AFLSMART Fuzzer Analysis

AFLSMART Fuzzer

- Based on AFL and input structure component of Peach
- Smart greybox fuzzing - Coverage based greybox fuzzing with input structure awareness
- Smart mutation operators
- Validity based power schedule

Analysis

- AFLSMART can be modified to accept .ksy files instead of Peach pit files.

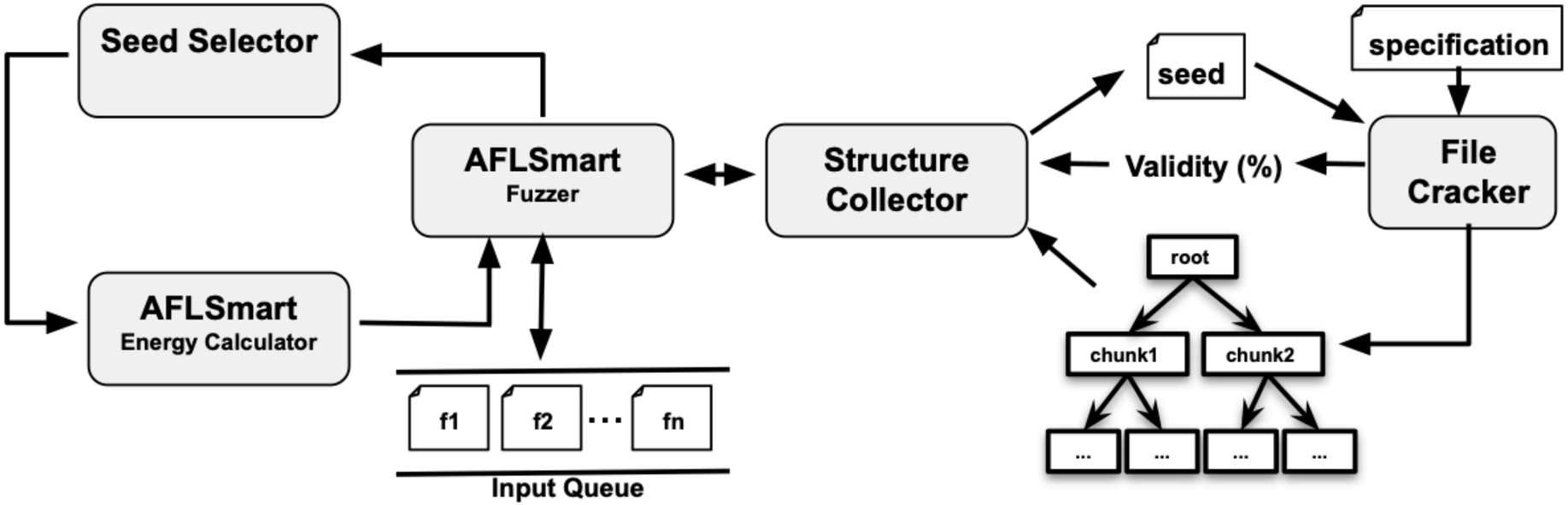

Fig. 6: Architecture of AFLSMART

AFLSMART File Cracker

- It parses an input file and de-composes it into data chunks and data attributes.
- It also calculates the validity of the input file based on how much of the file can be parsed.
- Implemented by modifying the Cracker component of the smart blackbox fuzzer Peach.

Function in AFLSMART

```
static void update_input_structure(u8* fname, struct queue_entry* q) {
  pid_t pid = 0;
  int pipefd[2];
  FILE* output;
  char line[256];
  int status;
  u8* ifname;
  u8* ofname;

  if (model_type == MODEL_PEACH) {

    if (pipe(pipefd) < 0) {
      PFATAL("AFLSmart cannot create a pipe to communicate with Peach");
      exit(1);
    }

    pid = fork();
    if (pid == 0) {
      close(pipefd[0]);
      dup2(pipefd[1], STDOUT_FILENO);
      dup2(pipefd[1], STDERR_FILENO);
      ifname = alloc_printf("-inputFilePath=%s", fname);
      ofname = alloc_printf("-outputFilePath=%s/chunks/%s.repaired", out_dir,
                            basename(fname));
      execlp("peach", "peach", "-1", ifname, ofname, input_model_file, (char*) NULL);
      exit(1); /* Stop the child process upon failure. */
    } else {
      close(pipefd[1]);
      output = fdopen(pipefd[0], "r");
```

```
  while (fgets(line, sizeof(line), output)) {
    /* Extract validity percentage and update the current queue entry. */
    q->validity = 0;
    if (!strncmp(line, "ok", 2)) {
      q->validity = 100;
      break;
    } else if (!strncmp(line, "error", 5)) {
      char *s = line + 5;
      while (isspace(*s)) { s++; }
      char *start = s;
      while (isdigit(*s)) { s++; }
      *s = '\0';
      if (s != start) {
        q->validity = (u8) atoi(start);
      }
      break;
    }
  }

  waitpid(pid, &status, 0);

  u8* chunks_fname = alloc_printf("%s/chunks/%s.repaired.chunks", out_dir, basename(fname));
  struct chunk *chunk;

  get_chunks(chunks_fname, &chunk);
  q->chunk = chunk;
  q->cached_chunk = copy_chunks(chunk);

  fclose(output);
  ck_free(chunks_fname);
}
```

```
  } else {
    /// NOT SUPPORTED
    PFATAL("AFLSmart currently only supports Peach models! Please use -w peach option");
  }

  parsed_inputs++;
  validity_avg += (s8)(q->validity - validity_avg) / parsed_inputs;
  q->parsed = 1;
}
```

How to modify?

- Implement a parser for KSY files that can read the file format and generate a tree structure.
- Replace the Peach-specific code in **update_input_structure()** with the logic to parse the input KSY file using your parser.
- Modify the code to handle the parsed tree structure appropriately, updating the queue entry and validity calculation based on the parsed data.