New r2 Parse API

Refreshing the state of parsing disassembly

Adam "satk0" Satko

r2con2025

Who am I?

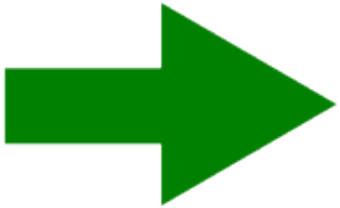
- Rzeszów University of Technology graduate
- Casual contributor to r2

What will be covered

- How parse plugins work
- 5.9 -> 6.x Parse API changes
- Example of applying the new API
- Future ideas

How parse plugins work

```
[0x00001070]> s main
[0x00001248]> af
[0x00001248]> pdf
```



Parse Plugin

```
0x00001248      55              push rbp
0x00001249      4889e5          mov rbp, rsp
0x0000124c      4883ec20        sub rsp, 0x20
0x00001250      488d05ad0d..    lea rax, str.teeest
0x00001257      488945f0        mov qword [var_10h], rax
0x0000125b      c745ec0a00..    mov dword [var_14h], 0xa
0x00001262      8345ec05        add dword [var_14h], 5
0x00001266      8b45ec          mov eax, dword [var_14h]
0x00001269      89c7            mov edi, eax
0x0000126b      e8f9feffff      call sym.test
0x00001270      89c6            mov esi, eax
0x00001272      488d05920d..    lea rax, str.test:__d
```

Previous Parse API

- Was too ambigious
- **Didn't support any data storage**

5.9 -> 6.x

RParsePlugin -> **RAsmPlugin**

```
typedef struct r_asm_plugin_t {
    RPluginMeta      meta;
    RAsmParseInit    init;
    RAsmParseFini    fini;
    RAsmParsePseudo  parse;
    RAsmParseFilter  filter;
    RAsmParseSubvar  subvar;
    RAsmParsePatch   patch;
} RAsmPlugin;
```

RPluginMeta

Information about a plugin

```
typedef struct r_plugin_meta_t {
    char *name;
    char *desc;
    char *author;
    char *version;
    char *license;
    char *contact;
    char *copyright;
    RPluginStatus status;
} RPluginMeta;
```

RAsmParseInit

```
void RAsmParseInit(RAsmPluginSession *s)
```

Initialization of a plugin

```
typedef struct r_asm_plugin_session_t {
    struct r_asm_t *rasm;
    struct r_asm_plugin_t *plugin;
    void *data;
} RAsmPluginSession;
```

RAsmParseFini

```
void RAsmParseFini(RAsmPluginSession *s)
```

Finalization of a plugin

RAsmParsePseudo

```
char *RAsmParsePseudo(RAsmPluginSession *s, const char *data)
```

Converts a raw disassembly text (or aop.mnemonic) like: ***mov a, #1*** into a high-level pseudocode string, e.g. ***a = 1***.

By setting ***.parse*** attribute one can modify its behavior.

RAsmParseFilter

```
char *RAsmParseFilter(RAsmPluginSession *s, ut64 addr,
                      RFlag *f, const char *data)
```

Normalizes/pretty-prints the disasm string, e.g.:

- Flip AT&T <-> Intel operand order for x86
- Lowercase mnemonics, strip whitespaces, etc.

RAsmParseSubvar

```
char *RAsmParseSubvar(RAsmPluginSession *s, RAnalFunction *f,
                      ut64 addr, int oplen, const char *data)
```

Replaces literals or placeholders with names coming from analysis: **flags, function names, local/arg names, string literals**, etc.

RAsmParsePatch

```
char *RAsmParsePatch(RAsmPluginSession *s, RAnalOp *aop,
                     const char *newop)
```

Maps a user-facing textual change to something writable (e.g.: *wa* <*asm*> or raw hex bytes).

Enabled when executing **patch** or **write** command.

Hands-on example

`https://github.com/radareorg/radare2-extras.git`

Future ideas

- Rename *parse* to *pseudo*
- Make exchanging data between other plugin types more accessible
- Make afen support renaming strings and symbols thanks to ***subvar***
- Multi instruction support (pack N instructions into 1)
- Support import/export/dump/restore the afen rules into an r2 script and json file
- Add testsuite + fuzzing for bugs
- And many more... (ask pancake)

Thank you all!