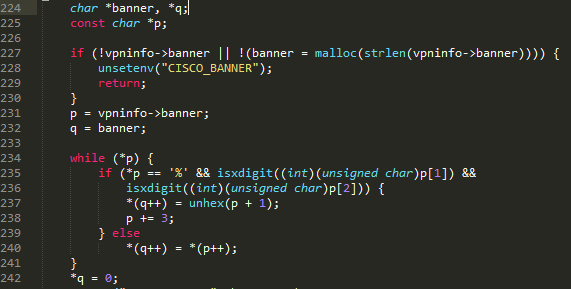


The array new\_comps gets its memory from:

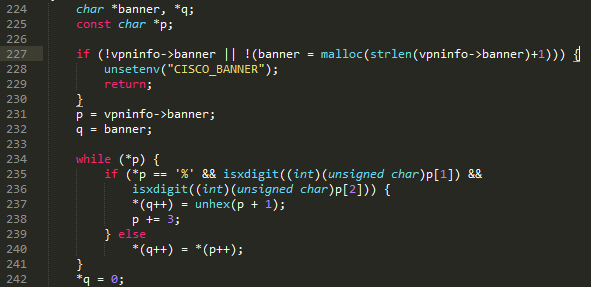


So it is obvious the size of this array is nr\_channels. In the fixed version, the index pcol is checked to make sure it’s smaller than nr\_channels.

NO.35 buggy version



NO.35 fixed version



Allocate more memory to banner

NO.38 buggy version



NO.38 fixed version



name+len may exceed the range of name

NO.39 buggy version

in line 852:

strlen\_buf = snprintf(output\_buf, MAX\_BUF, "%.\*s%.\*s" ERROR\_URL "%.\*s%s" REJECT\_CRLF, STRLEN\_REJECT\_SEVERITY, (substitute\_reject\_severity == NULL) ? target\_rejection->reject\_severity : substitute\_reject\_severity, target\_rejection->strlen\_reject\_message, target\_rejection->reject\_message, current\_settings->current\_options->strlen\_policy\_location, current\_settings->current\_options->policy\_location, target\_rejection->short\_reject\_message);

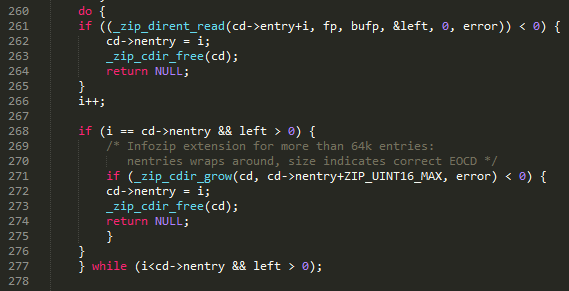
NO.39 fixed version

in line 852:

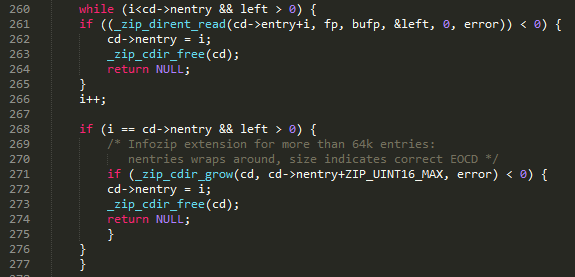
snprintf(output\_buf, MAX\_BUF, "%.\*s%.\*s" ERROR\_URL "%.\*s%s" REJECT\_CRLF "%n", STRLEN\_REJECT\_SEVERITY, (substitute\_reject\_severity == NULL) ? target\_rejection->reject\_severity : substitute\_reject\_severity, target\_rejection->strlen\_reject\_message, target\_rejection->reject\_message, current\_settings->current\_options->strlen\_policy\_location, current\_settings->current\_options->policy\_location, target\_rejection->short\_reject\_message, &strlen\_buf);

Using the return value of snprintf as the size of bits written into output\_buf is not right. Use “%n” instead.

NO.40 buggy version

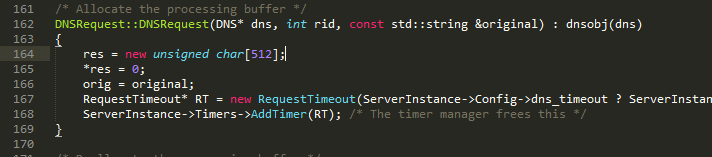


NO.40 fixed version



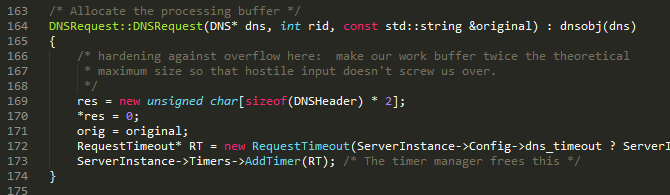
The use of do...while loop is not right, because it will execute the code once before checking the condition of while.

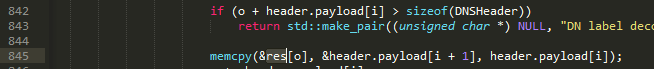
NO.41 buggy version





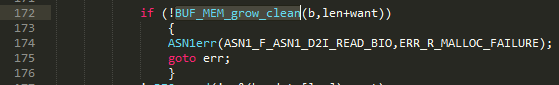
NO.41 fixed version

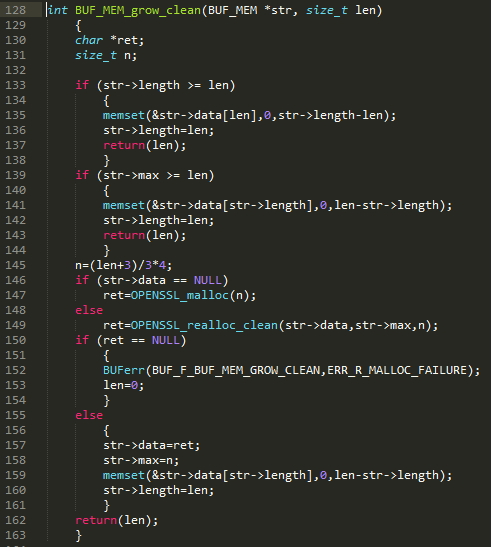




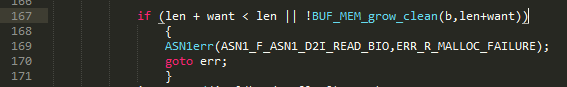
Buffer overflow happens in memcpy. A boundary check is needed. In addition, they make the buffer twice the theoretical maximum size.

NO.43 buggy version



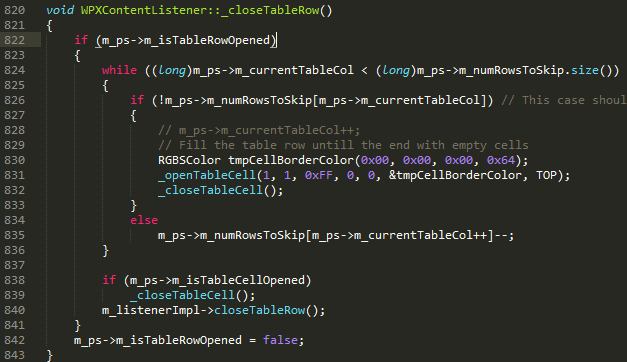


NO.43 fixed version

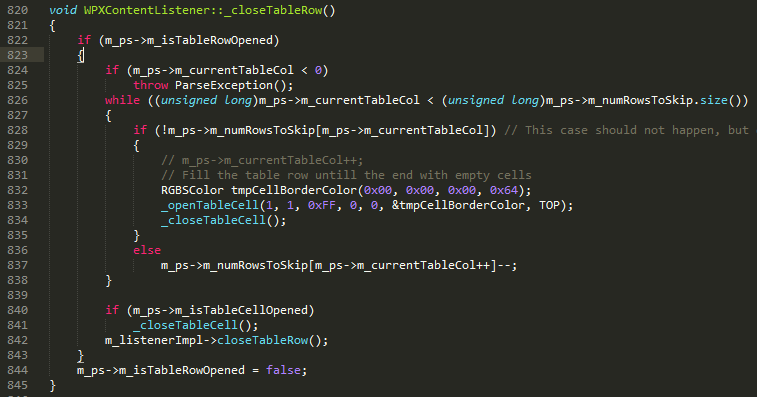


Integer overflow of len+want leads to a buffer overflow in memset in BUF\_MEM\_grow\_clean

NO.45 buggy version



NO.45 fixed version



The array m\_ps->m\_numRowsToSkip needs a boundary check.

NO.47 buggy version





NO.47 fixed version





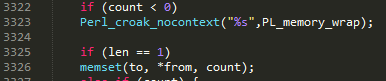


The function memcpy may overflow. The function g\_strndup from glib ensures the overflow won’t happen.

NO.48 buggy version

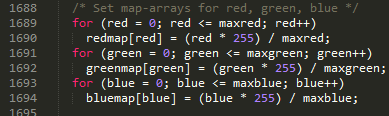


NO.48 fixed version

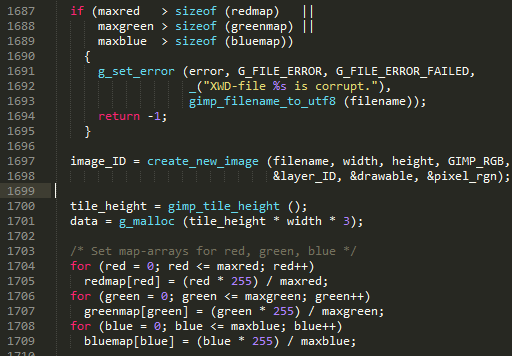


The function memset’s third parameter needs to be positive.

NO.49 buggy version

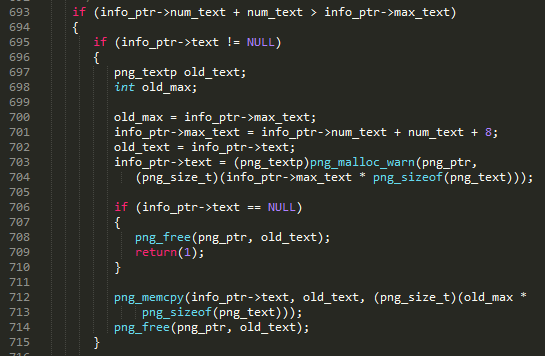


NO. 49 fixed version

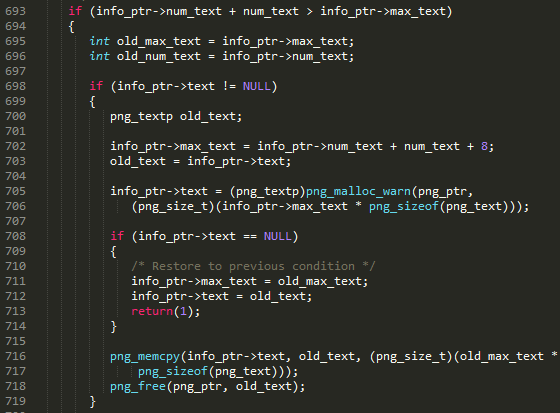


The access to the array redmap in the for loop needs a boundary check.

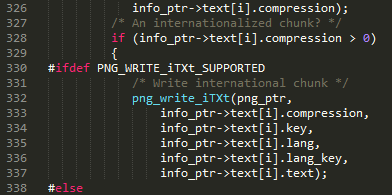
NO.51 buggy version



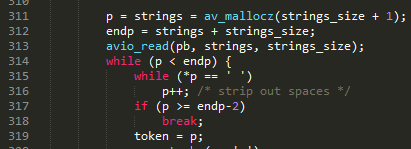
NO.51 fixed version



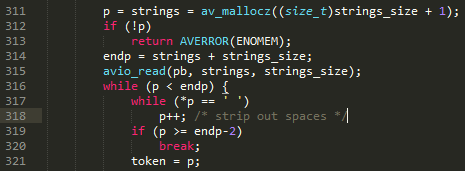
If the malloc fails, info\_ptr->text should be restored to previous value. The buggy version ignores this, so in the following picture a buffer overflow will happen.



NO.52 buggy version



NO.52 fixed version

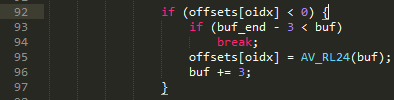


If malloc fails, p will be null. A overflow happens when assigning a value to \*p.

NO.53 buggy version



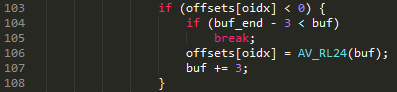




NO.53 fixed version

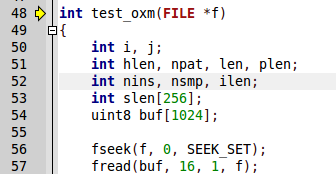


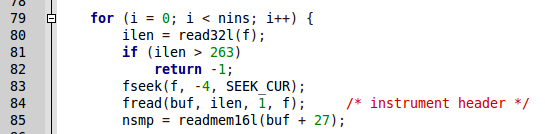




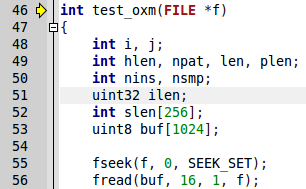
The value oidx may be 7. A large buffer is needed.

NO.54 buggy version



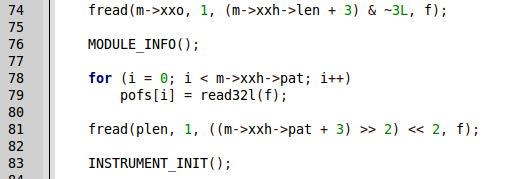


NO.54 fixed version

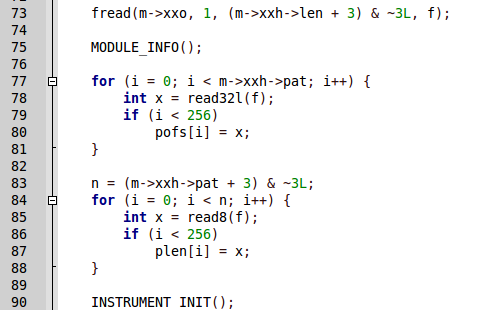


Setting the value of ilen to a negative number will bypass the boundary check.

NO.55 buggy version

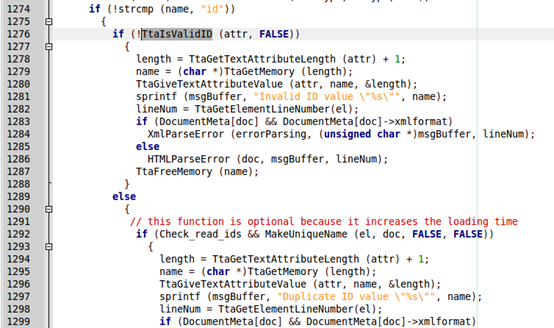


NO.55 fixed version

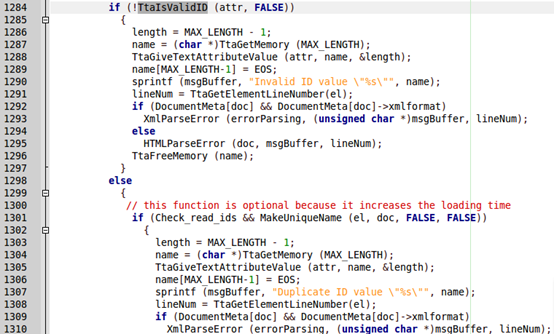


The array pofs needs a boundary check.

NO.56 buggy version

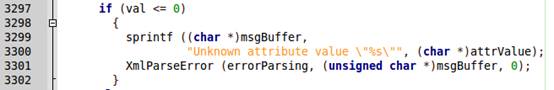


NO.56 fixed version

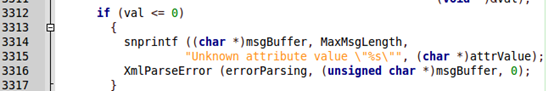


The function sprintf will not stop copying bits from name to msgBuffer until it reads a ‘\0’. A ‘\0’ is added to name.

NO.57 buggy version

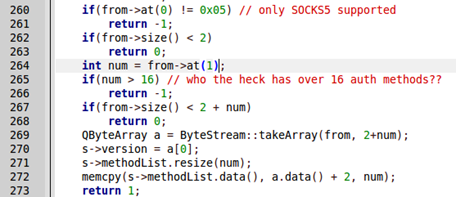


NO.57 fixed version

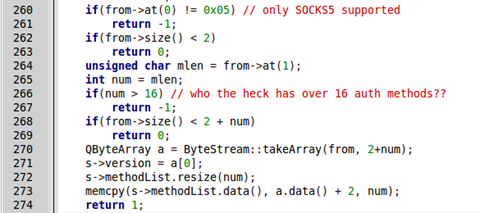


The length of attrValue may exceed the size of msgBuffer.

NO.58 buggy version

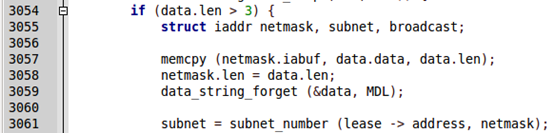


NO.58 fixed version

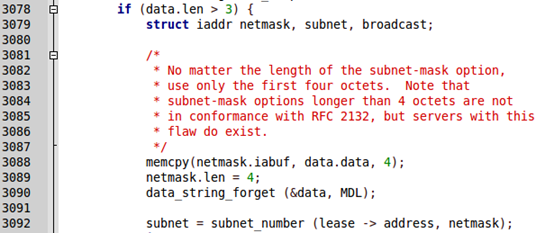


A negative number will bypass the boundary check.

NO.59 buggy version



NO.59 fixed version



A large data.len leads to a buffer overflow in memcpy. Setting the value to 4 can fix it.

NO.61 buggy version





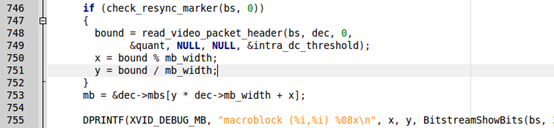
NO.61 fixed version



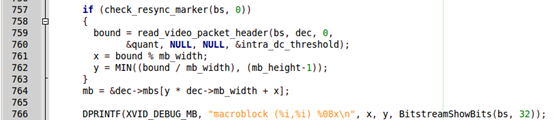


A large imagepath or name may lead to a buffer overflow in sprintf.

NO.62 buggy version



NO.62 fixed version



The value of y may cause an overflow.

NO.64 buggy version

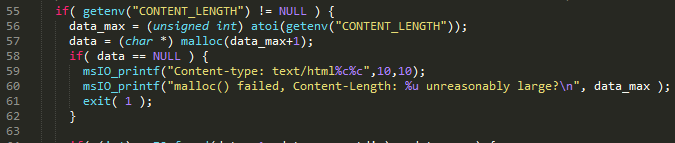


NO.64 fixed version



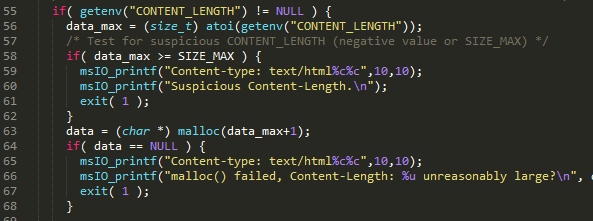
The function sprintf may cause a buffer overflow. Use snprintf instead.

NO.65 buggy version





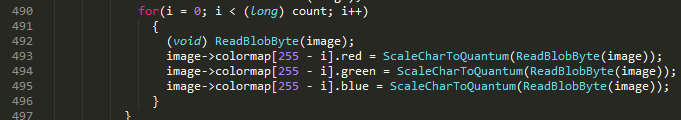
NO.65 fixed version



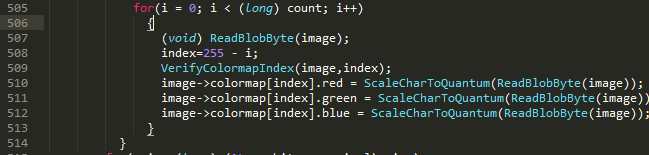


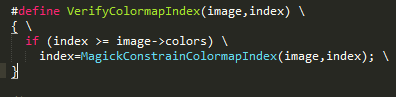
The parameter of the function malloc needs to be checked.

NO.66 buggy version



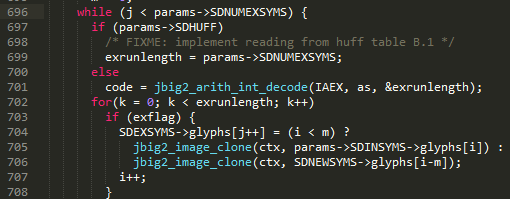
NO.66 fixed version



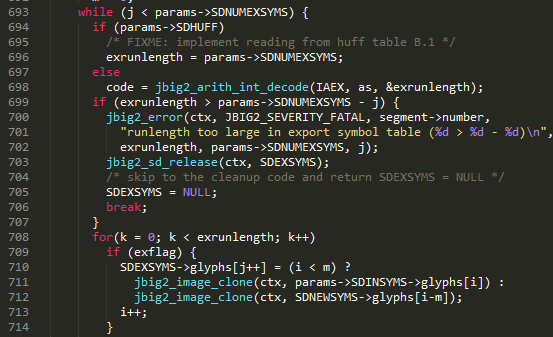


The array image->colormap may cause a buffer overflow.

NO.68 buggy version

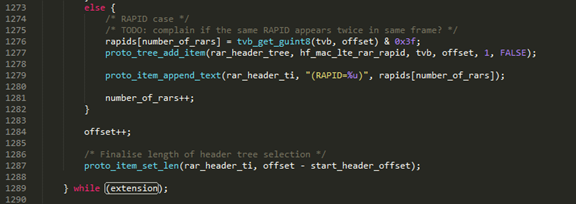


NO.68 fixed version



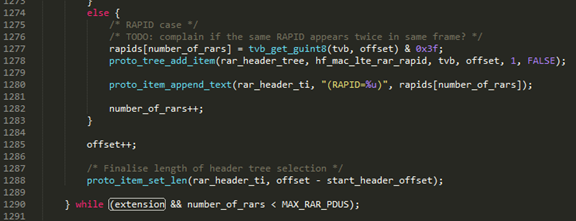
The index of SDEXSYMS->glyphs may reach to j+exrunlength-1. So a large exrunlength may cause a buffer overflow.

NO.69 buggy version





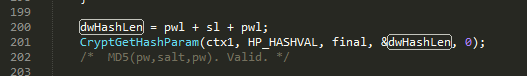
NO.69 fixed version





The array rapids may cause a buffer overflow.

NO.70 buggy version

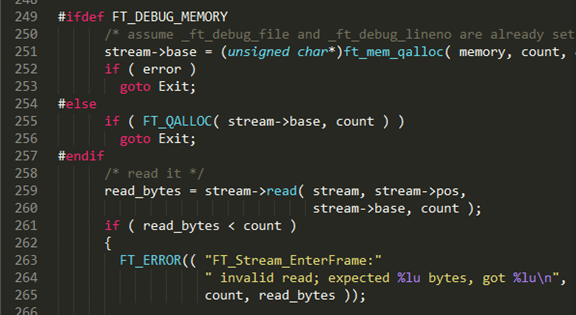


NO.70 fixed version

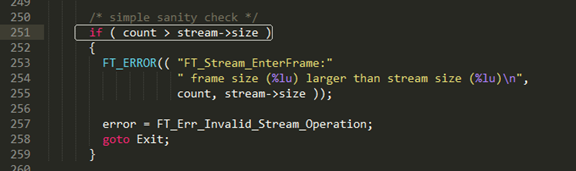


In different system, dwHashLen in the buggy version will get different value, which may cause a buffer overflow.

NO.71 buggy version



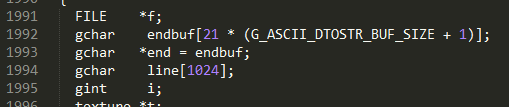
NO.71 fixed version



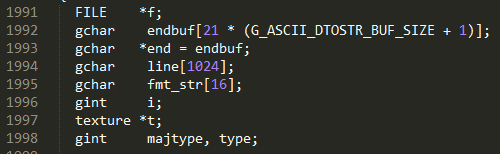
In the buggy version, if counter is larger than the size of stream, an overflow will happen. Check the value to prevent this.

NO.72 buggy version





NO.72 fixed version

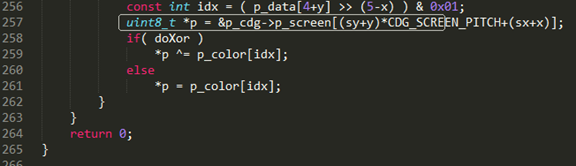




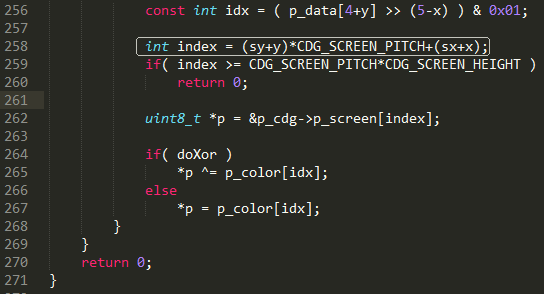


If the value of G\_ASCII\_DTOSTR\_BUF\_SIZE is relatively small, a buffer overflow will happen in sscanf.

NO.74 buggy version

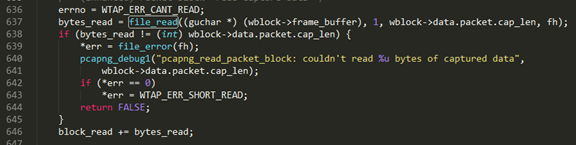


NO.74 fixed version

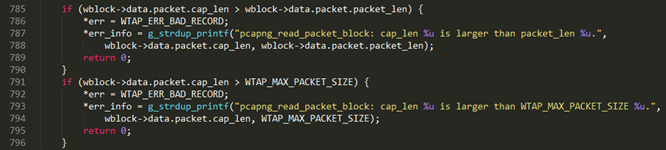


The array p\_screen may cause an overflow.

NO.75 buggy version

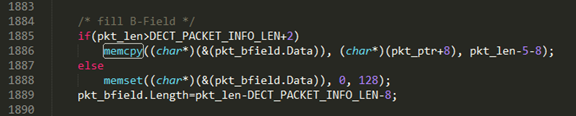


NO.75 fixed version

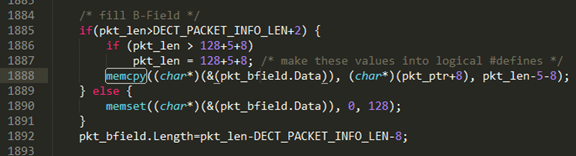


If cap\_len is larger than the size of buffer, a buffer overflow will happen.

NO.76 buggy version

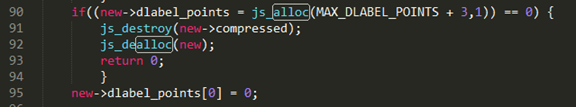


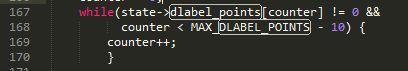
NO.76 fixed version



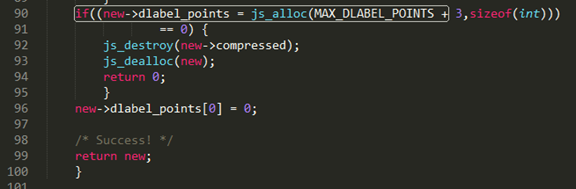
The function memcpy needs boundary check.

NO.78 buggy version



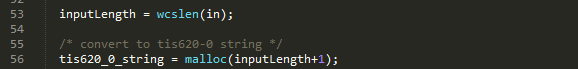


NO.78 fixed version

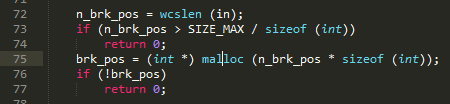


Not enough memory is allocated to dlabel\_points, which causes a buffer overflow in the while loop.

NO.79 buggy version

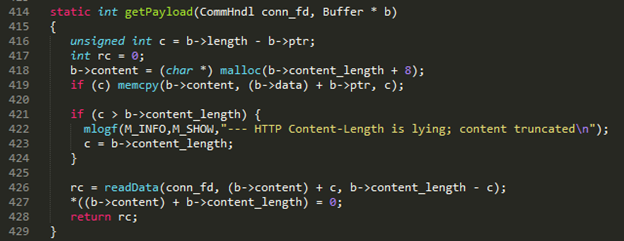


NO.79 fixed version

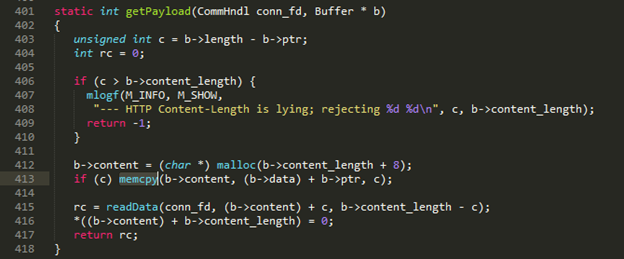


The function malloc’s parameter and return value need to be checked.

NO.80 buggy version

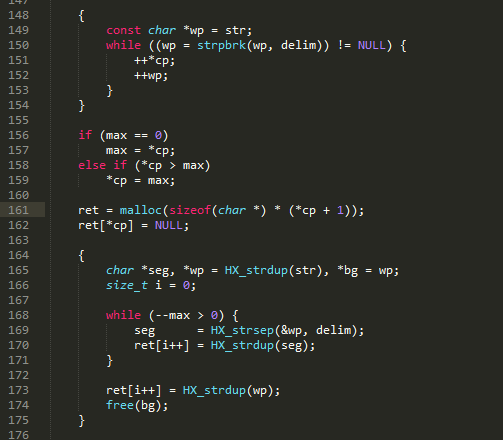


NO.80 fixed version

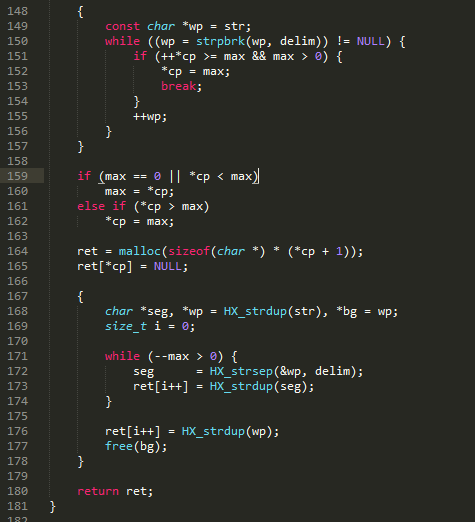


If the length of data(b->content\_length) is too short, a buffer overflow will happen in memcpy.

NO.81 buggy version

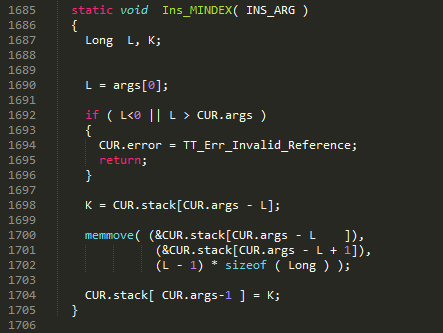


NO.81 fixed version

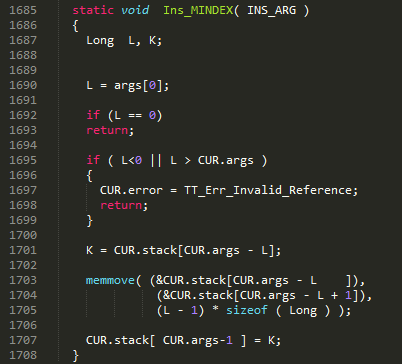


In the buggy version, the while loop in line 150 may end with \*cp<max. In line 161, ret is allocated memory with size \*cp+1 and line 170 will write max values into ret. This may lead to a buffer overflow.

NO.82 buggy version

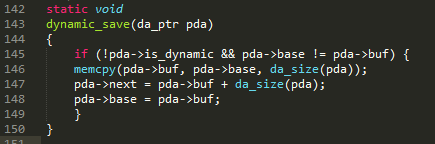


NO.82 fixed version

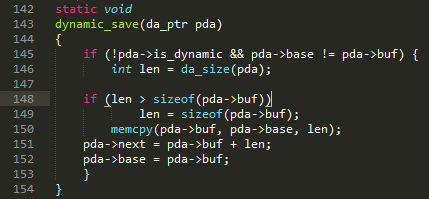


The function memmove may cause an overflow when L=0.

NO.83 buggy version

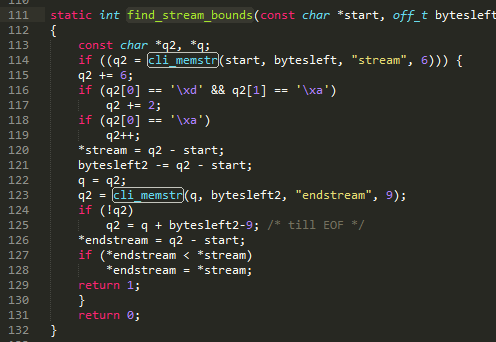


NO.83 fixed version

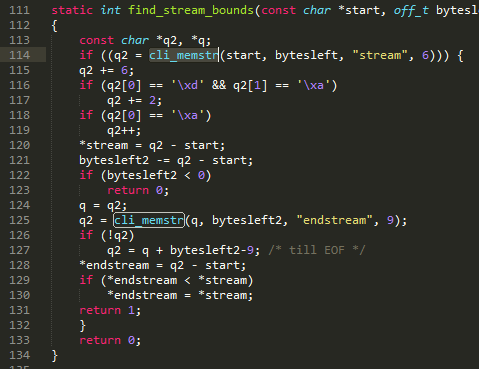


The function memcpy may overflow if da\_size(pda) is larger than the size of pda->buf.

NO.84 buggy version



NO.84 fixed version

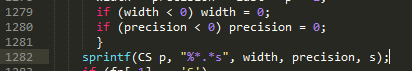


When the variable byteleft2 becomes negative, it will become a large positive number, which may cause an overflow in cli\_memstr. The API array is involved in this function.

NO.85 buggy version

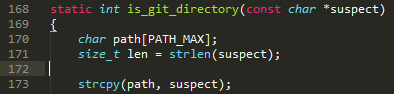


NO.85 fixed version

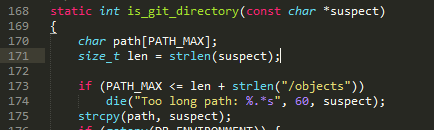


When width and precision is negative, they will turn into large positive numbers. An overflow will happen.

NO.87 buggy version

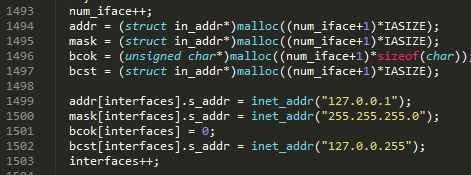


NO.87 fixed version

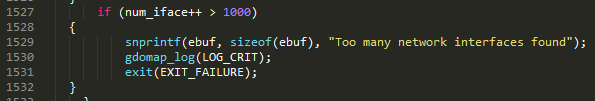


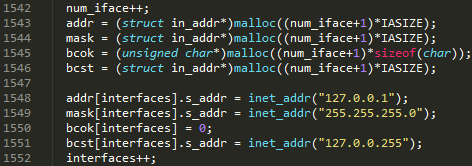
The function strcpy needs a boundary check.

NO.88 buggy version



NO.88 fixed version





If num\_iface is too large, malloc will fail. The following array access may cause an overflow.

NO.89 buggy version

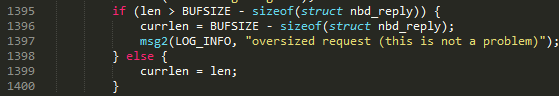




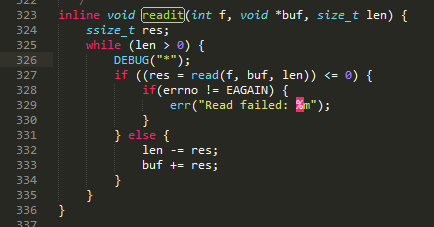


NO.89 fixed version



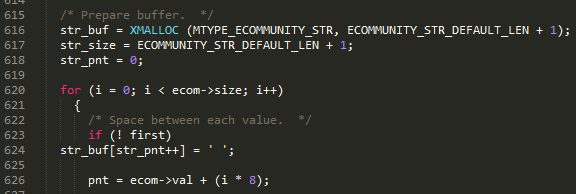


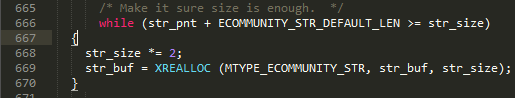
The function readit is implemented as:



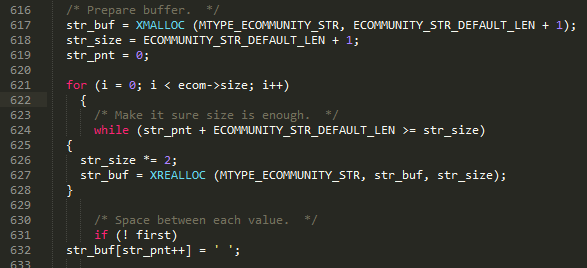
When len is not right, a buffer overflow may happen.

NO.90 buggy version



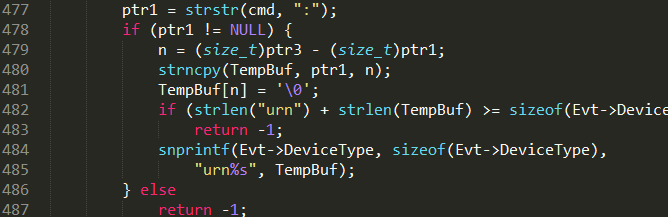


NO.90 fixed version

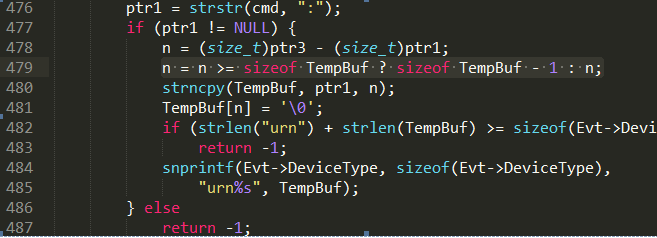


The position of the boundary check is not right.

NO.91 buggy verson

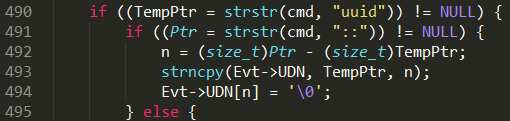


NO.91 fixed version

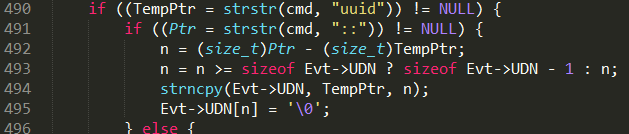


The function strncpy needs a boundary check.

NO.92 buggy version

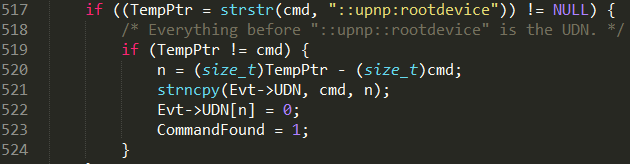


NO.92 fixed version

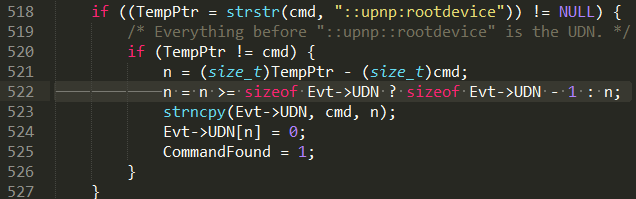


The function strncpy needs a boundary check.

NO.93 buggy version

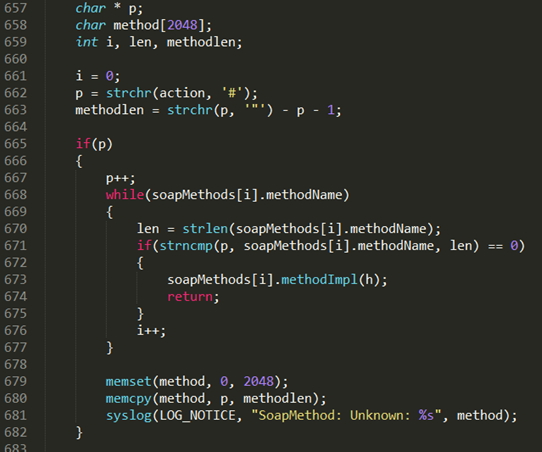


NO.93 fixed version



The function strncpy needs a boundary check.

NO.94 buggy version



NO.94 fixed version

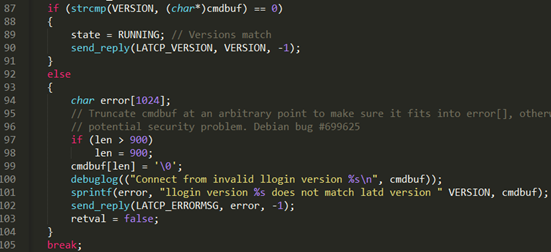


In the buggy version, if the function strchr in line 663 returns a null, methodlen will be negative. In memcpy, it will become a large positive number. In the fixed version, they change the implementation to ensure methodlen is not negative.

NO.95 buggy version



NO.95 fixed version

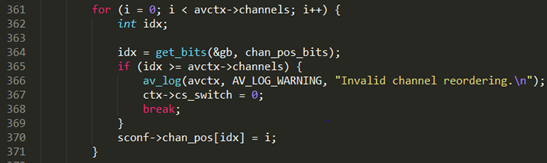


The parameter cmdbuf may cause a buffer overflow in the function sprintf.

NO.96 buggy version

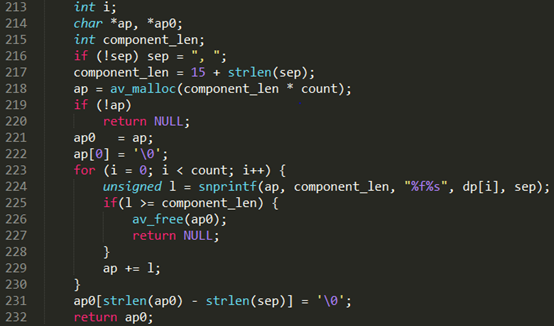


NO.96 fixed version

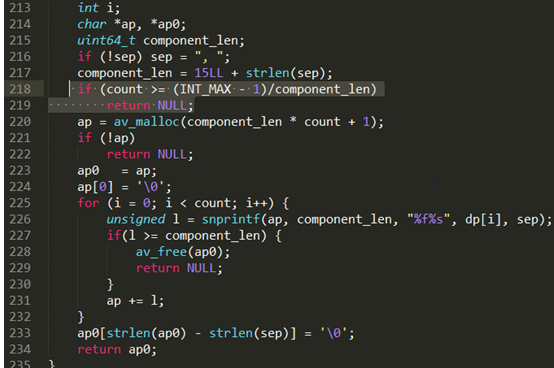


The array chan\_pos needs a boundary check to prevent the buffer overflow.

NO.97 buggy version



NO.97 fixed version



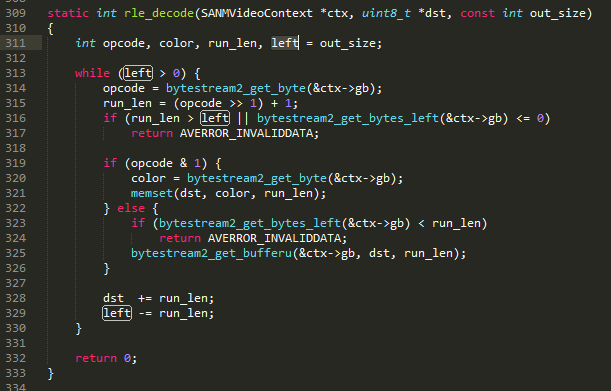
Not enough memory is allocated to ap.

NO.98 buggy version



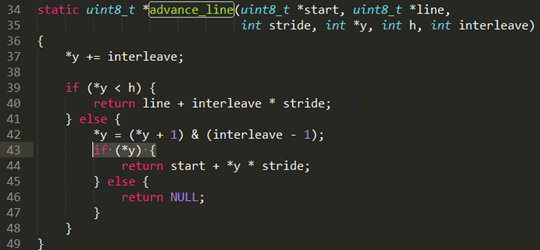
NO.98 fixed version



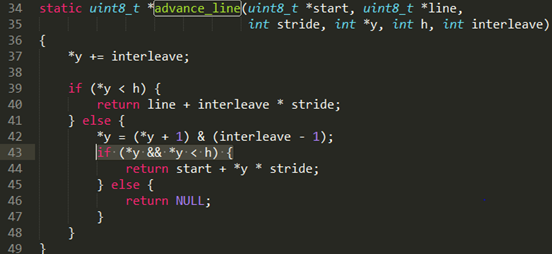


In the buggy version, the function memset in the function rle\_decode may cause a buffer overflow.

NO.99 buggy version



NO.99 fixed version



If the value of \*y is too large, the return value may cause an overflow.

NO.101 buggy version





NO.101 fixed version

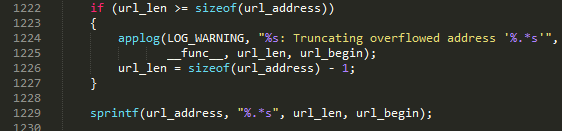


The function sprintf may cause a buffer overflow.

NO.102 buggy version

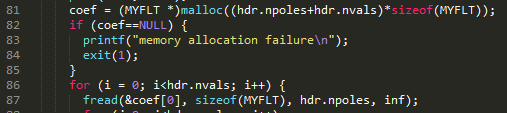


NO.102 fixed version

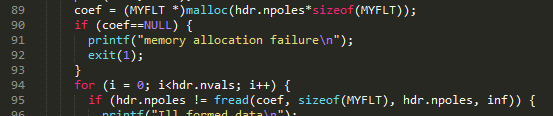


The function sprintf needs a boundary check to prevent the buffer overflow.

NO.103 buggy version

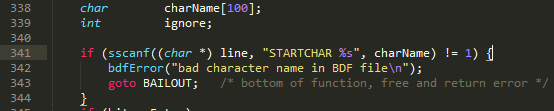


NO.103 fixed version

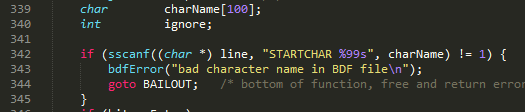


Using smaller parameter can prevent integer overflow in malloc, which may cause an overflow in the function fread.

NO.104 buggy version

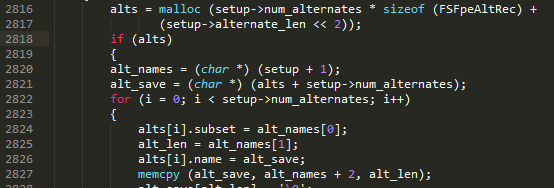


NO.104 fixed version

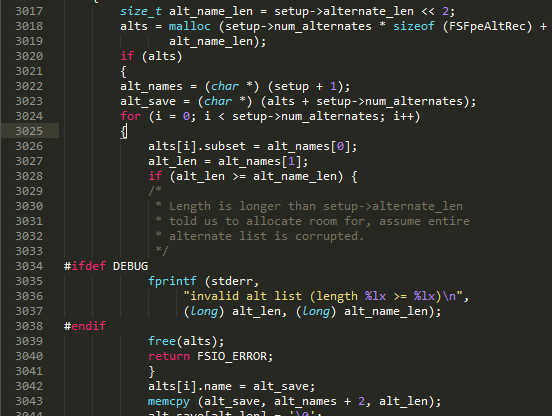


A buffer overflow happens in the function sscanf.

NO.105 buggy version



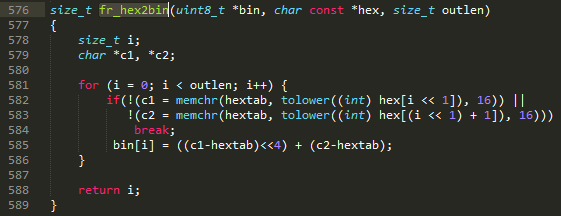
NO.105 fixed version



The function memcpy needs a boundary check to prevent the buffer overflow.

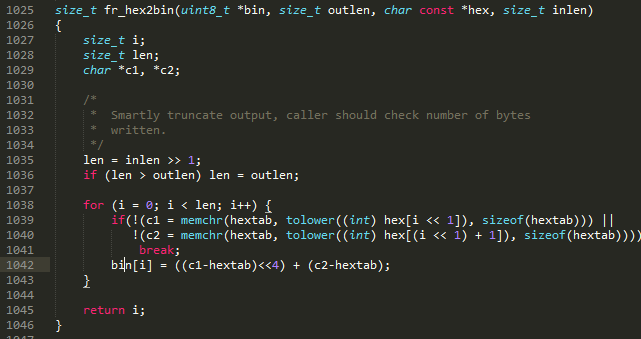
NO.106 buggy version





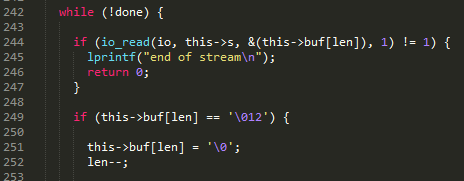
NO.106 fixed version

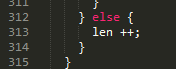




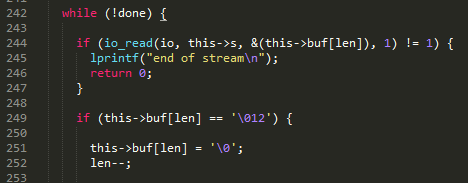
The array bin may cause a buffer overflow.

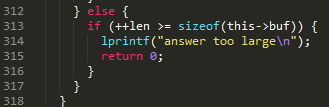
NO.107 buggy version





NO.107 fixed version



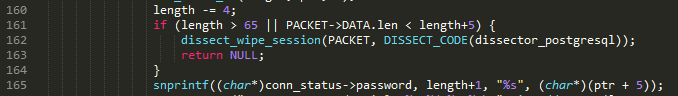


The array buf needs a boundary check to prevent the buffer overflow.

NO.108 buggy version

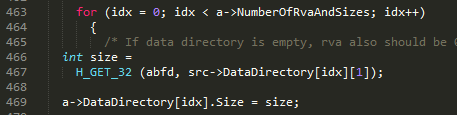


NO.108 fixed version

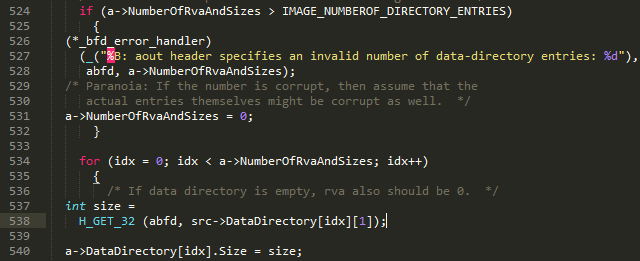


The function strncpy needs a boundary check.

NO.109 buggy version



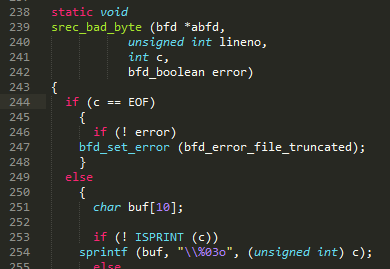
NO.109 fixed version





The array DataDirectory needs a boundary check.

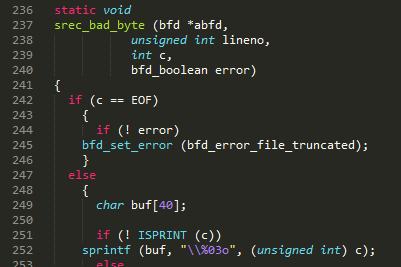
NO.110 buggy version







NO.110 fixed version





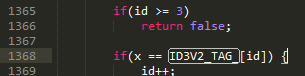


Using a larger buffer for the buffer buf can prevent the buffer overflow.

NO.111 buggy version



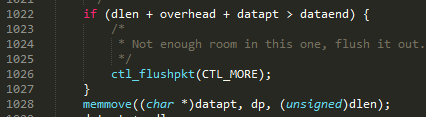
NO.111 fixed version



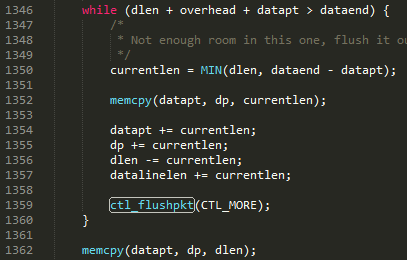
The buffer overflow happens in the array ID3V2\_TAG\_, which is defined as



NO.112 buggy version



NO.112 fixed version

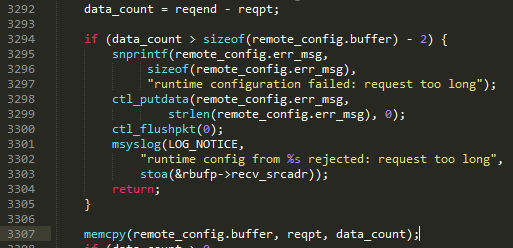


Replacing if with while can ensure the function memmove won’t overflow.

NO.113 buggy version



NO.113 fixed version

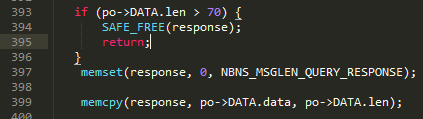


The function memcpy needs a boundary check.

NO.114 buggy version



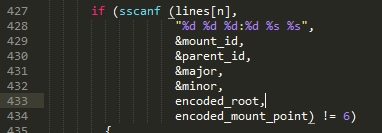
NO.114 fixed version



The function memcpy needs a boundary check.

NO.115 buggy version

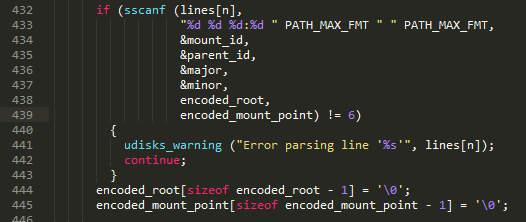




NO.115 fixed version







A buffer overflow happens in the function sscanf.

NO.116 buggy version





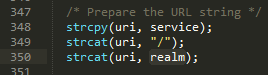
NO.116 fixed version





Substituting snprintf for sprintf to prevent a buffer overflow.

NO.117 buggy version



NO.117 fixed version



A buffer overflow happens in the function strcat.

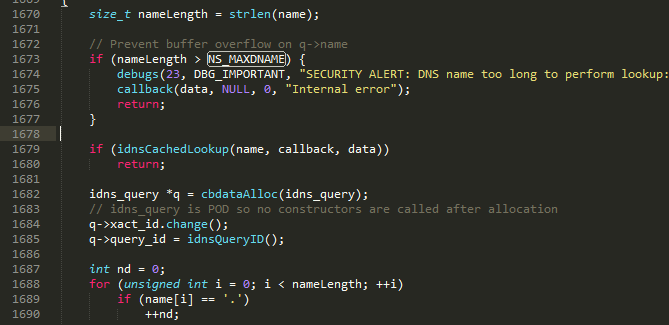
NO.118 buggy version







NO.118 fixed version





The function strcpy needs a boundary check.

NO.119 buggy version





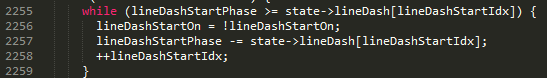
NO.119 fixed version



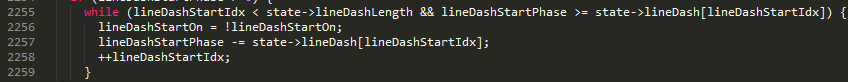


Substituting snprintf for sprintf to prevent a buffer overflow.

NO.120 buggy version



NO.120 fixed version



The array lineDash needs a boundary check.

NO.121 buggy version

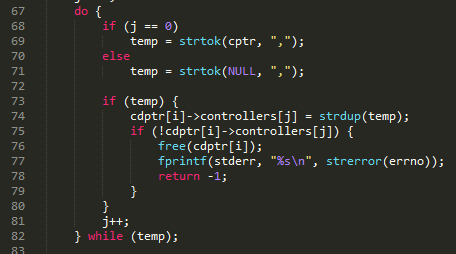


NO.121 fixed version

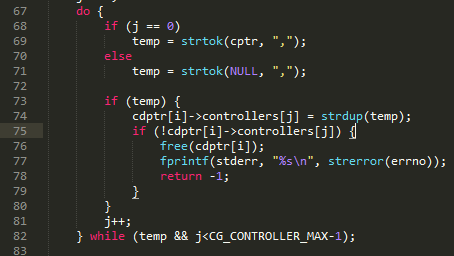


Using the function min to prevent the buffer overflow in memcpy.

NO.122 buggy version



NO.122 fixed version

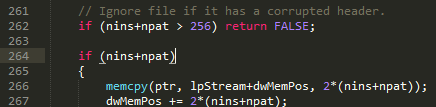


A buffer overflow happens in the array controllers.

NO.123 buggy version



NO.123 fixed version



The function memcpy needs a boundary check to prevent a buffer overflow.