radareorg / **radare2-mcp** Public

Code | Issues 5 | Pull requests 2 | Actions | Models | Security and quality

Commit 482cde6

trufae authored on Mar 24 · 6 / 7 · Verified

Fix #45 - Enable sandbox by default -g to setup the granularity

main (#44) · 1.7.2 1.7.0

1 parent f05e404 commit 482cde6

3 files changed +24 -1

Top

Filter files…

- src
  - main.c
  - r2api.inc.c
  - r2mcp.h

Search within code

src/main.c

```
@@ -38,6 +38,7 @@ void r2mcp_help(void) {
38  38          " -d [pdc]   select a different decompiler (pdc by default)\n"
39  39          " -D [tool]  disable the specified tool (repeatable)\n"
40  40          " -e [tool]  enable only the specified tool (repeatable)\n"
    41  +       " -g [grain] set cfg.sandbox.grain (default: exec,socket; use all to disable sandbox)\n"
41  42          " -h         show this help\n"
42  43          " -i         ignore analysis level specified in analyze calls\n"
43  44          " -l [file]  append debug logs to this file\n"
```

```
@@ -82,6 +83,7 @@ int r2mcp_main(int argc, const char **argv) {
82  83          char *baseurl = NULL;
83  84          char *svc_baseurl = NULL;
84  85          char *sandbox = NULL;
    86  +       char *sandbox_grain = strdup ("exec,socket");
85  87          char *logfile = NULL;
86  88          char *prompts_dir = NULL;
87  89          bool load_prompts = true;
@@ -90,7 +92,7 @@ int r2mcp_main(int argc, const char **argv) {
90  92          const char *dsl_tests = NULL;
91  93          RList *disabled_tools = NULL;
92  94          RGetopt opt;
93      -       r_getopt_init (&opt, argc, argv, "hmvpd:nc:u:l:s:rite:D:RT:S:P:NL");
    95  +       r_getopt_init (&opt, argc, argv, "hmvpd:nc:u:g:l:s:rite:D:RT:S:P:NL");
94  96          int c;
95  97          while ((c = r_getopt_next (&opt)) != -1) {
96  98              switch (c) {
@@ -111,6 +113,10 @@ int r2mcp_main(int argc, const char **argv) {
111 113                  baseurl = strdup (opt.arg);
112 114                  R_LOG_INFO ("[R2MCP] HTTP r2pipe client mode enabled, baseurl=%s",
    baseurl);
113 115                  break;
    116 +           case 'g':
    117 +               free (sandbox_grain);
    118 +               sandbox_grain = strdup (opt.arg);
    119 +               break;
114 120             case 'l':
115 121                 logfile = strdup (opt.arg);
116 122                 break;
@@ -205,6 +211,7 @@ int r2mcp_main(int argc, const char **argv) {
205 211             .baseurl = baseurl,
206 212             .svc_baseurl = svc_baseurl,
207 213             .sandbox = sandbox,
    214 +           .sandbox_grain = sandbox_grain,
208 215             .logfile = logfile,
209 216             .prompts_dir = prompts_dir,
210 217             .load_prompts = load_prompts,
@@ -269,6 +276,7 @@ int r2mcp_main(int argc, const char **argv) {
```

```
                                                                
269  276            free (ss.baseurl);
270  277            free (ss.svc_baseurl);
271  278            free (ss.sandbox);
     279  +         free (ss.sandbox_grain);
272  280            free (ss.logfile);
273  281            free (ss.prompts_dir);
274  282            if (ss.enabled_tools) {
                    @@ -294,6 +302,7 @@ int r2mcp_main(int argc, const char **argv) {
294  302        /* Cleanup */
295  303        free (ss.baseurl);
296  304        free (ss.sandbox);
     305  +     free (ss.sandbox_grain);
297  306        free (ss.logfile);
298  307        free (ss.prompts_dir);
299  308        if (ss.enabled_tools) {
                                                                
```

src/r2api.inc.c

```
                    @@ -21,6 +21,16 @@ static void r2state_settings(RCore *core) {
21   21       r_config_set_i (core->config, "scr.limit", 16768);
22   22   }
23   23
     24  + static void r2state_sandbox_settings(ServerState *ss, RCore *core) {
     25  +     const char *sandbox_grain = (ss && ss->sandbox_grain)? ss->sandbox_grain:
                "exec,socket";
     26  +     if (!strcmp (sandbox_grain, "all")) {
     27  +         r_config_set_b (core->config, "cfg.sandbox", false);
     28  +     } else {
     29  +         r_config_set_b (core->config, "cfg.sandbox", true);
     30  +         r_config_set (core->config, "cfg.sandbox.grain", sandbox_grain);
     31  +     }
     32  + }
     33  +
24   34   static bool logcb(void *user, int type, const char *origin, const char *msg) {
25   35       if (type > R_LOG_LEVEL_WARN) {
26   36           return false;
                    @@ -191,6 +201,7 @@ R_IPI bool r2_open_file(ServerState *ss, const char
                    *filepath) {
191  201           R_LOG_ERROR ("Failed to initialize r2 core\n");
```

```
192  202          return false;
193  203      }
     204  +   r_config_set_b (core->config, "cfg.sandbox", false);
194  205
195  206      if (ss->rstate.file_opened) {
196  207          R_LOG_INFO ("Closing previously opened file: %s", ss-
                >rstate.current_file);
     @@ -230,6 +241,7 @@ R_IPI bool r2_open_file(ServerState *ss, const char
     *filepath) {
230  241      free (ss->rstate.current_file);
231  242      ss->rstate.current_file = strdup (filepath);
232  243      ss->rstate.file_opened = true;
     244  +   r2state_sandbox_settings (ss, core);
233  245      R_LOG_INFO ("File opened successfully: %s", filepath);
234  246
235  247      return true;
```

src/r2mcp.h

```
     @@ -52,6 +52,8 @@ typedef struct {
52   52      char *svc_baseurl;
53   53      /* Optional sandbox path. When set, only allow opening files under this dir
            */
54   54      char *sandbox;
     55  +   /* Optional radare2 sandbox grain mask; "all" disables cfg.sandbox */
     56  +   char *sandbox_grain;
55   57      /* Optional path to append debug logs when set via -l */
56   58      char *logfile;
57   59      /* Optional custom prompts directory path */
```

Comments 0

Please sign in to comment.