****
6057 Mon Mar 18 19:38:48 2019
new/usr/src/tools/smatch/src/check_free.c
smatch: check libld_* allocation functions
****
_____unchanged_portion_omitted_

```
 282 void check_free(int id)
 283 {
 284         my_id = id;

 286         if (option_project == PROJ_KERNEL) {
 287                 /* The kernel use check_free_strict.c */
 288                 return;
 289         }

 291         add_function_hook("free", &match_free, INT_PTR(0));

 293         if (option_project == PROJ_ILLUMOS_USER)
 294                 add_function_hook("libld_free", &match_free, INT_PTR(0));

 296 #endif /* ! codereview */
 297         if (option_spammy)
 298                 add_hook(&match_symbol, SYM_HOOK);
 299         add_hook(&match_dereferences, DEREF_HOOK);
 300         add_hook(&match_call, FUNCTION_CALL_HOOK);
 301         add_hook(&match_return, RETURN_HOOK);

 303         add_modification_hook(my_id, &ok_to_use);
 304         select_return_implies_hook(PARAM_FREED, &set_param_freed);
 305         add_pre_merge_hook(my_id, &pre_merge_hook);
 306 }
```

****
1457 Mon Mar 18 19:38:48 2019
new/usr/src/tools/smatch/src/check_freeing_null.c
smatch: check libld_* allocation functions
****
_____***unchanged_portion_omitted_***

```
 38 void check_freeing_null(int id)
 39 {
 40         my_id = id;
 41         if (!option_spammy)
 42                 return;
 43         if (option_project == PROJ_KERNEL) {
 43         if (option_project == PROJ_KERNEL)
 44                 add_function_hook("kfree", &match_free, NULL);
 45         } else {
 45         else
 46                 add_function_hook("free", &match_free, NULL);
 47         }

 49         if (option_project == PROJ_ILLUMOS_USER)
 50                 add_function_hook("libld_free", &match_free, NULL);
 51 #endif /* ! codereview */
 52 }
```

****
3530 Mon Mar 18 19:38:49 2019
new/usr/src/tools/smatch/src/check_frees_argument.c
smatch: check libld_* allocation functions
****
_____***unchanged_portion_omitted_***

```
130 void check_frees_argument(int id)
131 {
132         if (!option_info)
133                 return;

135         my_id = id;
136         add_hook(&match_function_def, FUNC_DEF_HOOK);
137         if (option_project == PROJ_KERNEL)
138                 add_function_hook("kfree", &match_kfree, NULL);
139         else
140                 add_function_hook("free", &match_kfree, NULL);

142         if (option_project == PROJ_ILLUMOS_USER)
143                 add_function_hook("libld_free", &match_kfree, NULL);

145 #endif /* ! codereview */
146         add_hook(&match_return, RETURN_HOOK);
147         add_hook(&match_end_func, END_FUNC_HOOK);
148         add_hook(&match_after_func, AFTER_FUNC_HOOK);
149 }
```

****
3085 Mon Mar 18 19:38:49 2019
new/usr/src/tools/smatch/src/check_frees_param.c
smatch: check libld_* allocation functions
****
_____***unchanged_portion_omitted_***

```
 105 void check_frees_param(int id)
 106 {
 107         my_id = id;

 109         if (option_project == PROJ_KERNEL) {
 110                 /* The kernel uses check_frees_param_strict.c */
 111                 return;
 112         }

 114         add_hook(&match_function_def, FUNC_DEF_HOOK);

 116         add_function_hook("free", &match_free, INT_PTR(0));
 117         if (option_project == PROJ_ILLUMOS_USER)
 118                 add_function_hook("libld_free", &match_free, INT_PTR(0));
 119 #endif /* ! codereview */

 121         select_return_implies_hook(PARAM_FREED, &set_param_freed);
 122         add_modification_hook(my_id, &set_ignore);

 124         all_return_states_hook(&process_states);
 125 }
```

```
**********************************************************
    2700 Mon Mar 18 19:38:50 2019
new/usr/src/tools/smatch/src/check_kmalloc_wrong_size.c
smatch: check libld_* allocation functions
**********************************************************
______unchanged_portion_omitted_

  83 void check_kmalloc_wrong_size(int id)
  84 {
  85         my_id = id;

  87         if (option_project == PROJ_KERNEL) {
  88                 add_function_assign_hook("kmalloc", &match_alloc, NULL);
  89                 add_function_assign_hook("kcalloc", &match_calloc, INT_PTR(1));
  90         }
  91
  87         if (option_project != PROJ_KERNEL) {
  92         add_function_assign_hook("malloc", &match_alloc, NULL);
  93         add_function_assign_hook("calloc", &match_calloc, INT_PTR(1));
  90                 return;
  91         }

  95         if (option_project == PROJ_ILLUMOS_USER)
  96                 add_function_assign_hook("libld_calloc", &match_calloc, INT_PTR(
  93         add_function_assign_hook("kmalloc", &match_alloc, NULL);
  94         add_function_assign_hook("kcalloc", &match_calloc, INT_PTR(1));
  97 }
______unchanged_portion_omitted_
```

```
**********************************************************
    6041 Mon Mar 18 19:38:50 2019
new/usr/src/tools/smatch/src/check_leaks.c
smatch: check libld_* allocation functions
**********************************************************
```

```
 1 /*
 2  * Copyright (C) 2010 Dan Carpenter.
 3  *
 4  * This program is free software; you can redistribute it and/or
 5  * modify it under the terms of the GNU General Public License
 6  * as published by the Free Software Foundation; either version 2
 7  * of the License, or (at your option) any later version.
 8  *
 9  * This program is distributed in the hope that it will be useful,
10  * but WITHOUT ANY WARRANTY; without even the implied warranty of
11  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
12  * GNU General Public License for more details.
13  *
14  * You should have received a copy of the GNU General Public License
15  * along with this program; if not, see http://www.gnu.org/copyleft/gpl.txt
16  */

18 /*
19  * The point of this check is to look for leaks.
20  * foo = malloc();  // <- mark it as allocated.
21  * A variable becomes &ok if we:
22  * 1) assign it to another variable.
23  * 2) pass it to a function.
24  *
25  * One complication is dealing with stuff like:
26  * foo->bar = malloc();
27  * foo->baz = malloc();
28  * foo = something();
29  *
30  * The work around is that for now what this check only
31  * checks simple expressions and doesn't check whether
32  * foo->bar is leaked.
33  *
34  */

36 #include <fcntl.h>
37 #include <unistd.h>
38 #include "parse.h"
39 #include "smatch.h"
40 #include "smatch_slist.h"

42 static int my_id;

44 STATE(allocated);
45 STATE(ok);

47 static void set_parent(struct expression *expr, struct smatch_state *state);

49 static const char *allocation_funcs[] = {
50         "malloc",
51         "kmalloc",
52         "kzalloc",
53         "kmemdup",
54 };

49 static char *alloc_parent_str(struct symbol *sym)
50 {
51         static char buf[256];

53         if (!sym || !sym->ident)
54                 return NULL;
```



```
56         snprintf(buf, 255, "%s", sym->ident->name);
57         buf[255] = '\0';
58         return alloc_string(buf);
59 }
______unchanged_portion_omitted_

247 void check_leaks(int id)
248 {
256         int i;

249         my_id = id;

251         switch (option_project) {
252         case PROJ_KERNEL:
253                 add_function_assign_hook("kmalloc", &match_alloc, NULL);
254                 add_function_assign_hook("kzalloc", &match_alloc, NULL);
255                 add_function_assign_hook("kmemdup", &match_alloc, NULL);
256                 break;
257         case PROJ_ILLUMOS_USER:
258                 add_function_assign_hook("libld_malloc", &match_alloc, NULL);
259                 add_function_assign_hook("libld_calloc", &match_alloc, NULL);
260                 /* FALLTHROUGH */
261         default:
262                 add_function_assign_hook("malloc", &match_alloc, NULL);
263                 add_function_assign_hook("calloc", &match_alloc, NULL);
264         }
260         for (i = 0; i < ARRAY_SIZE(allocation_funcs); i++)
261                 add_function_assign_hook(allocation_funcs[i], &match_alloc, NULL
265
266         add_hook(&match_condition, CONDITION_HOOK);

268         add_hook(&match_function_call, FUNCTION_CALL_HOOK);
269         add_hook(&match_assign, ASSIGNMENT_HOOK);

271         add_hook(&match_return, RETURN_HOOK);
272         add_hook(&match_end_func, END_FUNC_HOOK);
273 }
______unchanged_portion_omitted_
```

****
10163 Mon Mar 18 19:38:51 2019
new/usr/src/tools/smatch/src/check_memory.c
smatch: check libld_* allocation functions
****
_____***unchanged_portion_omitted_***

```
 447 void check_memory(int id)
 448 {
 449         my_id = id;
 450         add_unmatched_state_hook(my_id, &unmatched_state);
 451         add_hook(&match_function_def, FUNC_DEF_HOOK);
 452         add_hook(&match_declarations, DECLARATION_HOOK);
 453         add_hook(&match_function_call, FUNCTION_CALL_HOOK);
 454         add_hook(&match_condition, CONDITION_HOOK);
 455         add_hook(&match_assign, ASSIGNMENT_HOOK);
 456         add_hook(&match_return, RETURN_HOOK);
 457         add_hook(&match_end_func, END_FUNC_HOOK);
 458         add_hook(&match_after_func, AFTER_FUNC_HOOK);
 459         add_modification_hook(my_id, &set_unfree);
 460         if (option_project == PROJ_KERNEL) {
 461                 add_function_hook("kfree", &match_free_func, (void *)0);
 462                 register_funcs_from_file();
 463         } else {
 464                 add_function_hook("free", &match_free_func, (void *)0);
 465                 if (option_project == PROJ_ILLUMOS_USER)
 466                         add_function_hook("libld_free", &match_free_func,
 467                                 (void *)0);
 468 #endif /* ! codereview */
 469         }
 470 }
```

```
**********************************************************
   15272 Mon Mar 18 19:38:51 2019
new/usr/src/tools/smatch/src/smatch_buf_comparison.c
smatch: check libld_* allocation functions
**********************************************************
______unchanged_portion_omitted_

 563 void register_buf_comparison(int id)
 564 {
 565         size_id = id;

 567         add_unmatched_state_hook(size_id, &unmatched_state);

 569         add_allocation_function("malloc", &match_alloc, 0);
 570         add_allocation_function("memdup", &match_alloc, 1);
 571         add_allocation_function("realloc", &match_alloc, 1);
 572         if (option_project == PROJ_KERNEL) {
 573                 add_allocation_function("kmalloc", &match_alloc, 0);
 574                 add_allocation_function("kzalloc", &match_alloc, 0);
 575                 add_allocation_function("vmalloc", &match_alloc, 0);
 576                 add_allocation_function("__vmalloc", &match_alloc, 0);
 577                 add_allocation_function("sock_kmalloc", &match_alloc, 1);
 578                 add_allocation_function("kmemdup", &match_alloc, 1);
 579                 add_allocation_function("kmemdup_user", &match_alloc, 1);
 580                 add_allocation_function("dma_alloc_attrs", &match_alloc, 1);
 581                 add_allocation_function("pci_alloc_consistent", &match_alloc, 1)
 582                 add_allocation_function("pci_alloc_coherent", &match_alloc, 1);
 583                 add_allocation_function("devm_kmalloc", &match_alloc, 1);
 584                 add_allocation_function("devm_kzalloc", &match_alloc, 1);
 585                 add_allocation_function("kcalloc", &match_calloc, 0);
 586                 add_allocation_function("devm_kcalloc", &match_calloc, 1);
 587                 add_allocation_function("kmalloc_array", &match_calloc, 0);
 588                 add_allocation_function("krealloc", &match_alloc, 1);
 589         } else if (option_project == PROJ_ILLUMOS_USER) {
 590                 add_allocation_function("libld_malloc", &match_alloc, 0);
 591                 add_allocation_function("libld_calloc", &match_calloc, 0);
 592                 add_allocation_function("libld_realloc", &match_calloc, 1);
 593 #endif /* ! codereview */
 594         }

 596         add_hook(&array_check, OP_HOOK);
 597         add_hook(&array_check_data_info, OP_HOOK);

 599         add_hook(&match_call, FUNCTION_CALL_HOOK);
 600         select_caller_info_hook(set_param_compare, ARRAY_LEN);
 601         select_caller_info_hook(set_arraysize_arg, ARRAYSIZE_ARG);
 602         add_hook(&munge_start_states, AFTER_DEF_HOOK);
 603 }

 605 void register_buf_comparison_links(int id)
 606 {
 607         link_id = id;
 608         add_merge_hook(link_id, &merge_links);
 609         add_modification_hook(link_id, &match_link_modify);
 610 }
```

```
**********************************************************
   21904 Mon Mar 18 19:38:52 2019
new/usr/src/tools/smatch/src/smatch_buf_size.c
smatch: check libld_* allocation functions
**********************************************************
______unchanged_portion_omitted_

 872 void register_buf_size(int id)
 873 {
 874         my_size_id = id;

 876         add_unmatched_state_hook(my_size_id, &unmatched_size_state);

 878         select_caller_info_hook(set_param_buf_size, BUF_SIZE);
 879         select_return_states_hook(BUF_SIZE, &db_returns_buf_size);
 880         add_split_return_callback(print_returned_allocations);

 882         allocation_funcs = create_function_hashtable(100);
 883         add_allocation_function("malloc", &match_alloc, 0);
 884         add_allocation_function("calloc", &match_calloc, 0);
 885         add_allocation_function("memdup", &match_alloc, 1);
 886         add_allocation_function("realloc", &match_alloc, 1);
 887         if (option_project == PROJ_KERNEL) {
 888                 add_allocation_function("kmalloc", &match_alloc, 0);
 889                 add_allocation_function("kmalloc_node", &match_alloc, 0);
 890                 add_allocation_function("kzalloc", &match_alloc, 0);
 891                 add_allocation_function("kzalloc_node", &match_alloc, 0);
 892                 add_allocation_function("vmalloc", &match_alloc, 0);
 893                 add_allocation_function("__vmalloc", &match_alloc, 0);
 894                 add_allocation_function("kcalloc", &match_calloc, 0);
 895                 add_allocation_function("kmalloc_array", &match_calloc, 0);
 896                 add_allocation_function("drm_malloc_ab", &match_calloc, 0);
 897                 add_allocation_function("drm_calloc_large", &match_calloc, 0);
 898                 add_allocation_function("sock_kmalloc", &match_alloc, 1);
 899                 add_allocation_function("kmemdup", &match_alloc, 1);
 900                 add_allocation_function("kmemdup_user", &match_alloc, 1);
 901                 add_allocation_function("dma_alloc_attrs", &match_alloc, 1);
 902                 add_allocation_function("pci_alloc_consistent", &match_alloc, 1)
 903                 add_allocation_function("pci_alloc_coherent", &match_alloc, 1);
 904                 add_allocation_function("devm_kmalloc", &match_alloc, 1);
 905                 add_allocation_function("devm_kzalloc", &match_alloc, 1);
 906                 add_allocation_function("krealloc", &match_alloc, 1);
 907                 add_allocation_function("__alloc_bootmem", &match_alloc, 0);
 908                 add_allocation_function("alloc_bootmem", &match_alloc, 0);
 909                 add_allocation_function("kmap", &match_page, 0);
 910                 add_allocation_function("get_zeroed_page", &match_page, 0);
 911                 add_allocation_function("alloc_pages", &match_alloc_pages, 1);
 912                 add_allocation_function("alloc_pages_current", &match_alloc_page
 913                 add_allocation_function("__get_free_pages", &match_alloc_pages,
 914         } else if (option_project == PROJ_ILLUMOS_USER) {
 915                 add_allocation_function("libld_malloc", &match_alloc, 0);
 916                 add_allocation_function("libld_realloc", &match_alloc, 1);
 917                 add_allocation_function("libld_calloc", &match_calloc, 0);
 918 #endif /* ! codereview */
 919         }

 921         add_allocation_function("strndup", match_strndup, 0);
 922         if (option_project == PROJ_KERNEL)
 923                 add_allocation_function("kstrndup", match_strndup, 0);

 925         add_modification_hook(my_size_id, &set_size_undefined);

 927         add_merge_hook(my_size_id, &merge_size_func);

 929         if (option_info)
 930                 add_hook(record_global_size, BASE_HOOK);
```



```
 931 }

 933 void register_buf_size_late(int id)
 934 {
 935         /* has to happen after match_alloc() */
 936         add_hook(&match_array_assignment, ASSIGNMENT_HOOK);

 938         add_hook(&match_call, FUNCTION_CALL_HOOK);
 939         add_member_info_callback(my_size_id, struct_member_callback);
 940 }
```

```
**********************************************************
   13653 Mon Mar 18 19:38:52 2019
new/usr/src/tools/smatch/src/smatch_constraints_required.c
smatch: check libld_* allocation functions
**********************************************************
______unchanged_portion_omitted_

 29 static struct allocator illumos_user_allocator_table[] = {
 30         {"libld_malloc", 0},
 31         {"libld_realloc", 1},
 32 };

 34 #endif /* ! codereview */
 35 static struct allocator generic_allocator_table[] = {
 36         {"malloc", 0},
 37         {"memdup", 1},
 38         {"realloc", 1},
 39 };

 41 static struct allocator kernel_allocator_table[] = {
 42         {"kmalloc", 0},
 43         {"kzalloc", 0},
 44         {"vmalloc", 0},
 45         {"__vmalloc", 0},
 46         {"vzalloc", 0},
 47         {"sock_kmalloc", 1},
 48         {"kmemdup", 1},
 49         {"kmemdup_user", 1},
 50         {"dma_alloc_attrs", 1},
 51         {"pci_alloc_consistent", 1},
 52         {"pci_alloc_coherent", 1},
 53         {"devm_kmalloc", 1},
 54         {"devm_kzalloc", 1},
 55         {"krealloc", 1},
 56 };

 58 static struct allocator illumos_user_calloc_table[] = {
 59         {"libld_calloc", 0, 1},
 29 static struct allocator calloc_table[] = {
 60         {"calloc", 0, 1},
 61         {"kcalloc", 0, 1},
 62         {"kmalloc_array", 0, 1},
 63         {"devm_kcalloc", 1, 2},
 64 };

 66 static struct allocator generic_calloc_table[] = {
 67         {"calloc", 0, 1},
 68 };

 70 static struct allocator kernel_calloc_table[] = {
 71         {"kcalloc", 0, 1},
 72         {"kmalloc_array", 0, 1},
 73         {"devm_kcalloc", 1, 2},
 74 };

 76 #endif /* ! codereview */
 77 static int bytes_per_element(struct expression *expr)
 78 {
 79         struct symbol *type;

 81         type = get_type(expr);
 82         if (!type)
 83                 return 0;

 85         if (type->type != SYM_PTR && type->type != SYM_ARRAY)
 86                 return 0;
```



```
 88         type = get_base_type(type);
 89         return type_bytes(type);
 90 }

 92 static void save_constraint_required(struct expression *pointer, int op, struct
 93 {
 94         char *data, *limit;

 96         data = get_constraint_str(pointer);
 97         if (!data)
 98                 return;

100         limit = get_constraint_str(constraint);
101         if (!limit) {
102                 // FIXME deal with <= also
103                 if (op == '<')
104                         set_state_expr(my_id, constraint, alloc_state_expr(point
105                 goto free_data;
106         }

108         sql_save_constraint_required(data, op, limit);

110         free_string(limit);
111 free_data:
112         free_string(data);
113 }

115 static int handle_zero_size_arrays(struct expression *pointer, struct expression
116 {
117         struct expression *left, *right;
118         struct symbol *type, *array, *array_type;
119         sval_t struct_size;
120         char *limit;
121         char data[128];

123         if (size->type != EXPR_BINOP || size->op != '+')
124                 return 0;

126         type = get_type(pointer);
127         if (!type || type->type != SYM_PTR)
128                 return 0;
129         type = get_real_base_type(type);
130         if (!type || !type->ident || type->type != SYM_STRUCT)
131                 return 0;
132         if (!last_member_is_resizable(type))
133                 return 0;
134         array = last_ptr_list((struct ptr_list *)type->symbol_list);
135         if (!array || !array->ident)
136                 return 0;
137         array_type = get_real_base_type(array);
138         if (!array_type || array_type->type != SYM_ARRAY)
139                 return 0;
140         array_type = get_real_base_type(array_type);

142         left = strip_expr(size->left);
143         right = strip_expr(size->right);

145         if (!get_implied_value(left, &struct_size))
146                 return 0;
147         if (struct_size.value != type_bytes(type))
148                 return 0;

150         if (right->type == EXPR_BINOP && right->op == '*') {
151                 struct expression *mult_left, *mult_right;
152                 sval_t sval;
```

```
154                 mult_left = strip_expr(right->left);
155                 mult_right = strip_expr(right->right);

157                 if (get_implied_value(mult_left, &sval) &&
158                     sval.value == type_bytes(array_type))
159                         size = mult_right;
160                 else if (get_implied_value(mult_right, &sval) &&
161                     sval.value == type_bytes(array_type))
162                         size = mult_left;
163                 else
164                         return 0;
165         }

167         snprintf(data, sizeof(data), "(struct %s)->%s", type->ident->name, array
168         limit = get_constraint_str(size);
169         if (!limit) {
170                 set_state_expr(my_id, size, alloc_state_expr(
171                                member_expression(deref_expression(pointer), '*',
172                 return 1;
173         }

175         sql_save_constraint_required(data, '<', limit);

177         free_string(limit);
178         return 1;
179 }

181 static void match_alloc_helper(struct expression *pointer, struct expression *si
182 {
183         struct expression *size_orig, *tmp;
184         sval_t sval;
185         int cnt = 0;

187         pointer = strip_expr(pointer);
188         size = strip_expr(size);
189         if (!size || !pointer)
190                 return;

192         size_orig = size;
193         if (recurse) {
194                 while ((tmp = get_assigned_expr(size))) {
195                         size = strip_expr(tmp);
196                         if (cnt++ > 5)
197                                 break;
198                 }
199                 if (size != size_orig) {
200                         match_alloc_helper(pointer, size, 0);
201                         size = size_orig;
202                 }
203         }

205         if (handle_zero_size_arrays(pointer, size))
206                 return;

208         if (size->type == EXPR_BINOP && size->op == '*') {
209                 struct expression *mult_left, *mult_right;

211                 mult_left = strip_expr(size->left);
212                 mult_right = strip_expr(size->right);

214                 if (get_implied_value(mult_left, &sval) &&
215                     sval.value == bytes_per_element(pointer))
216                         size = mult_right;
217                 else if (get_implied_value(mult_right, &sval) &&
218                     sval.value == bytes_per_element(pointer))
```



```
219                         size = mult_left;
220                 else
221                         return;
222         }

224         if (size->type == EXPR_BINOP && size->op == '+' &&
225             get_implied_value(size->right, &sval) &&
226             sval.value == 1)
227                 save_constraint_required(pointer, SPECIAL_LTE, size->left);
228         else
229                 save_constraint_required(pointer, '<', size);
230 }

232 static void match_alloc(const char *fn, struct expression *expr, void *_size_arg
233 {
234         int size_arg = PTR_INT(_size_arg);
235         struct expression *call, *arg;

237         call = strip_expr(expr->right);
238         arg = get_argument_from_call_expr(call->args, size_arg);

240         match_alloc_helper(expr->left, arg, 1);
241 }

243 static void match_calloc(const char *fn, struct expression *expr, void *_start_a
244 {
245         struct expression *pointer, *call, *size;
246         struct expression *count = NULL;
247         int start_arg = PTR_INT(_start_arg);
248         sval_t sval;

250         pointer = strip_expr(expr->left);
251         call = strip_expr(expr->right);

253         size = get_argument_from_call_expr(call->args, start_arg);
254         if (get_implied_value(size, &sval) &&
255             sval.value == bytes_per_element(pointer))
256                 count = get_argument_from_call_expr(call->args, start_arg + 1);
257         else {
258                 size = get_argument_from_call_expr(call->args, start_arg + 1);
259                 if (get_implied_value(size, &sval) &&
260                     sval.value == bytes_per_element(pointer))
261                         count = get_argument_from_call_expr(call->args, start_ar
262         }

264         if (!count)
265                 return;

267         save_constraint_required(pointer, '<', count);
268 }

270 static void add_allocation_function(const char *func, void *call_back, int param
271 {
272         add_function_assign_hook(func, call_back, INT_PTR(param));
273 }

275 static void match_assign_size(struct expression *expr)
276 {
277         struct smatch_state *state;
278         char *data, *limit;

280         state = get_state_expr(my_id, expr->right);
281         if (!state || !state->data)
282                 return;

284         data = get_constraint_str(state->data);
```

```
285         if (!data)
286                 return;
287
288         limit = get_constraint_str(expr->left);
289         if (!limit)
290                 goto free_data;
291
292         sql_save_constraint_required(data, '<', limit);
293
294         free_string(limit);
295 free_data:
296         free_string(data);
297 }
298
299 static void match_assign_has_buf_comparison(struct expression *expr)
300 {
301         struct expression *size;
302
303         if (expr->op != '=')
304                 return;
305         if (expr->right->type == EXPR_CALL)
306                 return;
307         size = get_size_variable(expr->right);
308         if (!size)
309                 return;
310         match_alloc_helper(expr->left, size, 1);
311 }
312
313 static void match_assign_data(struct expression *expr)
314 {
315         struct expression *right, *arg, *tmp;
316         int i;
317         int size_arg;
318         int size_arg2 = -1;
319
320         if (expr->op != '=')
321                 return;
322
323         /* Direct calls are handled else where (for now at least) */
324         tmp = get_assigned_expr(expr->right);
325         if (!tmp)
326                 return;
327
328         right = strip_expr(tmp);
329         if (right->type != EXPR_CALL)
330                 return;
331
332         if (right->fn->type != EXPR_SYMBOL ||
333             !right->fn->symbol ||
334             !right->fn->symbol->ident)
335                 return;
336
337         for (i = 0; i < ARRAY_SIZE(generic_allocator_table); i++) {
338                 if (strcmp(right->fn->symbol->ident->name,
339                            generic_allocator_table[i].func) == 0) {
340                         size_arg = generic_allocator_table[i].param;
341                         goto found;
342                 }
343         }
344
345         for (i = 0; i < ARRAY_SIZE(generic_calloc_table); i++) {
346                 if (strcmp(right->fn->symbol->ident->name,
347                            generic_calloc_table[i].func) == 0) {
348                         size_arg = generic_calloc_table[i].param;
349                         size_arg2 = generic_calloc_table[i].param2;
350                         goto found;
```



```
351                 }
352         }
353
354         if (option_project == PROJ_ILLUMOS_USER) {
355                 if (strcmp(right->fn->symbol->ident->name,
356                            illumos_user_allocator_table[i].func) == 0) {
357                         size_arg = illumos_user_allocator_table[i].param;
358                         goto found;
359                 }
360
361                 for (i = 0; i < ARRAY_SIZE(illumos_user_calloc_table); i++) {
362                         if (strcmp(right->fn->symbol->ident->name,
363                             illumos_user_calloc_table[i].func) == 0) {
364                                 size_arg = illumos_user_calloc_table[i].param;
365                                 size_arg2 = illumos_user_calloc_table[i].param2;
366                                 goto found;
367                         }
368                 }
369         }
370
371 #endif /* ! codereview */
372         if (option_project != PROJ_KERNEL)
373                 return;
374
375         for (i = 0; i < ARRAY_SIZE(kernel_allocator_table); i++) {
376                 if (strcmp(right->fn->symbol->ident->name,
377                            kernel_allocator_table[i].func) == 0) {
378                         size_arg = kernel_allocator_table[i].param;
379                         goto found;
380                 }
381         }
382
383         for (i = 0; i < ARRAY_SIZE(kernel_calloc_table); i++) {
 36         for (i = 0; i < ARRAY_SIZE(calloc_table); i++) {
384                 if (strcmp(right->fn->symbol->ident->name,
385                            kernel_calloc_table[i].func) == 0) {
386                         size_arg = kernel_calloc_table[i].param;
387                         size_arg2 = kernel_calloc_table[i].param2;
 38                            calloc_table[i].func) == 0) {
 39                         size_arg = calloc_table[i].param;
 40                         size_arg2 = calloc_table[i].param2;
388                         goto found;
389                 }
390         }
391
392         return;
393
394 found:
395         arg = get_argument_from_call_expr(right->args, size_arg);
396         match_alloc_helper(expr->left, arg, 1);
397         if (size_arg2 == -1)
398                 return;
399         arg = get_argument_from_call_expr(right->args, size_arg2);
400         match_alloc_helper(expr->left, arg, 1);
401 }
_____unchanged_portion_omitted_

499 void register_constraints_required(int id)
500 {
501         my_id = id;
502
503         add_hook(&match_assign_size, ASSIGNMENT_HOOK);
504         add_hook(&match_assign_data, ASSIGNMENT_HOOK);
505         add_hook(&match_assign_has_buf_comparison, ASSIGNMENT_HOOK);
506
507         add_hook(&match_assign_ARRAY_SIZE, ASSIGNMENT_HOOK);
```

```
 508         add_hook(&match_assign_ARRAY_SIZE, GLOBAL_ASSIGNMENT_HOOK);
 509         add_hook(&match_assign_buf_comparison, ASSIGNMENT_HOOK);
 510         add_hook(&match_assign_constraint, ASSIGNMENT_HOOK);

 512         add_allocation_function("malloc", &match_alloc, 0);
 513         add_allocation_function("memdup", &match_alloc, 1);
 514         add_allocation_function("realloc", &match_alloc, 1);
 515         add_allocation_function("calloc", &match_calloc, 0);
 516         if (option_project == PROJ_ILLUMOS_USER) {
 517                 add_allocation_function("libld_malloc", &match_alloc, 0);
 518                 add_allocation_function("libld_realloc", &match_alloc, 1);
 519                 add_allocation_function("libld_calloc", &match_calloc, 0);
 520         }
 168         add_allocation_function("realloc", &match_calloc, 0);
 521         if (option_project == PROJ_KERNEL) {
 522                 add_allocation_function("kmalloc", &match_alloc, 0);
 523                 add_allocation_function("kzalloc", &match_alloc, 0);
 524                 add_allocation_function("vmalloc", &match_alloc, 0);
 525                 add_allocation_function("__vmalloc", &match_alloc, 0);
 526                 add_allocation_function("vzalloc", &match_alloc, 0);
 527                 add_allocation_function("sock_kmalloc", &match_alloc, 1);
 528                 add_allocation_function("kmemdup", &match_alloc, 1);
 529                 add_allocation_function("kmemdup_user", &match_alloc, 1);
 530                 add_allocation_function("dma_alloc_attrs", &match_alloc, 1);
 531                 add_allocation_function("pci_alloc_consistent", &match_alloc, 1)
 532                 add_allocation_function("pci_alloc_coherent", &match_alloc, 1);
 533                 add_allocation_function("devm_kmalloc", &match_alloc, 1);
 534                 add_allocation_function("devm_kzalloc", &match_alloc, 1);
 535                 add_allocation_function("kcalloc", &match_calloc, 0);
 536                 add_allocation_function("kmalloc_array", &match_calloc, 0);
 537                 add_allocation_function("devm_kcalloc", &match_calloc, 1);
 538                 add_allocation_function("krealloc", &match_alloc, 1);
 539         }
 540 }
```

*unchanged_portion_omitted_*