CI-Fuzz Proof of Concept

E3SP Security

11 August, 2025

Table of Contents

1 Prologue 26
1.1 Purpose of this document 26
1.2 Scope 26
1.3 Out of Scope 26
2 Introduction to Fuzzing 27
2.1 Motivation 27
2.2 Fuzzing 28
2.2.1 What is fuzzing? 28
2.2.2 How does fuzzing work? 28
2.2.3 Types of fuzzer 30
2.2.4 Stateless and statefull SUT while fuzzing 30
2.3 Code-Intelligence and CIFUZZ 31
3 The fuzzing process simplified 35
3.1 Local build 35
3.2 Local run 37
3.3 Coverage 37
3.3.1 Local 37
3.4 Debug 38
3.4.1 Local 38
4 The fuzzing process in detail 39
4.1 General workflow (in progress) 39
4.2 What to fuzz? 40
4.2.1 Risk driven fuzzing process 40
4.2.2 Types of fuzz targets 43
4.2.3 What is a good target? 44
4.2.4 How to document what to fuzz with Doxygen? 45
4.3 Setup – Installation 48
4.3.1 PMT build process (for cifuzz maintainer) 49
4.3.2 How to add fuzzing option to the current PMT build process? (for cifuzz maintainer) 50

4.3.3 Prepare your VM ... 63
4.3.4 Setting up cifuzz environment ... 66
4.4 Creation of fuzz target ... 72
4.5 Maintaining a fuzz target ... 75
4.5.1 Handling Fuzz Targets expecting different inputs ... 75
4.5.2 Improve by adding a switch case ... 77
4.5.3 Improve by setting cifuzz runtime flag ... 78
4.5.4 Improve fuzzing via dictionary ... 78
4.5.5 Improve fuzzing via Value Profile ... 79
4.5.6 Improve fuzzing via preparing corpus ... 80
4.5.7 Improve fuzzing via Custom Mutator ... 80
4.5.8 Mocking ... 81
4.5.9 Debugging a fuzz target ... 88
4.6 Fuzzing on CICD pipeline ... 92
4.6.1 Building block 1: tools-bb-fuzzing-src ... 93
4.6.2 Building block 2: tools-bb-fuzzing-report-src ... 94
4.7 Reporting fuzzing-findings / bugs ... 94
4.7.1 Manually creating a Jira bug ticket ... 94
4.7.2 Via CICD ... 100
5 Practical Examples ... 101
6 Tips, Tricks and Best Practices ... 102
7 Download ... 104
8 Abbreviation & Terminology ... 105
9 References ... 106
10 TODO ... 108
11 (Deprecated) Archive ... 109
11.1 Working with CI-Sense Server ... 109
11.1.1 Server Information ... 109
11.1.2 Creating and uploading a bundle to the server ... 109
11.1.3 CICTL tool ... 112
12 (Deprecated) CI Fuzz Integration/Training ... 114
12.1 Agenda, Purpose and Training Materials ... 114

12.1.1 DAST in this case Fuzzing on Architecture level (Draft) 114
12.2 Introduction to Fuzzing 115
12.2.1 What is fuzzing 115
12.2.2 Motivation 116
12.2.3 Overview Fuzzing 116
12.2.4 Introduction CI Fuzz 118
12.2.5 Fuzzing Workflow 119
12.3 Dockerized CI Fuzz 120
12.3.1 Pulling the cifuzz docker image from artefactory 121
12.3.2 Prepare your component folder for cifuzz 123
12.3.3 Implement a fuzzing test in the stub file 125
12.3.4 Run you first fuzz test 126
12.4 Fuzzing your E3 project with CI Fuzz 126
12.4.1 Creation of a fuzz target 126
12.4.2 Maintaining a fuzz target 136
12.4.3 Debugging a fuzz target 137
12.5 Deploying fuzz target to the CIFuzz Server 144
12.5.1 Server Information 144
12.5.2 Creating and uploading the bundle to the server 145
12.6 Outlook 148
12.6.1 Server side fuzzing 148
12.6.2 SDK Integration 149
12.7 References & Further Reading 150
12.8 Downloads 151
13 (Deprecated) Process of fuzzing & release criteria/definition of done (in progress) 152
13.1 Abbreviations 152
13.2 Agenda & Purpose 152
13.3 Timeline 152
13.4 Good metrics 153
13.5 Process of fuzzing and exit criteria 154
13.5.1 Release criteria 154
13.6 Open Questions 155

13.6.1 How to extract a list of security relevant component from SRS 156
14 CI-Fuzz List of Onboarding Teams 158
14.1 CI Fuzz on-boarding sessions and components status 158
14.2 Onboarding slides 210
14.3 2023.07.04 Onboarding session MID / MACRO / MICRO Stack 211
14.3.1 Participants 211
14.3.2 Protocol 213
14.3.3 FAQ 214
14.4 2023.07.11 Onboarding session SOCD-Babelfish/Comet 215
14.4.1 Participants 215
14.4.2 Protocol 216
14.4.3 FAQ 216
14.5 2023.07.12 Onboarding session MID Stack 218
14.5.1 Participants 218
14.5.2 Protocol 219
14.5.3 FAQ 224
14.6 2023.07.14 Fuzzing Activity for LUM 224
14.6.1 Participants 224
14.6.2 Protocol 224
14.6.3 FAQ 224
14.7 2023.07.14 Security Activities on MACRO Level 224
14.7.1 Participants 224
14.7.2 Protocol 225
14.7.3 FAQ 226
14.8 2023.07.19 Sync session regarding fuzzing with MICRO Stack 226
14.9 2023.07.20 Hands-on session CodeBuster 228
14.9.1 Participants 228
14.9.2 Protocol 228
14.9.3 FAQ 229
14.9.4 Notes 229
14.10 2023.07.24 Sync Session with MACRO Stack 229
14.10.1 Participants 229

14.10.2 Protocol 230
14.10.3 FAQ 231
14.10.4 Notes 231
14.11 2023.07.25 Summary of Stacks language usage and component type 231
14.11.1 Participants 231
14.12 2023.08.08 Sync and Hands-On session TLR 232
14.12.1 Participants 232
14.12.2 Agenda 2023.08.08 232
14.12.3 Protocol 2023.08.08 232
14.12.4 Minutes 2023.08.08 232
14.12.5 Minutes 2023.08.10 Follow-up 233
14.12.6 FAQ 2023.08.08 233
14.12.7 FAQ 2023.08.10 233
14.12.8 Interesting Links 234
14.13 2023.08.09 Sync and Hands-On session UDSM 234
14.13.1 Participants 234
14.13.2 Agenda 235
14.13.3 Protocol 235
14.13.4 Minutes 235
14.13.5 FAQ 235
14.13.6 TODOs 236
14.13.7 Interesting Links 236
14.14 2023.08.21 3rd Hands-On session TLR 237
14.14.1 Participants 237
14.14.2 Interesting Links 238
14.15 2023.08.23 2nd Hands-On session UDSM 238
14.15.1 Participants 238
14.15.2 Agenda 239
14.15.3 Protocol 239
14.15.4 FAQ 239
14.15.5 TODOs 239
14.15.6 Interesting Links 240

14.16 2023.08.29 1st Hands-On session DiagMaster 240
14.16.1 Participants 240
14.16.2 Agenda 240
14.16.3 Protocol 240
14.16.4 Q&A 241
14.16.5 TODOs 242
14.16.6 Interesting Links 242
14.17 2023.10.20 Sync MICRO STACK Fuzzing 1.1, 1.2 242
14.17.1 Participants 242
14.17.2 Agenda 243
14.17.3 Protocol 243
14.17.4 Q&A 243
14.17.5 TODOs 243
14.17.6 Interesting Links 243
14.18 2023.10.26 Sync OTA Master Fuzzing 1.2 244
14.18.1 Participants 244
14.18.2 Agenda 244
14.18.3 Protocol 245
14.18.4 Q&A 245
14.18.5 TODOs 246
14.18.6 Interesting Links 246
14.19 2023.11.14 Hands-on OTA Master Fuzzing 1.2 246
14.19.1 Participants 246
14.19.2 Agenda 246
14.19.3 Protocol 247
14.19.4 List of fuzz targets for OTA Master 248
14.19.5 Q&A 250
14.19.6 TODOs 250
14.19.7 Interesting Links 251
14.20 2023.11.24 Sync fuzzing topic team DRM 251
14.20.1 Participants 251
14.20.2 Agenda 251

14.20.3 Protocol 252
14.20.4 Q&A 252
14.20.5 TODOs 252
14.20.6 Interesting Links 252
14.21 2023.12.14 Sync Sanitizer Enabled Fuzzing 252
14.21.1 Participants 252
14.21.2 Agenda 253
14.21.3 Protocol 253
14.21.4 TODOs 253
14.21.5 Interesting Links 253
14.22 2024.01.08 Sync Sanitizer Enabled Fuzzing 253
14.22.1 Participants 253
14.22.2 Agenda 254
14.22.3 Protocol 254
14.22.4 TODOs 254
14.22.5 Interesting Links 255
14.23 2024.01.11 Sync fuzzing topic team DRM (2. Part) 255
14.23.1 Participants 255
14.23.2 Agenda 255
14.23.3 Protocol 255
14.23.4 Q&A 256
14.23.5 TODOs 256
14.23.6 Interesting Links 256
14.24 2024.01.12 Sync fuzzing topic team Remote Repair (1. Part) 257
14.24.1 Participants 257
14.24.2 Agenda 257
14.24.3 Protocol 258
14.24.4 Q&A 258
14.24.5 TODOs 258
14.24.6 Interesting Links 259
14.25 2024.01.15 Sync fuzzing topic TM SP 259
14.25.1 Participants 259

14.25.2 Agenda 260
14.25.3 Protocol 260
14.25.4 Q&A 261
14.25.5 TODOs 262
14.25.6 Interesting Links 262
14.26 2024.01.17 Sync fuzzing topic TMBase & TMObd 262
14.26.1 Participants 262
14.26.2 Agenda 263
14.26.3 Protocol 263
14.26.4 Q&A 264
14.26.5 TODOs 264
14.26.6 Interesting Links 264
14.27 2024.01.30 Sync fuzzing topic RPC+ 264
14.27.1 Participants 264
14.27.2 Agenda 265
14.27.3 Protocol 265
14.27.4 Q&A 266
14.27.5 TODOs 266
14.27.6 Interesting Links 266
14.28 2024.02.01 Sync fuzzing topic Lost Pilot 267
14.28.1 Participants 267
14.28.2 Agenda 267
14.28.3 Q&A 268
14.28.4 TODOs 268
14.28.5 Interesting Links 269
14.29 2024.02.09 Sync fuzzing topic RPC+ regarding effort for fuzzing 269
14.29.1 Participants 269
14.29.2 Agenda 269
14.29.3 Protocol 270
14.29.4 Q&A 270
14.29.5 TODOs 270
14.29.6 Interesting Links 271

14.30 2024.02.13 Hands-on creating fuzz targets for Lost Pilot 271
14.30.1 Participants 271
14.30.2 Minutes 271
14.30.3 Q&A 272
14.30.4 TODOs 272
14.30.5 Interesting Links 272
14.31 2024.02.19 Sync fuzzing topic TM SP 272
14.31.1 Participants 272
14.31.2 Agenda 273
14.31.3 Protocol 273
14.31.4 Q&A 274
14.31.5 TODOs 274
14.31.6 Interesting Links 274
14.32 2024.02.20 Fuzzing session with TMBase 274
14.32.1 Participants 274
14.32.2 Agenda 274
14.32.3 Protocol 274
14.32.4 Q&A 275
14.32.5 TODOs 275
14.32.6 Interesting Links 275
14.33 2024.02.20 Hands-on creating fuzz targets for Lost Pilot 2 275
14.33.1 Participants 275
14.33.2 Minutes 275
14.33.3 Q&A 276
14.33.4 TODOs 276
14.33.5 Interesting Links 276
14.34 2024.02.23 Hands-on creating fuzz targets for Lost Pilot 3 277
14.34.1 Participants 277
14.34.2 Minutes 277
14.34.3 Q&A 277
14.34.4 TODOs 278
14.34.5 Interesting Links 278

14.35 2024.03.01 Support session Lost Pilot 278
14.35.1 Participants 278
14.35.2 Minutes 278
14.35.3 Q&A 279
14.35.4 TODOs 279
14.35.5 Interesting Links 279
14.36 2024.03.05 Sync fuzzing topic team FlashMob (VUM) 279
14.36.1 Participants 279
14.36.2 Agenda 279
14.36.3 Protocol 279
14.36.4 Q&A 282
14.36.5 TODOs 282
14.36.6 Interesting Links 282
14.37 2024.03.06 Sync fuzzing topic TM SP 283
14.37.1 Participants 283
14.37.2 Agenda 283
14.37.3 LIST OF FUZZ RELEVANT FUNCTIONS 283
14.37.4 TODOs 283
14.37.5 Interesting Links 284
14.38 2024.03.08 Sync fuzzing topic TMBase & TMObd 284
14.38.1 Participants 284
14.38.2 Agenda 284
14.38.3 Protocol 284
14.38.4 Q&A 284
14.38.5 TODOs 284
14.38.6 Interesting Links 284
14.39 2024.03.12 Sync fuzzing topic team Remote Repair (2. Part) 285
14.39.1 Participants 285
14.39.2 Agenda 285
14.39.3 Protocol 285
14.39.4 Q&A 285
14.39.5 TODOs 285

14.39.6 Interesting Links 286
14.40 2024.03.15 [DRM] Session with Rahul to clarify fuzzing related questions 286
14.40.1 Participants 286
14.40.2 FYI 286
14.40.3 Minutes & Questions 286
14.40.4 TODOs 287
14.40.5 Interesting Links 287
14.41 2024.03.27 [TMBase & TMObd] Setting up the dev environment for fuzzing 288
14.41.1 Participants 288
14.41.2 FYI 288
14.41.3 Minutes 288
14.41.4 Q&A 288
14.41.5 TODOs 288
14.41.6 Interesting Links 288
14.42 2024.04.05 [TMBase & TMObd] Sync with Oliviera 289
14.42.1 Participants 289
14.42.2 FYI 289
14.42.3 Minutes 289
14.42.4 Q&A 289
14.42.5 TODOs 289
14.42.6 Interesting Links 290
14.43 2024.04.11 [TMBase & TMObd] Review of Fuzz-Target List 290
14.43.1 Participants 290
14.43.2 FYI 290
14.43.3 Minutes 291
14.43.4 Q&A 291
14.43.5 TODOs 291
14.43.6 Interesting Links 291
14.44 2024.04.16 Sync session regarding OTA-Master Story Points in PI 24.2 291
14.44.1 Participants 291
14.44.2 Minutes 292
14.44.3 Result 292

14.44.4 Q&A 292
14.44.5 TODOs 292
14.44.6 Interesting Links 292
14.45 2024.04.17 [DRM] Sync session regarding DRM's Story Points in PI 24.2 293
14.45.1 Participants 293
14.45.2 Minutes 293
14.45.3 Result 293
14.45.4 Q&A 293
14.45.5 TODOs 293
14.45.6 Interesting Links 294
14.46 2024.04.22 Sync fuzzing topic team FlashMob (VUM) Part 2: Building and defining targets. 294
14.46.1 Participants 294
14.46.2 Agenda 294
14.46.3 Protocol 294
14.46.4 Q&A 295
14.46.5 TODOs 295
14.46.6 Interesting Links 295
14.47 2024.05.02 Sync fuzzing topic [TMSP] 295
14.47.1 Participants 295
14.47.2 FYI 296
14.47.3 Agenda 296
14.47.4 LIST OF FUZZ RELEVANT FUNCTIONS 296
14.47.5 TODOs 296
14.47.6 Interesting Links 296
14.48 2024.05.08 Analyse Risk of Buffer Overflow in TLR 296
14.48.1 Participants 296
14.48.2 FYI 297
14.48.3 Agenda 297
14.48.4 TODOs 297
14.49 2024.05.14 [DRM] Sync session regarding DRM's Story Points in PI 24.2 Part 2 298
14.49.1 Participants 298
14.49.2 FYI 298

14.49.3 Minutes 298
14.49.4 Result 298
14.49.5 Q&A 298
14.49.6 TODOs 298
14.49.7 Interesting Links 298
14.50 2024.05.14 Analyse Risk of Buffer Overflow in TLR for HCP3 Part 2 299
14.50.1 Participants 299
14.50.2 FYI 299
14.50.3 Analysis 299
14.50.4 Result 302
14.51 2024.06.03 [TMBase] Discussion on where to store corpus for TMBase 302
14.51.1 Participants 302
14.51.2 FYI 302
14.51.3 Minutes 302
14.51.4 Result 303
14.51.5 TODOs 303
14.51.6 Interesting Links 303
14.52 2024.06.24 Sync with Team Remote Repair regarding upcoming task in PI 24.3 303
14.52.1 Participants 303
14.52.2 Agenda 303
14.52.3 Minutes 307
14.52.4 Result 307
14.52.5 TODOs 307
14.52.6 Interesting Links 307
14.53 2024.06.28 Sync with Team [Flashmob] - Alternatives to fuzzing 307
14.53.1 Participants 307
14.53.2 FYI 308
14.53.3 Agenda 308
14.53.4 Minutes 308
14.53.5 Result 309
14.53.6 TODOs 309
14.53.7 Interesting Links 309

14.54 2024.07.08 Sync with Team [TMSP-TimeTravelers] 309
14.54.1 Participants 309
14.54.2 FYI 310
14.54.3 Agenda 310
14.54.4 Minutes 310
14.54.5 TODO 310
14.54.6 Links 310
14.55 2024.07.10 Refinement of Targets List [TimeTravelers- TMBase] 311
14.55.1 Participants 311
14.55.2 FYI 311
14.55.3 Agenda 311
14.55.4 Interesting Links 311
14.56 2024.07.18 Onboarding Radu Troubleshoot conan issue 311
14.56.1 Participants 311
14.56.2 Agenda 312
14.56.3 Interesting Links 313
14.57 2024.07.30 Sync with Team Remote Repair regarding Refining linking of Dataflows and code 313
14.57.1 Participants 313
14.57.2 FYI 313
14.57.3 Agenda 313
14.57.4 Minutes 316
14.57.5 TODOs 316
14.57.6 Interesting Links 316
14.57.7 Information form previous meetings 316
14.58 2024.08.20 Sync with Team Remote Repair Fuzzing Requirement 318
14.58.1 Participants 318
14.58.2 FYI 318
14.58.3 Information 318
14.58.4 Minutes 320
14.58.5 TODOs 321
14.58.6 Interesting Links 321

14.59 2024.09.20 Fuzzing Activites recap for 24.3 322
14.59.1 Remote Repair 322
14.59.2 TM Base 322
14.59.3 TM OBD 322
14.59.4 TM SP 322
14.59.5 LostPilot 322
14.59.6 Team Adaptive 323
14.59.7 RPC+ 323
14.60 2024.09.30 Discuss E3-122194 - DiagMaster Risk 323
14.60.1 Participants 323
14.60.2 Agenda 323
14.60.3 Minutes 323
14.61 2024.10.07 Discuss E3-122194 - DiagMaster Risk 324
14.61.1 Participants 324
14.61.2 FYI 324
14.61.3 Agenda 324
14.61.4 Minutes 324
14.61.5 TODO 325
14.62 2024.10.15 SW Requirement interface-hardening / Fuzz Testing Remote Repair 325
14.63 2024.10.24 Sync with Team DiagNinja regarding fuzzing 326
14.63.1 Participants 326
14.63.2 FYI 326
14.63.3 Agenda 326
14.63.4 Minutes 327
14.63.5 TODO 327
14.63.6 Links 328
14.64 2024.11.13 Sync with Team Adaptive regarding fuzzing status 328
14.64.1 Participants 328
14.64.2 FYI 328
14.64.3 Agenda 328
14.64.4 Minutes 329
14.64.5 TODO 329

14.64.6 Links ... 329
14.65 2024.11.21 Sync with Team TM SP (Adaptive Autosar) and TM Obd (Classic Autosar) regarding fuzzing status ... 329
14.65.1 Participants ... 329
14.65.2 FYI ... 330
14.65.3 Agenda ... 330
14.65.4 Minutes ... 330
14.65.5 TODO ... 331
14.65.6 Links ... 331
14.66 2024.11.28 Hands-on session with Team DiagNinja regarding fuzzing ... 332
14.66.1 Participants ... 332
14.66.2 FYI ... 332
14.66.3 Agenda ... 332
14.66.4 Minutes ... 333
14.66.5 TODO ... 335
14.66.6 Links ... 335
14.67 2024.12.02 Sync with Martin OTA-Master SRA extraction ... 335
14.67.1 Participants ... 335
14.67.2 Agenda ... 335
14.67.3 Minutes ... 336
14.67.4 TODO ... 336
14.68 2024.12.05 Sync with OTA-Master Code to Data flow mapping and Requirements ... 336
14.68.1 Participants ... 336
14.68.2 Agenda ... 337
14.68.3 Minutes ... 337
14.68.4 Data Flows ... 337
14.68.5 otam_tm ... 337
14.68.6 otam_vsm ... 342
14.69 2024.12.10 Sync with Code Spiders ... 345
14.69.1 Participants ... 345
14.69.2 Agenda ... 346
14.69.3 Minutes ... 346

14.69.4 Link ... 346
14.70 2025.02.17 Sync/onboarding with Team CodeSpiders regarding fuzzing ... 346
14.70.1 Participants ... 346
14.70.2 FYI ... 347
14.70.3 Agenda ... 347
14.70.4 Minutes ... 347
14.70.5 TODO ... 347
14.70.6 Links ... 347
15 CI-Sparc Fuzzing Guideline to generate Fuzzing test with LLM ... 348
16 CI-Sparc Fuzzing Presentation to Management Slides ... 349
16.1 1. Overview ... 349
16.2 2. Term Definitions ... 349
16.3 3. Fuzzing Relevant Criteria ... 350
16.3.1 3.1. Do Fuzz ... 350
16.3.2 3.2. Don't Fuzz ... 350
16.4 4. Guidelines to generate a list of fuzzing tests ... 351
16.4.1 4.1. Create your own list of fuzzing items ... 351
16.5 5. Documentation of Fuzzing Relevant tests ... 351
16.5.1 5.1. Motivation why we use LLM ... 351
17 CISpark/CIFuzz generate ... 352
17.1 Official Documentation ... 352
17.2 Architecture ... 352
17.3 Config with Azure OpenAI ... 352
17.3.1 CiFuzz generate Validate ... 354
17.4 Run with local LLMs ... 357
17.4.1 Compile Llama.cpp ... 357
17.4.2 Run the LLM Server: ... 357
17.4.3 Configure Cifuzz spark environmental variables: ... 358
17.5 Troubleshooting ... 359
17.5.1 Problem CMAKE_C_COMPILE_OBJECT ... 359
17.5.2 Out of Memory for 32B models in WSL ... 360
18 Fuzzing Pipeline incl. JIRA bug reporting with Jenkins ... 361

18.1 2023-12-15 Sync Fuzzing Pipeline with Jenkins 374
18.2 2024-01-17 Sync Fuzzing Pipeline with Jenkins Part 2 376
18.3 2024-05-27 Discuss running Fuzz-Targets directly in Jenkins 379
18.3.1 Participants 379
18.3.2 FYI 379
18.3.3 Agenda 379
18.3.4 TODOs 381
18.3.5 Helpful Link 381
18.4 2024-06-03 Discuss running Fuzz-Targets directly in Jenkins - Planning 381
18.4.1 Participants 381
18.4.2 FYI 382
18.4.3 Agenda 382
18.4.4 Information from past Discussion 383
18.4.5 TODOs 384
18.5 2024-07-05 Discuss Building Blocks 386
18.5.1 Participants 386
18.5.2 FYI 387
18.5.3 Agenda 387
18.6 2024-07-23 Discuss current cifuzz build process and merging of 2 PRs 388
18.6.1 Participants 388
18.6.2 FYI 388
18.6.3 Agenda 388
18.6.4 Protocol 389
18.6.5 Links 389
18.6.6 TODO 390
18.7 2024-08-05 Sync regarding future work for Fuzzing Overview page 390
18.7.1 Participants 390
18.7.2 FYI 390
18.7.3 Agenda 390
18.7.4 Protocol 390
18.7.5 Links 391
18.7.6 TODO 392

18.8 2024-08-12 Sync Fuzzing Pipeline with Jenkins Part 3 ... 392
18.9 2024-08-13 Sync Fuzzing Pipeline with Jenkins Part 4 - System Build & Building Block ... 393
18.10 2024-08-30 Sync regarding BB-reporting ... 395
18.10.1 Participants ... 395
18.10.2 Agenda ... 395
18.10.3 Protocol ... 395
18.10.4 Links ... 396
18.10.5 TODO ... 396
18.11 2024-09-05 Sync Fuzzing Pipeline with Jenkins Part 5 - Pipeline and Building Block Status ... 397
18.12 2024-09-05 Sync Fuzzing Pipeline with Jenkins Part 6 - Pipeline and Building Block Status ... 400
18.13 2024-09-16 Sync Fuzzing Pipeline with Jenkins Part 7 - Pipeline and Building Block Status ... 403
18.14 2024-09-20 Sync Fuzzing Pipeline with Jenkins Part 8 - Pipeline and Building Block Status ... 408
18.15 2024-09-25 Sync Fuzzing Pipeline with Jenkins Part 9 - Pipeline and Building Block Status ... 411
18.15.1 Agenda: ... 412
18.15.2 Decisions so far: ... 412
18.16 2024-10-02 Sync Fuzzing Pipeline with Jenkins Part 10 - Pipeline and Building Block Status ... 413
18.16.1 Agenda ... 413
18.16.2 Decisions so far ... 414
18.16.3 TODO ... 415
18.17 2024-10-16 Sync Fuzzing Pipeline with Jenkins Part 11 - Pipeline and Building Block Status ... 415
18.17.1 Agenda ... 415
18.17.2 Meeting minutes ... 416
18.17.3 TODO ... 417
18.18 2024-10-22 Sync regarding visualization of fuzzing result from pipeline ... 418
18.18.1 Participants ... 418
18.18.2 Agenda ... 418
18.18.3 Protocol ... 418
18.18.4 Links ... 420
18.18.5 TODO ... 420
18.19 2024-10-24 Sync Fuzzing Pipeline with Jenkins Part 12 - Pipeline and Building Block Status ... 421

18.19.1 Agenda ... 421
18.19.2 Meeting minutes ... 422
18.19.3 TODO ... 423
18.20 2024-11-06 Sync Fuzzing Pipeline with Jenkins Part 13 - Pipeline and Building Block Status ... 424
18.20.1 TODOs ... 424
18.20.2 Agenda ... 425
18.20.3 Meeting minutes ... 425
18.20.4 TODO ... 427
18.21 2024-11-20 Sync Fuzzing Pipeline with Jenkins Part 15 - Pipeline and Building Block Status ... 428
18.21.1 TODOs ... 428
18.21.2 Agenda ... 429
18.21.3 Meeting minutes ... 429
18.21.4 TODO ... 433
19 Fuzzing Presentation to Management Slides ... 434
20 Q&A and Troubleshooting CIFuzz ... 436
20.1 ICP ART ... 436
20.1.1 LUM Local Update Manager ... 436
20.1.2 Adaptive Adaptive Stack (EB) ... 436
20.1.3 DiagMaster Serviceorientierte Diagnose ... 436
20.2 BSrv ... 437
20.2.1 CodeBuster Function on demand (2.0) ... 437
20.2.2 LostPilot ... 438
20.3 General Troubleshooting ... 439
20.3.1 Docker Related ... 439
20.3.2 Questions regarding building and error fixing ... 440

status	version	date modified	plan part owner	scope
In Progress	85 (see page 22)	2025-02-15 04:51	Neeb, Daniel (T5-SG)	

VW .os	org unit
2.0	Solution, BSrv, ComSec, EDGE, MICRO, MID, MACRO, VW.os Core
1.2	VW.os Core
1.1	not applicable

further meta information ...

general page information	
page ID	1483434491
page created by / on	Dang, Quang Anh (EXTERN: ITK)[1] / 2023-05-05 17:13
page editors	• Dang, Quang Anh (EXTERN: ITK)[2] • Muckenschnabl, David Jose (EXTERN: ITK)[3] • Kutscher, Max (EXTERN: PTG)[4] • Klug, Michael (EXTERN: Webware-Expe)[5]
page commenters	• Dang, Quang Anh (EXTERN: ITK)[6]
used template	Template: Plan Part[7]

1 https://devstack.vwgroup.com/confluence/display/~gchogqb
2 https://devstack.vwgroup.com/confluence/display/~gchogqb
3 https://devstack.vwgroup.com/confluence/display/~t42mhf2
4 https://devstack.vwgroup.com/confluence/display/~tr742ps
5 https://devstack.vwgroup.com/confluence/display/~wag00z7
6 https://devstack.vwgroup.com/confluence/display/~gchogqb
7 https://devstack.vwgroup.com/confluence/x/j_W1Ig

used template version	39
page classification	
classification	Unable to render include or excerpt-include. Could not retrieve page.
confidentiality	Unable to render include or excerpt-include. Could not retrieve page.
approval information	
last approval version	73
last approval date	Jul 10, 2024
last approval by	gchogqb
change history	

Version	Published	Changed By	Comment
CURRENT (see page 22) **(v. 85)**	Feb 15, 2025 04:51	Dang, Quang Anh (EXTERN: ITK)[8] Klug, Michael (EXTERN: Webware-Expe)[9]	Finished the last TODOs
v. 84[10]	Dec 02, 2024 04:48	Dang, Quang Anh (EXTERN: ITK)[11]	Finished the last TODOs
v. 83[12]	Dec 02, 2024 04:37	Dang, Quang Anh (EXTERN: ITK)[13]	- Updated link to APfuzzing.cmake
v. 82[14]	Dec 02, 2024 01:40	Dang, Quang Anh (EXTERN: ITK)[15]	Added changes about doxygen code marking
v. 81[16]	Nov 20, 2024 03:00	Dang, Quang Anh (EXTERN: ITK)[17]	Added information about CICD Pipeline and Building blocks
v. 80[18]	Nov 06, 2024 14:49	Dang, Quang Anh (EXTERN: ITK)[19] Muckenschnabl, David Jose (EXTERN: ITK)[20]	added TODO for fuzzing profile and updated build command
v. 79[21]	Sep 05, 2024 15:33	Muckenschnabl, David Jose (EXTERN: ITK)[22]	
v. 78[23]	Aug 21, 2024 15:37	Kutscher, Max (EXTERN: PTG)[24]	

8 https://devstack.vwgroup.com/confluence/display/~gchogqb
9 https://devstack.vwgroup.com/confluence/display/~wag00z7
10 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=4728449293
11 https://devstack.vwgroup.com/confluence/display/~gchogqb
12 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=4308877940
13 https://devstack.vwgroup.com/confluence/display/~gchogqb
14 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=4308877903
15 https://devstack.vwgroup.com/confluence/display/~gchogqb
16 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=4308876429
17 https://devstack.vwgroup.com/confluence/display/~gchogqb
18 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=4234985387
19 https://devstack.vwgroup.com/confluence/display/~gchogqb
20 https://devstack.vwgroup.com/confluence/display/~t42mhf2
21 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=4148809392
22 https://devstack.vwgroup.com/confluence/display/~t42mhf2
23 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=3793907057
24 https://devstack.vwgroup.com/confluence/display/~tr742ps

Version	Published	Changed By	Comment
v. 77[25]	Aug 16, 2024 17:43	Kutscher, Max (EXTERN: PTG)[26]	
v. 76[27]	Aug 14, 2024 17:24	Kutscher, Max (EXTERN: PTG)[28]	

Go to Page History[29]

25 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=3700681077
26 https://devstack.vwgroup.com/confluence/display/~tr742ps
27 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=3677676351
28 https://devstack.vwgroup.com/confluence/display/~tr742ps
29 https://devstack.vwgroup.com/confluence/pages/viewpreviousversions.action?pageId=1483434491

1 Prologue

1.1 Purpose of this document

This document is a summary of work done sofar regarding fuzz-testing.

It aims to provide documentation to setup fuzzing into an existing testing process.

This document provides the following information:

- Introduction to fuzzing
- The current fuzzing process
  - Workflow
  - Fuzzing requirements
  - Release criteria
  - Setup
  - Writing/Maintaining/Improving fuzz targets
  - Fuzzing CICD Pipeline
  - Fuzzing bugs reporting

The indended readers consists of:

- Fuzzing stakeholders
- Members of the Shared Service Team Security
- Onboarded/To be onboarded Teams
- Anybody who is interested in the fuzzing topic

1.2 Scope

It is best practice to implement fuzzing to test the robustness of a code base.

However, it is not a must for all teams.

Currently, fuzzing is required for all Security Relevant teams (result of Security Relevance Evaluation - SRE[30]).

1.3 Out of Scope

Anything related to the setup of the CIFuzz server is currently responsiblity of the PMT Solution[31].

30 https://devstack.vwgroup.com/confluence/display/public/VWOSS/Security+Relevance+Evaluation
31 https://devstack.vwgroup.com/confluence/spaces/PMT/pages/81521145/PMT+Solution+Train

2 Introduction to Fuzzing

2.1 Motivation

Used by attackers
„Today, if you're a hacker trying to crack a system, the first thing you do is fuzz test it.“ - Barton Miller [2].

Attacker use fuzzing to develop zero-day-exploits. In [3] an attacker developed a zero-click exploit for Tesla MCUs with following steps:

- Reviewed code
- Found bad smells in ConnMan component
- Fuzzed ConnMan
- Found vulnerability
- Exploited vulnerability

In conclusion, if attackers use fuzzing to find vulnerabilites, we should use fuzzing to harden our code.

Hardening

Fuzzing is an established method for negative testing.

Microsoft SSDL [1] recommends to fuzz all code which parses information received across a trust boundary.

Code Intelligence states: “OSS Fuzz and Code Intelligence recommend that fuzzing should be used to test every interface that accepts some form of input. In practice fuzzing should be used to test at least every interface that takes in input from potentially malicious sources, i.e. Internet or user provided files.”

Static Application Security Testing (SAST) is not enough. While very effective, SAST by it self misses potentially critical vulnerabilities that can only be found while the programm is running. This is why we need fuzzing as a form of Dynamic Application Security Testing (DAST) to complement.

Standard

Regulations like UNECE / ISO21434 suggest fuzzing and list it as a method for searching vulnerabilities while performing penetration tests.

ISO21434-Appendix F.2.7 states that “Fuzz testing can be used as a technique for penetration testing."

Requirement

Last but not least, fuzzing is to become requirement and part of the Master Test Plan

2.2 Fuzzing

2.2.1 What is fuzzing?

Wikipedia definition
"… fuzzing or fuzz testing is an automated software testing technique that involves providing invalid, unexpected, or random data as inputs to a computer program.

The program is then monitored for exceptions such as crashes, failing built-in code assertions, or potential memory leaks."

Source: [4]

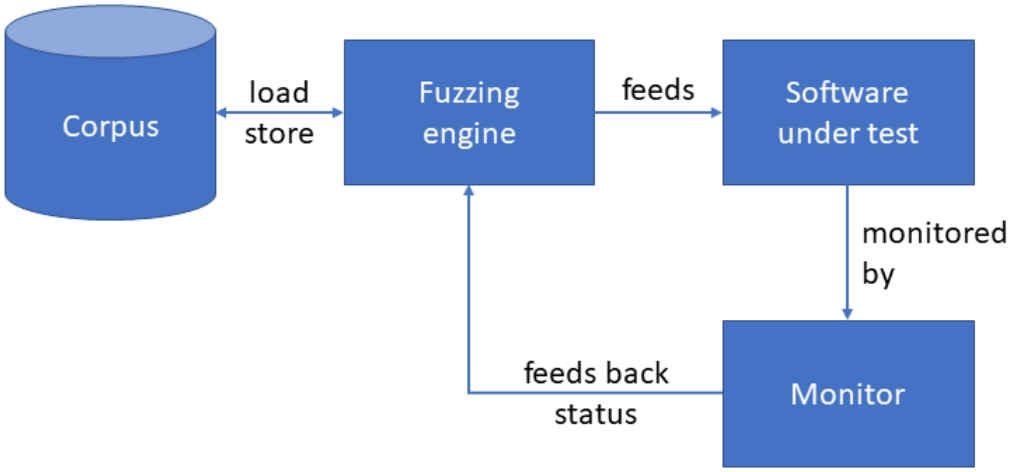
Fuzzing in the secure software development life-cycle (SSDL):

- use available interfaces to inject tainted data
- find vulnerabilities: crashes, memory, timing
- monitor software under test
- collect & report issues
- ensure reproducibility

2.2.2 How does fuzzing work?

Fuzzing tools generate inputs for the system under test (SUT). Modern fuzzing techniques use guided mutations with the goal to discover all paths in a system. The quality of the mutations depend on how much information the tool has over the SUT.

General setup of a feedback-based fuzzer



Corpus

- Stores input vectors

- Grows over time as progress is made
- In coverage-guided / feedback-based fuzzing, each program state reached leads to at least one entry in the corpus
- Requires persistence an maintenance

Fuzzing engine

- Reads inputs from corpus
- Mutates promising data sets based on statistics and prediction algorithms
- Saves input vectors that reach a better result
- Controls SUT, e.g. spawns fresh instance
- Feeds new input into SUT
- Computes feedback from the monitor

Software Under Test

- The unit under test: a function, module, component or system.
- Executes given input
- Optionally instrumented to provide best information to monitor

Monitor

- Observes status of software under test
- Feeds back this status to the Fuzzing engine

2.2.3 Types of fuzzer

Black box	Grey box	White box	Hybrid
Black box fuzzers have the least information on the SUT. They randomly mutate inputs, grammar-based mutators may be used, to generate inputs with valid grammar (e.g., JSON). These fuzzers run "blind" and fast: they can only monitor external application state. Observable status: timing, memory consumption, crashes, unspecified behavior	Grey box fuzzers instrument do not have access to the code base. They instrument the binary to gather status information. They use this trace feedback to explore new paths. The the feedback loop controls their progress while they optimize for coverage. Observable status: timing, memory consumption, crashes, unspecified behavior, jump instructions and more	White box fuzzers possess access to the code base. They use symbolic code execution to solve constraints and find new paths. An instrumented build is used to record the conditions they encounter during an execution. Rather than relying on brute force they resolve paths analytically, which makes them accurate, but slower. Observable status: overall execution trace of the SUT.	Hybrid fuzzers combine a white box and grey box They share the same corpus. The white box fuzzer shall temporarily step in to resolve blockers of the grey box fuzzer. In this way this type is faster than a complete whitebox fuzzer.

The use of sanitazers while fuzzing gives a stronger result as these complement eachother really well. Giving the fuzzer aditional information: out of bound accesses, use-after-free, memory leaks, undefined behavior, data races and more. For details refer to [10].

2.2.4 Stateless and statefull SUT while fuzzing

A SUT may or may not contain states.

Calls into a stateless SUT are independent from each other, thus we can continuously execute inputs.

A statefull fuzz target is a challenge. The SUT will have a different state after every execution and the fuzzer will not be able to mutate correxctly to reach better coverage.

Take away:

- In stateless fuzzing, the state of the software under test is reset between calls.
- In statefull fuzzing, the state of the software under test is maintained between calls.

“Calls” in this context are executions of the SUT, not necessarily executions from the fuzzer. A fuzz-test may distribute its input into multiple calls to the SUT.

2.3 Code-Intelligence and CIFUZZ

Code Intelligence (founded 2017) is an ambitious startup (spin-off of the University Bonn). Their focus is to improve fuzzing technologies and management. Their mission is to shift fuzzing technologies left from pentesters to generic developers by making it more approachable to developers with limited fuzzing experience.

CI Fuzz is the fuzzing middleware developed by Code Intelligence which has been evaluated and finally chosen for CARIAD's fuzzing activities (see. CI-Fuzz Tool Profile[32])

Code-Intelligence's statement regarding their features:

- Fuzzing Core
  - Feedback-Based / Coverage-Guided Fuzz Testing
    Feedback-based fuzz testing (also called coverage-guided fuzzing, instrumented fuzzing, or modern fuzzing) is a technique to increase code coverage while fuzz-testing. A software under test is fed with a series of inputs, which are purposefully mutated in the testing process. The fuzzing engine gets feedback about the code covered during the execution of inputs. Unlike traditional or black-box fuzzing, feedback-based fuzzing explores the program states more effectively and efficiently. This way it discovers more bugs and vulnerabilities hidden deep in the code. To be able to get feedback from the instrumentation, it is required to compile the software code with our instrumentation (offered in our solution).
  - API Fuzzing
    API fuzzing performs fuzz testing with defined API parameters in a software library testing for unexpected behavior. This feature only considers local API interfaces, i.e. local function and method calls. Web and Network APIs are considered in their corresponding features.
  - Basic Network Fuzzing
    Network fuzzing allows providing fuzzing inputs over sockets rather than triggering APIs. The basic network fuzzing supports TCP and UDP sockets.
  - FileHandle Fuzzing
    In some cases, the function under test gets input data from files rather than arguments. Filehandle fuzzing allows creating virtual in-memory files. Depending on the use case, the user is able to retrieve or overwrite the path, filehandle, etc.
  - Web Service Fuzzing
    In addition to the API, filehandle, and network fuzzing, Web Services (structure-aware fuzzing for URL Encoded + REST interfaces) can be fuzzed. Supported frameworks such as Spring and Spring Boot definitions allow an automatic resolution of endpoints and the corresponding request formats. This enables the fuzzer to perform smart structure-aware mutations to effectively fuzz such applications. CZ Fuzz also supports providing an OpenAPI definition for the API of the tested application.
  - Programming Languages Support
    We support C/C++, Java, Kotlin, and Golang. In the programming language column, the number of the supported programming languages can be limited.

32 https://devstack.vwgroup.com/confluence/display/public/VWOSS/CI+Fuzz

- Vulnerability + Bug Detection
  - Bug Reporting
    After each fuzz test, a bug and vulnerability reporting is generated. Here, the developers having access to CI Fuzz are able to view which inputs were able to produce the corresponding errors with an explanation for the reasoning. Here is a non-comprehensive list of the type of bugs and vulnerabilities that CIFuzz detects:
  - Undefined Behavior Detection
    Our platform supports the detection of undefined behavior in C/C++, e.g. by using UBSan. Here, the program is instrumented at compile-time to catch various kinds of undefined behavior during program execution, e.g. using misaligned or null pointers, signed integer overflows, conversions to, from, or between floating-point types, etc.
  - Memory Corruption Detection
    Our platform is able to detect memory corruption bugs and vulnerabilities in C/C++, such as
    - Out-of-bounds accesses to heap, stack, and globals
    - Use-after-free / use-after-return / use-after-scope
    - Double-free, invalid free
    - Memory leaks
      In memory-safe languages such as Java, memory corruption bugs such as OutOfMemoryExceptions, limited stack / heap settings are detected and reported.
  - Crashes / Uncaught Exceptions Detection
    A crash occurs when a software application stops functioning properly and exits. Exception handling is the process of responding to the occurrence of exceptions – anomalous or exceptional conditions requiring special processing - during the execution of a program. In all programming languages, crashes and uncaught exceptions will be detected and reported.
  - Failed Assertion Detection
    An assertion is a predicate (a Boolean-valued function over the state space, usually expressed as a logical proposition using the variables of a program) connected to a point in the program, that always should evaluate to true at that point in code execution. Same as in traditional testing, fuzz testing is able to find and report inputs leading to failed assertions.
  - Race Conditions Detection
    A race condition is a condition where the applications' substantive behavior is dependent on the sequence of uncontrollable events. It can be a bug or vulnerability when the possible behavior is undesirable. Our platform offers detection and reporting mechanisms for various programming languages.
  - Infinite Loops / Timeout Detection
    If an input takes longer than a specified timeout, the process is treated as a failure case. This way, mostly infinite bugs or waiting for never satisfied events will be detected and reported.

  - Out-of-Memory Bugs
    If the tested software utilizes more memory than a configurable threshold, CIFuzz reports the corresponding test case that causes this behavior.
  - Bug Detection during Unit Testing
    CI Fuzz can run existing unit tests with the bug detectors provided by CI Fuzz. This enables users to find bugs completely automatically without having to write any dedicated fuzz tests.
  - OWASP Web / OWASP API
    CI Fuzz supports fuzzing web applications written in JVM-based languages such as Java and Kotlin. In addition to integrating open source bug detectors such as OWASP ZAP, CI Fuzz comes with instrumentation-based bug detectors enabling it to find all categories of bugs in the OWASP Top 10 list. To assist automatic testing of these applications, CI Fuzz can automatically detect the end points offered by the tested application of several frameworks, or use a corresponding OpenAPI definition.
- Process Workflow
  - Continuous Fuzzing in CI/CD
    CI Fuzz provides a seamless integration including report summaries with CI/CD solutions such as GitHub, GitLab, and Jenkins. The integration into any other CI/CD solution is supported by using our CLI interface that can be used for each pipeline script to trigger fuzzing inside the fuzzing server (on-premise and SaaS). Fuzzing configurations are checked in with the source code of the system under test to guarantee the synchronization between the source code and corresponding fuzz tests.
  - Coverage Reporting
    In order to give detailed insights for the fuzzing runs, a source-based coverage report is provided on statement level. This provides a detailed overview of which lines have been covered by fuzzing in each file. This enables users to see which parts of the code have not yet been tested and provide test cases to CI Fuzz so that it can cover them and make mutations out of them.
  - Error Deduplication
    Sometimes, the same error can be triggered with various inputs. Our error deduplication allows listing certain errors only once rather than listing those with each crashing input.
  - Error Filtering
    Found bugs can be filtered according to various criteria such as the bug type and severity level. This allows users to have a better overview of a specific set of interesting bugs.
  - Automatic API Dependency Resolution
    When creating new fuzz tests (also called fuzz targets or test harnesses), developers have to resolve dependencies manually. When using our IDE plugins with the “fuzz this function” feature, dependencies can be resolved automatically in the majority of the use cases (based on the compilation database generated in the initialization process).
  - Debugging
    Errors found in the dynamic analysis also provide the input triggering the bug. Our platform allows loading the inputs for specific fuzz tests into a debugger on the command line or VS Code Debugging.
- Collaboration + Integrations

  - Instant Alerts
    Additionally, to reporting results over CI/CD and result dashboards, findings can be reported over email and slack. Further communication channels such as Google Chat or Microsoft Teams will be added soon.
  - Extended Reporting (incl. PDF)
    Fuzzing results can be easily exported to various formats including PDF, JSON, and Excel.
  - Role-based Access Control
    In the CI Fuzz platform, users can be given different roles that specify what they are allowed to do. This is similar to the role model one has in popular code hosting platforms such as GitHub and GitLab.
  - Supported Hosting Platforms
    CI Fuzz is agnostic to the source hosting platform. For Git-based platforms such as GitHub and GitLab, the user has to provide the location so that CI Fuzz is able to pull the source code of the application to be tested (if dependencies have to be resolved, a script can be provided). For other types of hosting platforms such as SVN, we provide a checkout script integration that can be used to check out new versions of the code.
- Service + Support
  - Standard Support
    Standard support contains an initial response within 24 hours from our fuzzing experts over slack, ticket system, and e-mail for P1 issues.
  - Premium Support (all from standard included)
    Premium support includes everything from standard support and provides additional phone and video conference support.

3 The fuzzing process simplified

3.1 Local build

Requirements

e3sdk_metadata

```
# the target we will use for fuzzing has to have a buildcontainer equal or higher than
master/e3sdk-build-bp1:4.5.0
{
    ...
    "targets": {
        "e3swp": {
            "ignore_target": "false",
            "build_container": master/e3sdk-build-bp1:4.5.0",
            "jenkins_pod_template_build": "e3sdk-build-bp1_4.5.0",
            "conan_build_profile": "--all-variants"
        },
    ...
    }
}
```

cmake/APFuzzing.cmake Check for latest version https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/templates-e3comp/browse/cmake/APfuzzing.cmake?at=refs%2Fheads%2Fdevelop

CMakeLists.txt in root folder

```
# the cmake subdirectory has to be added before, add_cifuzz_build_options is defined in
there.
add_subdirectory(cmake)
...
# The call should have a surrounding block, this way only when specified the fuzzing
targets will be built.
if(VWOS_CMAKE_CIFUZZING)
    add_cifuzz_build_options()
endif()
...
add_subdirectory(src)
add_subdirectory(tests)
```

tests/fuzzing/CMakeLists.txt

```
cmake_minimum_required(VERSION 3.16)

# Set up target dependencies
find_package(... REQUIRED)

target_link_libraries(... ...)

target_include_directories(... ...)

# Add and link target
add_cifuzz_executable(fuzz_example fuzz_example.cpp)
target_link_libraries(fuzz_example LINK_PUBLIC ${...})
```

~/.conan/profiles/fuzzing_x86_64

```
[env]
CC=/usr/bin/clang-15
CXX=/usr/bin/clang++-15
LD=/usr/bin/ld
AR=/usr/bin/ar


[settings]
# Host settings
os_build=Linux
arch_build=x86_64

# Target settings
os=Linux
os.target=local
arch=x86_64
compiler=gcc
compiler.version=9.3
compiler.libcxx=libstdc++11
build_type=Debug
```

Execution

command for building fuzzing binary

```
# TODO: change the profile fuzzing_x86_64 to a new one once all dependencies are built
with clang
```

```
./build.sh --docker --target e3swp --profile fuzzing_x86_64 --verbose --fuzzing --
sanitizer=address:undefined --sanitizer-args=-fsanitize-recover=address:-fsanitize-
address-use-after-scope -j2
```

Result

build/bin/
fuzz_example1 fuzz_example2 fuzz_... ...

3.2 Local run

Requirements:

- binaries of fuzzing targets from last steps
- (optional but very recomended) cifuzz.sh[33] script
- in e3sdk_metadata build_container **master/e3sdk-build-bp1:4.5.0** or higher

Execution:

- ./cifuzz.sh run ./build/bin/fuzz_readConfigFile -e ASAN_OPTIONS=strict_string_checks=1:detect_stack_use_after_return=1:check_initialization_order=1:strict_init_order=1:abort_on_error=0:halt_on_error=0

Result

- .cifuzz-corpus is populated by generated inputs
- .cifuzz-findings is populated by findings if any are found

3.3 Coverage

3.3.1 Local

Requirements:

- in e3sdk_metadata build_container **master/e3sdk-build-bp1:4.5.0** or higher more information on the container here <insert link to container bible>

33 https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3-aux-security-repo/browse/DAST/cifuzz/cifuzz.sh?at=refs%2Fheads%2Fdevelop

- in cmake/APFuzzing.cmake with the latest version <insert link to latest version>
- .cifuzz-corpus populated with inputs

Execution

- # TODO: change the profile fuzzing_x86_64 to a new one once all dependencies are built with clang
- build binary for coverage generation ./build.sh --docker --target fuzzing --coverage --profile fuzzing_x86_64 --buildfolder=build-cov --verbose --fuzzing -j2
- generate coverage file ./cifuzz.sh coverage --format <lcov or html> --add-corpus <.cifuzz-corpus/path/to/corpus/of/target> <build-cov/path/to/coverage/binary>

Result

- lcov file with coverage information or html website with coverage report

3.4 Debug

3.4.1 Local

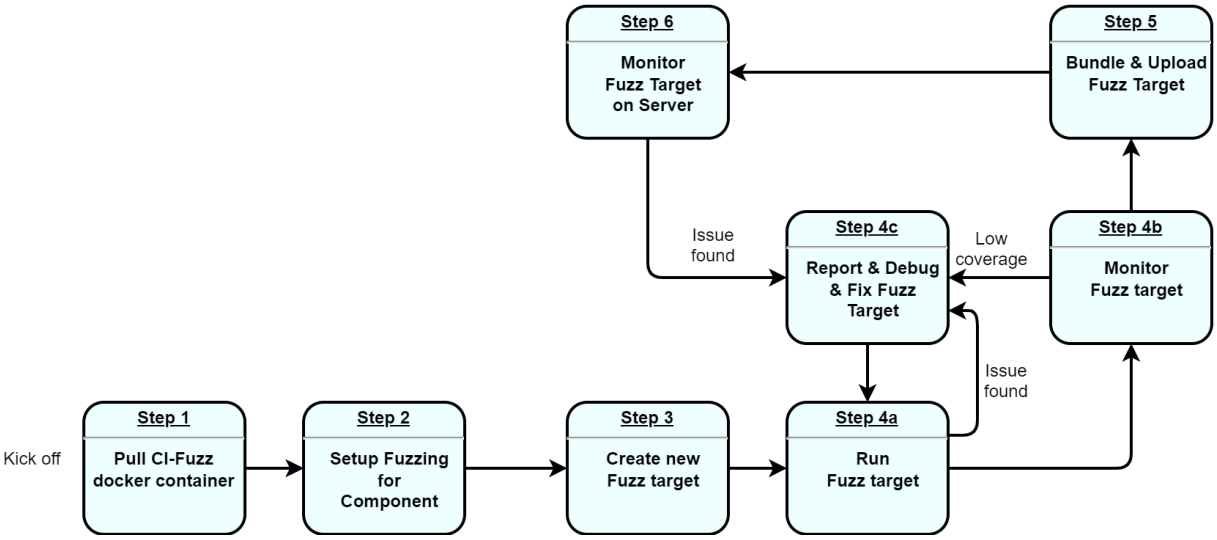
Requirements:

- cifuzz.sh to open gdb server in docker file
- gdb config on local machine to connect to gdb server

4 The fuzzing process in detail

4.1 General workflow (in progress)

This chapter describes the current fuzzing process. Note that at the time of writing this report, the CIFUZZ CICD server is not up and running yet, hence, this process focuses on creating the fuzz target on a local machine instead of letting the CICD pipeline (e.g., Jenkins, Github server) handle it.



Step	Phase	Description	Output
1	Pull CI-Fuzz docker container	Pull a prepared docker container with cifuzz required toolings and compiler from artefactory. This container will be used to build the fuzz target.	CI Fuzz enabled Docker container available
2	Setup fuzzing for component	Adapt component's CMakeLists.txt file to enable fuzzing.	Component modified for fuzzing
3	Create new fuzz target	Select functionality to fuzz test and implement the fuzz test. This functionality can be a function, a class or even a sub-component. To know more about what to fuzz, see our following chapter (see page 40).	Fuzz target can be built

Step	Phase	Description	Output
4a	Run fuzz target	Fuzz target is running until ... • no new edges are found (→ 4b) • time limit is reached (→ 4b) • one or multiple issue are found (→ 4c)	Corpus data, if some edges were found. Fuzzing input for when an issue occurrs
4b	Monitor fuzz target	Replay corpus data to create coverage information Coverage information can be used to identify blockers and fine-tune the fuzz target setup.	Coverage report available
4c	Report & debug & fix fuzz target	In case of an issue, replay fuzz target with the corresponding input and analyze finding => Fix the code to mitigate the issue In case of low coverage, we can analyze the coverage information to remove blockers => Fix the fuzz target to remove blocker	Patched vulnerability (prod. code) or improved fuzz target (test code)
5	Bundle & upload & fuzz target	Create self contained bundle of executables and dependencies to run on server. Upload the bundle to server project and schedule server runs	Fuzz target running on server
6	Monitor fuzz target on server	Check how fuzz target performs over time either via CIFUZZ server dashboard or the cictl tool	Progress of fuzzing or issue found

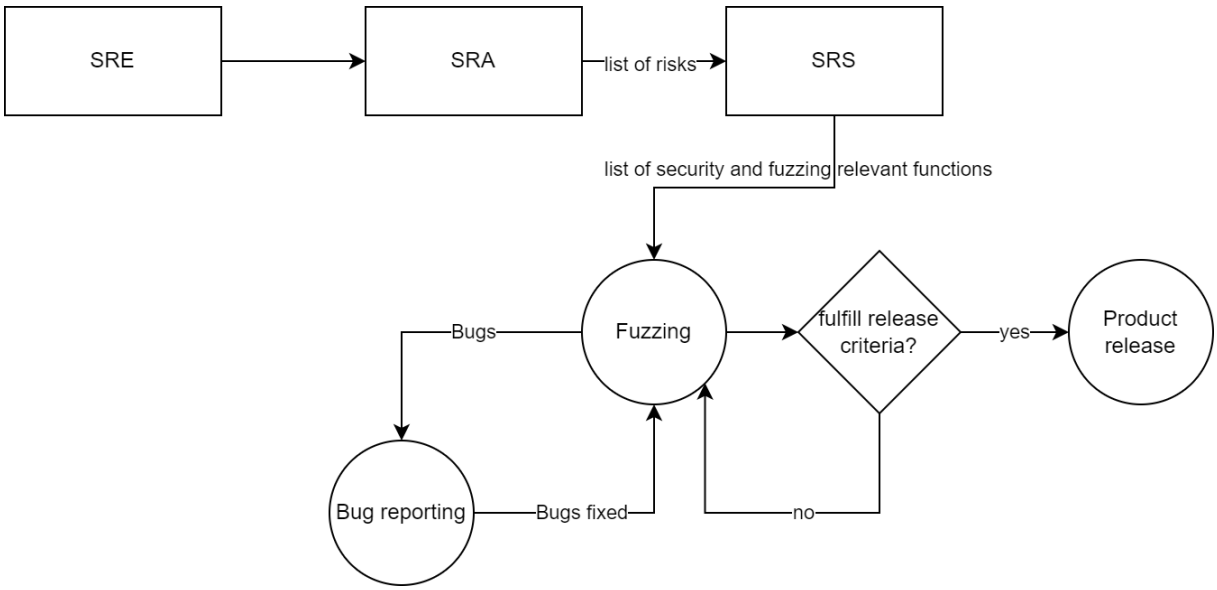
4.2 What to fuzz?

In general, if possible, it is best practice to fuzz all available components and functions test a component extensively. However, due to limited resources, some components must be prioritized over others. In that case, it is important to have a metric to decide which component should be prioritized other the other.

4.2.1 Risk driven fuzzing process

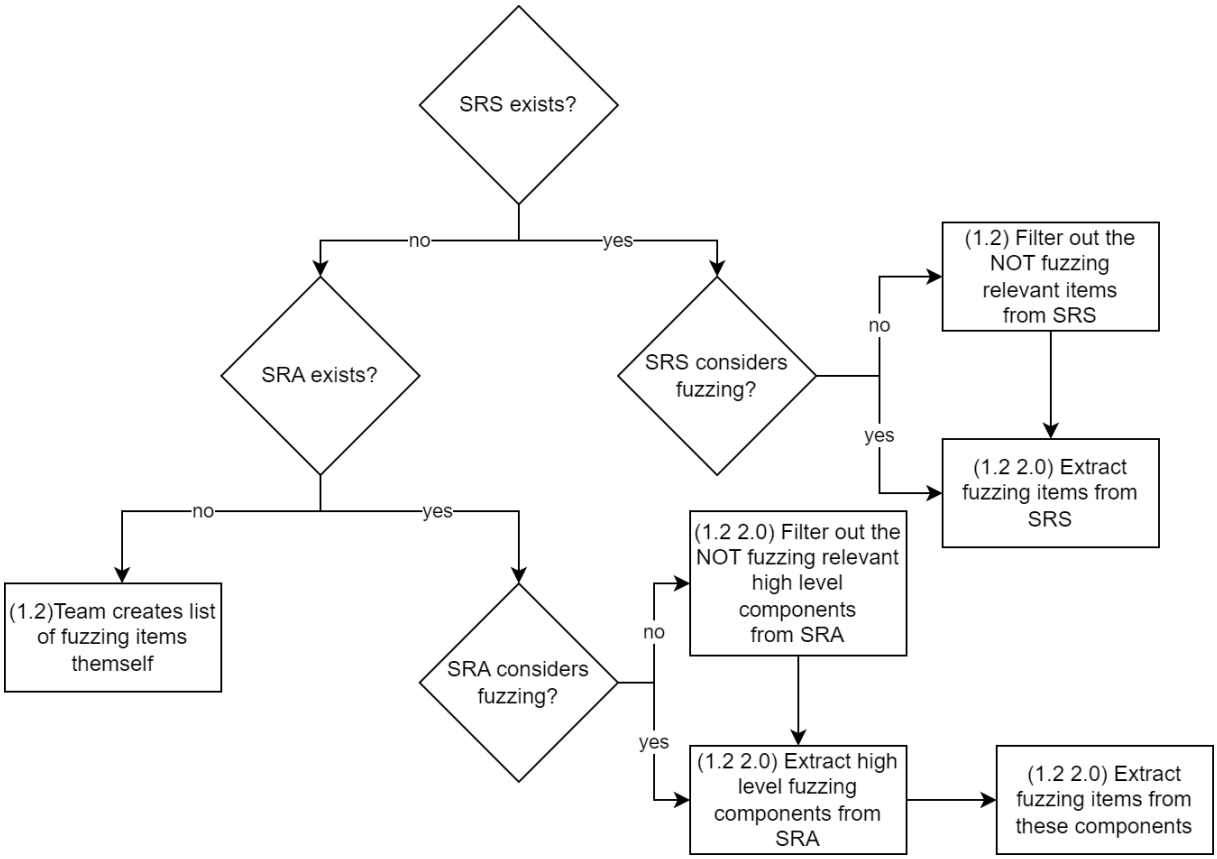
That is where the idea of "risk driven fuzzing process" comes into play. The risk driven fuzzing process prioritizes components with higher security risks over the lesser one. Consequently, fuzzing would focus more on the components with higher security risks. This information can be extracted from the

Security Relevance Evaluation[34] (whether or not a component is security relevant) and Security Risk Analysis[35] (what kind of security risks the component might have).



The figure above shows the process of extracting a list of security and fuzzing relevant functions and the general fuzzing workflow until product release. The process is documented in details and updated regularly on 1.1 +1.2 - VW.os - Workflow of fuzzing & release criteria/definition of done[36].

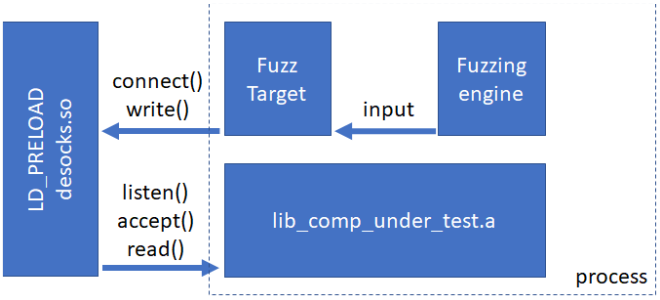
34 https://devstack.vwgroup.com/confluence/display/VWOSS/Security+Relevance+Evaluation
35 https://devstack.vwgroup.com/confluence/display/public/VWOSS/Security+Risk+Analysis
36 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239380/1.1+1.2+-+VW.os+-+Workflow+of+fuzzing+release+criteria+definition+of+done



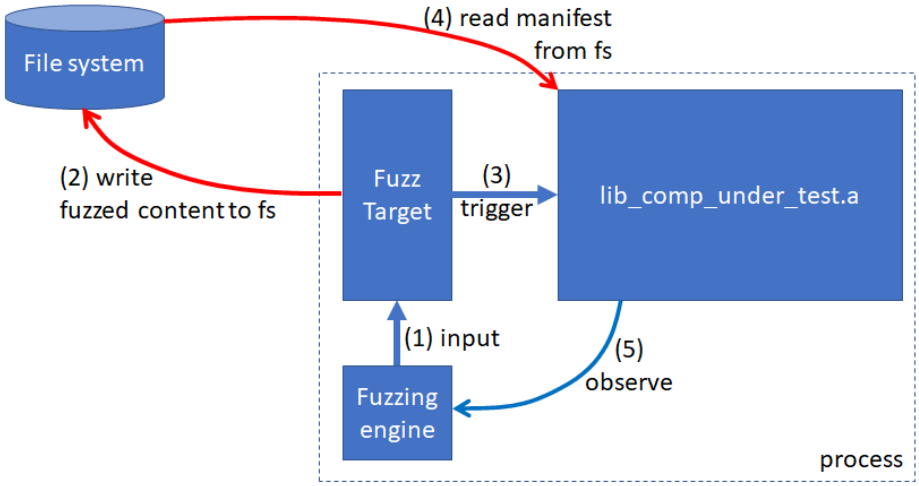
This figure above shows the detailed process of how to create a list of specific fuzzing items from the Security Requirement Specification (SRS), or Security Risk Analysis[37] or how the team will create a list of fuzzing items themselves in case SRA and SRS haven't been done yet. The process is documented in details and updated regularly on 1.2 - Guideline to extract a list of security relevant fuzzing items[38].

37 https://devstack.vwgroup.com/confluence/display/public/VWOSS/Security+Risk+Analysis
38 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

4.2.2 Types of fuzz targets

Type	Description
API Fuzzing	A software under test may be fuzzed using its public API. Predestined for API fuzzing are libraries, in particular if they perform some kind of parsing. The API function under test can receive a variety of inputs such as: • Raw input such as a byte array which can be forwarded directly from the fuzz engine to the function under test `const uint8_t* pData, size_t size` • Struct input: If an API requires more complex or multiple objects, a structure mutator can be defined. It ensures that the generated inputs are deserialized into a valid structure which is then used to construct the objects expected by the function under test. One way to achieve this is by using Google's protobuf mutator, which is made available by CI Fuzz. Plain structures are automatically translated into corresponding protobuf definitions that can be dealt with by the libprotobuf mutator and the creating structures out of the mutated protobuf messages. This means that the fuzzer will only produce valid structs. • Split input: As mentioned above, the API to be tested may require more than one input. E.g. options can be toggled or preconditions altered. In this case the received input needs to be split into the required number of sub-inputs, which are independently processed by the fuzz target. If input vectors too small in size are rejected. Various strategies exist on how to split input. Details can be found here: https://github.com/google/fuzzing/blob/master/docs/split-inputs.md [11] For more complex data structures such as C++ classes, CI Fuzz provides helper functions to fill the class instances. This is done by offering an interface that allows to consume data of certain types from the raw byte stream generated by the fuzzer. This includes for example ConsumeRandomLengthString(), ConsumeFloatingPoint(), PickValueInArray()..etc.
Socket Fuzzing	One major gateway and entrance point for attacks on application security for many software products is the network communication path, the socket associated with the connection acting as an end point and of course the following parsing parts of the components, which might contain a security-relevant flaw. From testing point of view it is advantageous to perform the tests with minimal changes to the software under test and that the component can be tested as a whole with little effort. Furthermore, without sacrificing maintainability due to introduction of artificial entry points only for testing purposes.  (Generel setup for socket fuzzing)

Type	Description

File System Fuzzing	If a function under test reads from a file, three approaches are possible to inject the fuzzing data into that file:
	• Injection of taint via mocked I/O
	  • Pros:
	  • Simple to use
	  • Easy to set up the fuzz target
	  • Cons:
	  • Library under test must be mockable
	• Intercepting file I/O calls via preloaded library
	  • Pros:
	  • No change to library under test
	  • Flexible
	  • Cons:
	  • Complexity added by the preloaded library
	• Creation of temporary files in shared memory
	  • Pros:
	  • No change to library under test
	  • Easy to set up the fuzz target
	  • Cons:
	  • Only applicable if the caller controls the file
	 

4.2.3 What is a good target?

There are different properties and criterias which might influence the effort on creating fuzz target.

Target's property	Preferably	Adding extra complexity
State fullness	stateless	stateful
Configuration options	few	many
Preconditions	few	many
Runtime	fast	slow
Memory footprint	low	high
Concurrency	Single threaded	Multithreaded
Deterministism	deterministic	non-deterministic
Focus	Narrow	wide
Build time	quick	slow
Structure of input	No or low complexity	High complexity
Dependencies	few	many

For more information, see our references:

https://github.com/google/fuzzing/blob/master/docs/good-fuzz-target.md [13]
https://github.com/google/fuzzing/blob/master/docs/why-fuzz.md [14]

For a full list of rules / recommendation what to fuzz and what not to fuzz, please refer to our up to date process page: 1.2 - Guideline to extract a list of security relevant fuzzing items[39]

4.2.4 How to document what to fuzz with Doxygen?

4.2.4.1 Motivation why we use Doxygen to label fuzzing relevant functions

In order to create an overview of the progress of writing fuzz targets and help the developer keep track of which functions / interfaces / classes to write fuzz targets for, we use doxygen to track this. Doxygen is already used currently to generate documentation. As the result, a developer can generate:

HTML doxygen documentation for fuzzing. Example list of all functions to be fuzzed:

39 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

OTAMasterTokenManagement 999.0.0

Main Page | Related Pages | Namespaces | Classes | Files

OTAMasterTokenManagement
- Detailed Design of OTA Token Managem
- Implementation Details for Concepts
- Token Management Offboard Communic
- Detailed Design of Protection of Vehicle I
- Detailed Design of Token Management S
- Token Management SFL with Wakeup
- Dynamic behavior
- Static structure
- Design decisions
- README
- README
- Functions required to be fuzzed
- Namespaces
- Classes
- Files

Functions required to be fuzzed

Member vwg::otamaster::token_management::sfl::ImmobilizerSpecialFunctionLoginSubSystem::changeJob (stdtypes::DT_uuid const &jobId, services::connect::otamastercapability::otatokenmanagement::immobilizerSpecialFunctionLoginChangeJob_requestedLoginStatus const &requestedLoginStatus) override

because of API endpoints. TODO: Add more info

Member vwg::otamaster::token_management::sfl::ImmobilizerSpecialFunctionLoginSubSystem::startJob (services::connect::otamastercapability::otatokenmanagement::immobilizerSpecialFunctionLoginStartJob_specialFunctionLoginParameters const &specialFunctionLoginParameters) override

because of API endpoints. TODO: Add more info

Member vwg::otamaster::token_management::sfl::ImmobilizerSpecialFunctionLoginSubSystem::stopJob (stdtypes::DT_uuid const &jobId) override

because of API endpoints. TODO: Add more info

Class vwg::otamaster::token_management::TokenManagementSkeletonImpl

because of API endpoints. TODO: Add more info

Member vwg::otamaster::token_management::TokenManagementSkeletonImpl::immobilizerSpecialFunctionLoginPollJob (stdtypes::DT_uuid const &jobId) override

because of API endpoints. TODO: Add more info

Member vwg::otamaster::token_management::TokenManagementSkeletonImpl::immobilizerSpecialFunctionLoginStartJob (otaTokenManagement::immobilizerSpecialFunctionLoginStartJob_specialFunctionLoginParameters const &specialFunctionLoginParameters) override

because of API endpoints. TODO: Add more info

Member vwg::otamaster::token_management::TokenManagementSkeletonImpl::immobilizerSpecialFunctionLoginStopJob (stdtypes::DT_uuid const &jobId) override

because of API endpoints. TODO: Add more info

◆ changeJob()

ara::core::Future< **IImmobilizerSpecialFunctionLoginSubSystem::ChangeJobOutput** >
vwg::otamaster::token_management::sfl::ImmobilizerSpecialFunctionLoginSubSystem::changeJob (
)

Requests to change the job, e.g. from 'prepared' to 'unlock'.

Parameters

[in] **jobId** unique ID of corresponding job

[in] **requestedLoginStatus** state of SFL which should be achieved

Returns

interface-defined structure with fields:

- requestStatus - Request status:
  - returns requestStatusOk in case of successful change of the request job
  - returns requestStatusError* in case of problem

Requirements

req-otam-all-persistence-save-on-change

Fuzzing required:

because of API endpoints. TODO: Add more info

Or an XML file to be parsed further such as this one: e3_comp_otam_tm_e3fuzz.xml[40]

This XML is automatically generated during build by the CICD Pipeline and can be automatically parsed. As a result, a Confluence Tracking page can be automatically generate (such as the example SNEAK PEEK tracking page for e3_comp_otam_tm[41])

4.2.4.2 How to set up your project so Doxygen for fuzzing work?

First of all, configure your doxygen dox file of your repository. This has to be done for each and every single repository ONCE.

40 https://devstack.vwgroup.com/confluence/download/attachments/1483434491/e3_comp_otam_tm_e3fuzz.xml?api=v2&modificationDate=1707140242350&version=1

41 https://devstack.vwgroup.com/confluence/display/E3SPVMAN/%5B%21SNEAKPEEK%21%5D+Fuzzing+Function+Overview

Edit repo/doc/doxygen/SWP-DOC-CONFIG.dox with following options

```
...
# Enable generation of XML file
GENERATE_XML = YES
...
# In some cases, you might have to disable this flag to avoid error
WARN_AS_ERROR = NO
...
# Adding alias at the end of file which used for generating fuzzing documentation
ALIASES += e3fuzz="@xrefitem e3fuzz \"Fuzzing required\" \"Functions required to be
fuzzed\""
```

After adjusting the doxygen configuration, mark the class / function to fuzz like follows:

Code example of how to label a class and a function as fuzzing relevant with doxygen ALIAS

Marking the fuzzing required classes and functions

```
/// @brief Provides an implementation of skeleton for handling requests for TM SFL and
SFD
///
/// @coversreqs
/// @req{req-otamaster-tm-interface}
/// @req{req-SW_COMP_REQ-otam-token-sfd-request-sfd-tokens}
/// @endcoversreqs
/// @e3fuzz The reason why this class must be fuzzed
class TokenManagementSkeletonImpl : public
 otaTokenManagement::skeleton::OtaTokenManagementSkeleton{
...

    // @brief Requests the result of SFL job
    /// @param[in] jobId unique identifier of corresponding SFL job
    ///
    /// @coversreqs
    /// @req{req-SW_COMP_REQ-otam-token-sfl-polljob}
    /// @endcoversreqs
    /// @e3fuzz  the reason why this function needs to be fuzzed?
    ::ara::core::Future<sfl::IImmobilizerSpecialFunctionLoginSubSystem::PollJobOutput>
    immobilizerSpecialFunctionLoginPollJob(stdtypes::DT_uuid const& jobId) override;
...
}
```

Finally, after labeling all classes and functions as fuzzing relevant, the CICD pipeline will generate the Doxygen documentation.

If you want to generate this locally by yourself, please follow the instruction provided by PMT Solution to set up IDE HUB https://devstack.vwgroup.com/confluence/display/TOOLCHAIN/

%5BPREVIEW%5D+IDE.HUB+Installation. After that, you can generate the doxygen documentation locally by executing:

Command to generate doxygen documentation locally

```
e3sdk.py doxygen generate --target e3swp

# The html documentation will be in
firefox build/doc/doxygen/html/e3fuzz.html

# The xml documentation will be in
vim build/doc/doxygen/xml/e3fuzz.xml
```

Here are some real examples from OTA Master repositories:

- OTA Common: https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_comp_otam_common/commits/4930f3807cefff5911c68635d5aa4f90b68d6897#doc%2Fdoxygen%2FSWP-DOC-CONFIG.dox
- OTA TM: https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_otam_tm/commits/f0126a4e820791b21b8f748d0664f8c07d4f387d#doc%2Fdoxygen%2FSWP-DOC-CONFIG.dox
- OTA VSM: https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_otam_vsm/commits/10807573a94f4d4c7ad76d4a0c9d6c6321576e00#doc%2Fdoxygen%2FSWP-DOC-CONFIG.dox

4.3 Setup – Installation

In this chapter, we will explain the process to setup cifuzz for both user and maintainer of the cifuzz tooling. We will go through the process of explaining:

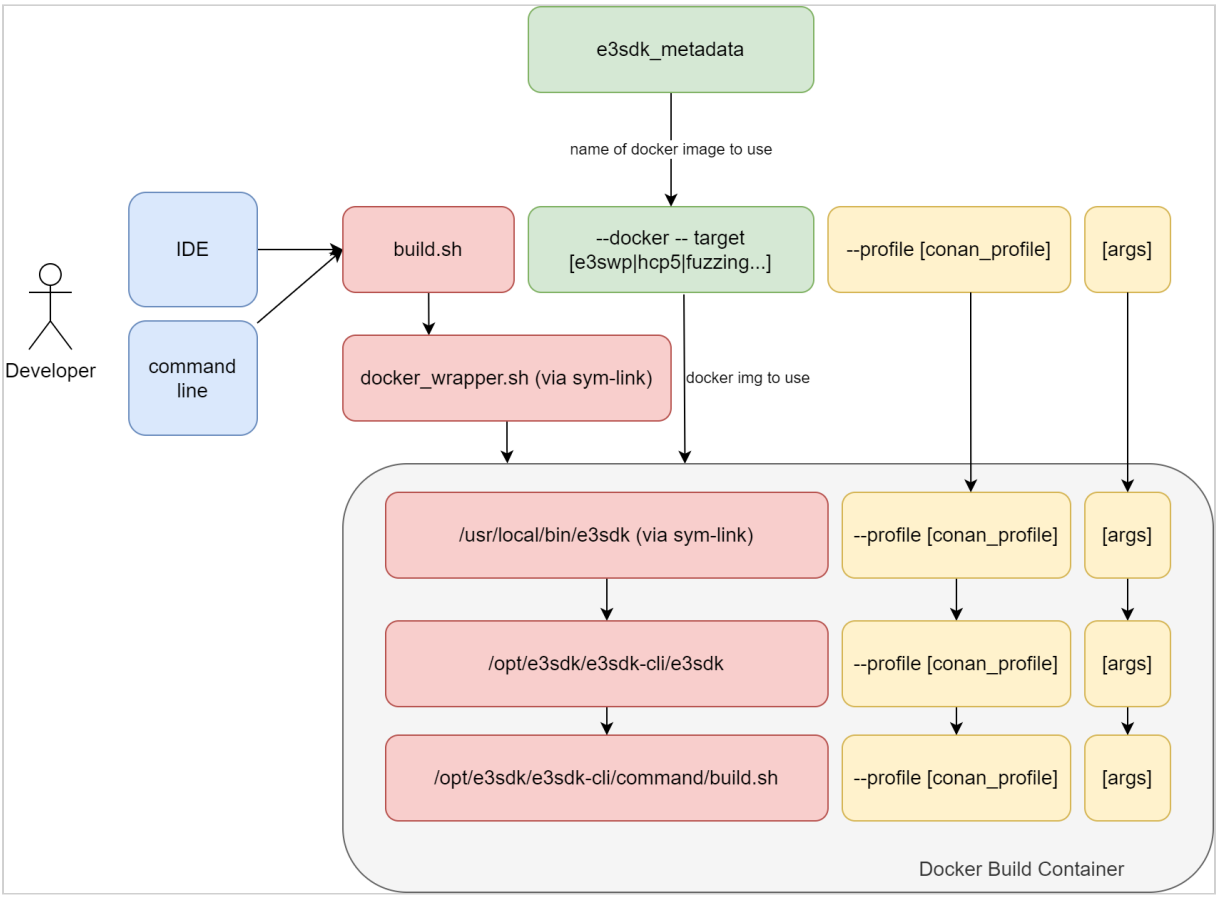
- How the current PMT solution's build process works?
- How we integrate our cifuzz tooling into that process (with docker container)?
- How to maintain / update / adjust the build process e.g., when a newer version of cifuzz is to be updated?
- How to install cifuzz on your local machine to test with it?
- How to install a CI Sense server locally to test with it?

4.3.1 PMT build process (for cifuzz maintainer)

(Reference: Build Docker usage overview[42] from PMT SW Factory ART)

All of VW.os C/C++ project are built using the dockerized approach developed and maintained by PMT SW Factory ART as described on . The designated docker container is the central element which gathers all the tools required for the build process such as:

- build and create conan package: the cross-compiler, the building tools (cmake, make, conan, ...), the configuration (conan profiles)
- create the doxygen documentation (for the SWAD)
- Parasoft (SCA and Unit-test) local usage: provide the configuration
- Parasoft (SCA and Unit-test) pipeline usage: provide the configuration and the tools
- providing the environment to run the unit-test executables
- clang-tidy and clang-format: provide the tools.



The figure above shows the current PMT build process for C/C++ code, which works as follows:

1. First, the developer calls the build.sh script in there repository with something like:

42 https://devstack.vwgroup.com/confluence/spaces/TOOLCHAIN/pages/203718988/Build+Docker+usage+overview

```
# TODO: change the profile fuzzing_x86_64 to a new one once all dependencies are
built with clang
./build.sh --docker --target e3swp --profile fuzzing_x86_64 --verbose --fuzzing
[args]
```

2. The combination of ./build.sh --docker --target e3swp will fetch the name of the docker image in e3sdk_metadata inside the project folder. Subsequently, the build.sh, which is technically a symbolic link to the docker_wrapper.sh script, either fetch the docker image from the docker registry (e3-e3sp-dockerdev.docker.devstack.vwgroup.com[43]). In case the docker container is available locally, the docker_wrapper.sh script will start the docker container right away.
3. After starting the docker container, the script /usr/local/bin/e3sdk , which is technically a symbolic link to /opt/e3sdk/e3sdk-cli/e3sdk inside the docker container will be called with the rest of the arguments marked as yellow above.
4. The /opt/e3sdk/e3sdk-cli/e3sdk then call the corresponding /opt/e3sdk-cli/command/build.sh with the rest of the arguments marked as yellow above.
   ⚠️Note that the script /opt/e3sdk/e3sdk-cli/e3sdk will replace the {command} in /opt/e3sdk-cli/command/{command}.sh with whatever script the developer calls in step 1 (more info about other scripts provided and managed by PMT team can be found Build Docker usage overview[44]) For example, if the developer were to call:

   ```
   # TODO: change the profile fuzzing_x86_64 to a new one once all dependencies are
   built with clang
   ./create_conan_package.sh --docker --target e3swp --profile fuzzing_x86_64 --
   verbose [args]
   ```

   instead, the /opt/e3sdk/e3sdk-cli/e3sdk script will call /opt/e3sdk-cli/command/create_conan_package.sh inside the docker container with the rest of the arguments marked as yellow above.
5. Finally the build process will start.

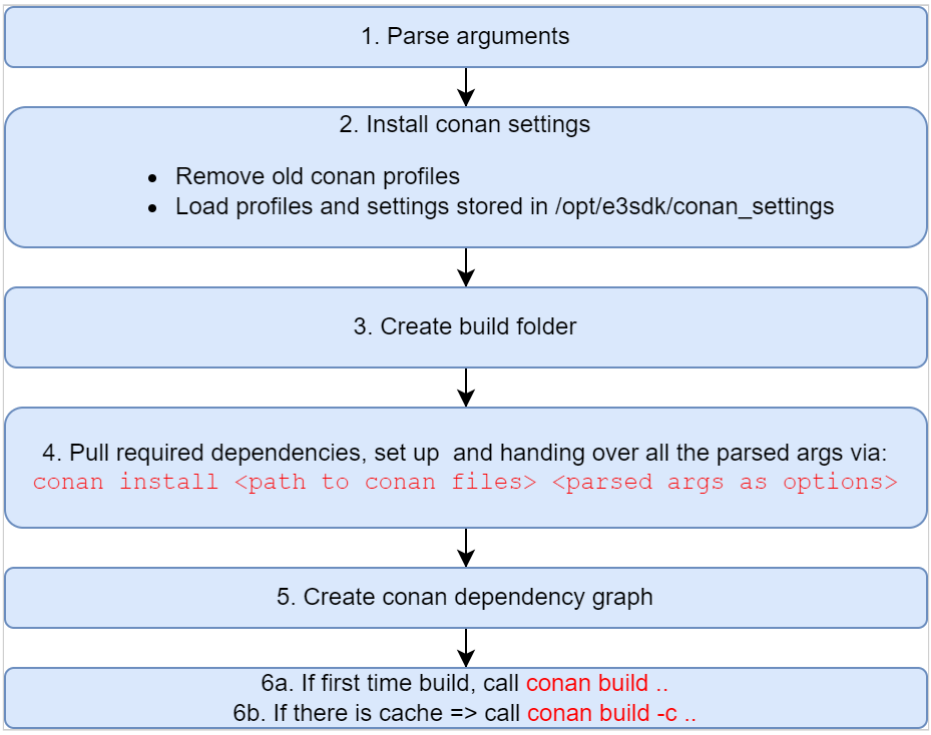
4.3.2 How to add fuzzing option to the current PMT build process? (for cifuzz maintainer)

In order to have fuzzing aka. our cifuzz toolchain fully integrated in the current PMT dockerized build process as explained above (see PMT_build_process (see page 49)), we need to understand the what the build.sh script does and how all the docker containers are built.

43 http://e3-e3sp-dockerdev.docker.devstack.vwgroup.com
44 https://devstack.vwgroup.com/confluence/spaces/TOOLCHAIN/pages/203718988/Build+Docker+usage+overview

4.3.2.1 What does build.sh do?

The build.sh[45] script (see script in docker_e3sdk_commonpart_build_bp1[46] repository) parses all the arguments passed by the developer to the build.sh script outside of the docker container. Subsequently, build.sh script either resolves them directly such as (--buildfolder) flag or transformed these and passes it on to conan for further transformation. A simplified workflow of the build.sh script can be illustrated like follows:



Step	Explaination	Product
1	Arguments passed to build.sh are parsed and either resolved directly or passed to conan install as options later on	list of options for conan
2	After parsing the arguments, the old conan profiles in ~/.conan/profiles/ are removed and replaced with the one from the docker container in /opt/e3sdk/conan_setting	list of corresponding conan profile depending on the docker container in use

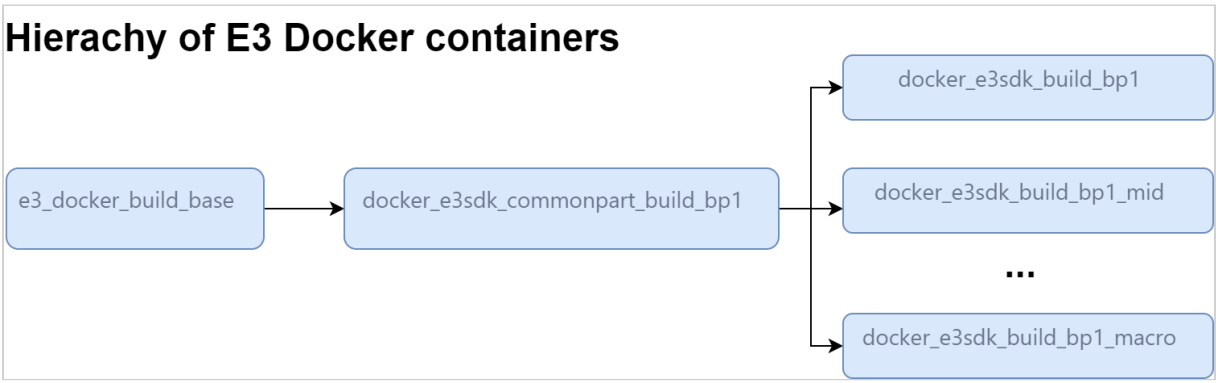
45 https://devstack.vwgroup.com/bitbucket/projects/E3INT/repos/docker_e3sdk_commonpart_build_bp1/browse/files/e3sdk-cli/commands/build.sh

46 https://devstack.vwgroup.com/bitbucket/projects/E3INT/repos/docker_e3sdk_commonpart_build_bp1/browse/files/e3sdk-cli/commands/build.sh

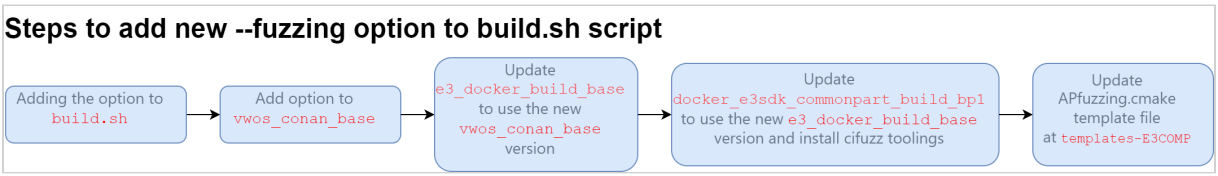
Step	Explaination	Product
3	The build folder is created in the project directory (which was mounted inside the docker container). The --buildfolder flag is responsible for this	designated build directory
4	Conan install executes requirements function in conenfile.py. This also passes the options to conan as preparation for the build step.	All required libs and packages are fetched from the library based on the conan profile specified. The fetched packages are stored in ~/.conan/data. Also all Find<libname>.cmake files are generated and placed in the build folder
5	The conan dependency graph is generated	conan dependency graph in png
6	conan build is called, which in turns configures all CMake parameters and call cmake build.	Makefiles and the build target

4.3.2.2 What is the hierachy for all current build docker containers?

Currently, the build docker containers have following hierachy.



4.3.2.3 How do I add a new flag for fuzzing in the build.sh script?



4.3.2.3.1 Adding the option to build.sh

The build.sh[47] script (currently maintained and managed in docker_e3sdk_commonpart_build_bp1[48] repository). To add a new flag such as --fuzzing, we add / modify the following to the script:

Adding --fuzzing flag to script

```
# Parse --fuzzing
if has_arg '--fuzzing'; then
  fuzzing="True"
  opt_fuzzing="-o fuzzing=${fuzzing}"
  ARGS=$(echo "$ARGS" | sed -r 's!--fuzzing!!g')
else
  fuzzing=""
fi

...
# Pull required dependencies
  {
    conan_wrapper install .. ${opt_sanitizer} ${opt_sanitizer_args} ${opt_coverage} ${opt_fuzzing} -pr ${conan_profile} ${build_profile} ${confirm_param}
  } || {
  echo "conan install not successful. You used the following conan profile: $conan_profile"
  echo ""
  echo "viable profiles are:"
  echo "  for SWPAC:      jante-aarch64 / jante-aarch64-debug / ubuntu-x86_64 / ubuntu-x86_64-debug"
  echo "  for HCP1:     hcp1-aarch64 / hcp1-aarch64-debug / ubuntu-x86_64 / ubuntu-x86_64-debug"
  echo "  for HCP5:     hcp5-aarch64 / hcp5-aarch64-debug / ubuntu-x86_64 / ubuntu-x86_64-debug"
  echo "  for ICAS1.2:  icas12-aarch64 / icas12-aarch64-debug / ubuntu-x86_64 / ubuntu-x86_64-debug"
}
```

Note that when conan_wrapper install .. <options> is called, the conanfile.py in project root is called. Then, following things are triggered:

1. the init method in vwos_conan_base.py[49] is called since the project class specified in project's conanfile.py for example this conanfile.py[50] inherits from the VwOsConanBase[51]class.

47 https://devstack.vwgroup.com/bitbucket/projects/E3INT/repos/docker_e3sdk_commonpart_build_bp1/browse/files/e3sdk-cli/commands/build.sh
48 https://devstack.vwgroup.com/bitbucket/projects/E3INT/repos/docker_e3sdk_commonpart_build_bp1/browse/files/e3sdk-cli/commands/build.sh
49 https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/vwos_conan_base/browse/vwos_conan_base/vwos_conan_base.py
50 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_tlrlib/browse/conanfile.py
51 https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/vwos_conan_base/browse

2. The init method sets all the options being passed in the variable options such as fuzzing="True"
3. Then, the conan_wrapper install .. <options> will triggers the method requirements(self) and build_requirements(self) in conanfile.py to download all required dependencies from the artefactory and/or build the packages specified in build_requirements(self) based on the specified profile in ${conan_profile} ${build_profile}.

4.3.2.3.2 Adding the option to vwos_conan_base package

After adding the --fuzzing option in build.sh[52] script, we must proceed to adding the appropriate options to the vwos_conan_base.py[53] package. In order to do so, we need to:

1. Create a branch on vwos_conan_base repo[54]
2. Make changes like so[55]. Also update the version of the package
3. Test it locally building and installing a new vwos_conan_base package in a respective docker container and try to execute build.sh with the --fuzzing flag to see if CMakeLists.txt file inherits the option `VWOS_CMAKE_CIFUZZING`

   ⚠️ Note that you would have to test this inside a local docker container. To do this, simply:

   a. Pull the latest docker commonpart image via

   Pull the latest docker commonpart image

   ```
   docker pull e3-e3sp-dockerdev.docker.devstack.vwgroup.com/master/e3sdk-build-bp1-commonpart:1.6.11
   # Then change the name of the image to cifuzz:latest
   docker tag e3-e3sp-dockerdev.docker.devstack.vwgroup.com/master/e3sdk-build-bp1-commonpart:1.6.11 cifuzz:latest
   ```

   b. Open a shell session to the docker commonpart container by using the SDK `e3sdk_scripts/shell.sh` script

   Opening a shell session to the docker container with cifuzz.sh script

   ```
   noname:~/workspace/e3_comp_tlrlib$ e3sdk_scripts/shell.sh --docker --target fuzzing
   ```

52 https://devstack.vwgroup.com/bitbucket/projects/E3INT/repos/docker_e3sdk_commonpart_build_bp1/browse/files/e3sdk-cli/commands/build.sh
53 https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/vwos_conan_base/browse/vwos_conan_base/vwos_conan_base.py
54 https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/vwos_conan_base/browse
55 https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/vwos_conan_base/commits/1b1aa577e2a0a04fa9eb4f5760bcf5e8b560d78e#setup.cfg

The `e3sdk_scripts/shell.sh` script is configure to create a docker container from the image specified in `e3sdk_metadata` for the corresponding target .

c. Then install the new vwos_conan_base package inside the docker container:

Install the new vwos_conan_base package

```
$ python3 -m pip install path/to/the/file/vwos_conan_base-X.X.X-py3-none-any.whl
# Then verify that the package has been installed via:
$ python3 -m pip list
```

d. Test if our implementation of vwos_conan_base was successful:
In the root CMakeLists.txt file, write something for debugging like:

Entry for debug in CMakeLists.txt

```
if(VWOS_CMAKE_CIFUZZING)
    message(STATUS "Building with CIFUZZ flag")
endif()
```

And finally, execute build.sh and check if our message shows up in the log:

Executing build

```
# TODO: change the profile fuzzing_x86_64 to a new one once all dependencies are built with clang
$ e3sdk build --fuzzing --profile fuzzing_x86_64 --verbose
```

4. After testing, upload the package to https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-sdk-pypi/vwos-conan-base with the correct version and directory hierachy.

4.3.2.3.3 Update e3_docker_build_base to use the new vwos_conan_base package version

1. Update the version of vwos_conan_base used in e3_docker_build_base and change log.
2. Create a pull request to master branch
3. After successfully merge to the master branch, tag that merge with the new version e.g., in this example, 1.5.7 by clicking

4. Finally, Jenkins CICD Pipeline will build a new e3_docker_build_base image which can be pulled from artefactory

4.3.2.3.4 Update/adjust docker_e3sdk_commonpart_build_bp1[56] to use the new e3_docker_build_base image version and install the cifuzz tools

1. Create a new branch. Currently, we have implemented our changes on branch VWOS-7577-fuzzing of docker_e3sdk_commonpart_build_bp1[57]. In the following, the steps and changes to the Dockerfile for the sake of fuzzing is explained.
2. In docker_e3sdk_commonpart_build_bp1[58], adjust the version of e3_docker_build_base in Dockerfile by adjusting this line

Adjusting base image version

```
ARG BASE_VERSION="1.5.7"
```

3. Install the required tools for cifuzz such as:
   a. CMake >= 3.16 (Installed)
   b. Clang >= 15 (Installed)
   c. LLVM >= 15 (Needed to be added to the docker_e3sdk_commonpart_build_bp1/files/apt_packagelist)
4. Next, have do download the cifuzz installer and cictl tools. Currently, the current cifuzz installer and cictl tools are hosted on the JFROG repo e3-e3sp-gen-teamcicd[59]
   a. In order to update the cifuzz installer to the next version, you must download the installer of the new version on Code-Intelligence Download Portal[60] and deploy it on to the JFROG repo e3-e3sp-gen-teamcicd[61] under ci_fuzz folder. You must name the installer file cifuzz_installer_linux_amd64_X.X.X with X.X.X being the current version.
   ⚠️ Note that it takes a while to get the authentication code with CARIAD email address (approx. 10 minutes). Alternatively, you can use following command with an access token to download the installer via shell:

56 https://devstack.vwgroup.com/bitbucket/projects/E3INT/repos/docker_e3sdk_commonpart_build_bp1/browse/files/e3sdk-cli/commands/build.sh
57 https://devstack.vwgroup.com/bitbucket/projects/E3INT/repos/docker_e3sdk_commonpart_build_bp1/browse/Dockerfile?at=refs%2Fheads%2Ffeature%2FVWOS-7577-fuzzing
58 https://devstack.vwgroup.com/bitbucket/projects/E3INT/repos/docker_e3sdk_commonpart_build_bp1/browse
59 https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-gen-teamcicd/ci_fuzz
60 https://downloads.code-intelligence.com/
61 https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-gen-teamcicd/ci_fuzz

How to download cifuzz installer via shell
`curl -fsSL https://downloads.code-intelligence.com/download/cifuzz/linux_amd64/<version>?token=<access token>`

5. After that, cifuzz will be installed
6. Then, we create symbolic links of all llvm and clang tools to the cifuzz/bin folder, which is in path. This is to ensure that cifuzz will use the correct compiler (clang-15)
7. Adjust changelog.md[62]
8. Create a pull request to master branch, wait for approval
9. After merging to master branch, go to the merge commit, and create a tag of the current docker image version by clicking

. After that, the Jenkins CICD pipeline

4.3.2.3.5 Update APfuzzing.cmake template

Lastly and most importantly, we need to configure certain CMAKE options which will be enabled when the --fuzzing flag is set during a build. As mentioned before, after our changes in VwOsConanBase[63] and the build.sh[64] script, when the --fuzzing option is specified, the variable VWOS_CMAKE_CIFUZZING will be set in CMake.

We then introduce the APfuzzing.cmake[65] file to set all the required compiler and linker flags to enable a fuzzing run:

Table of all APfuzzing.cmake flags provided by Code-Intelligence

	FLAGS	Case 1: Fuzz Run Build	Case 2: Coverage Build
Required for Fuzz Test and SUT	CC	clang	clang
	CXX	clang++	clang++

62 http://changelog.md
63 https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/vwos_conan_base/browse
64 https://devstack.vwgroup.com/bitbucket/projects/E3INT/repos/docker_e3sdk_commonpart_build_bp1/browse/files/e3sdk-cli/commands/build.sh
65 https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/templates-e3comp/browse/cmake/APfuzzing.cmake

	CMAKE_C_FLAGS	-g -Og -fno-omit-frame-pointer -DFUZZING_BUILD_MODE_UNSAFE_FOR_PRODUCTION -UNDEBUG -fsanitize=fuzzer-no-link -fsanitize=address,undefined -fsanitize-recover=address -fsanitize-address-use-after-scope -U_FORTIFY_SOURCE -gdwarf-4	-g -Og -fno-omit-frame-pointer -DFUZZING_BUILD_MODE_UNSAFE_FOR_PRODUCTION -UNDEBUG -fprofile-instr-generate -fcoverage-mapping -U_FORTIFY_SOURCE -mllvm -runtime-counter-relocation -gdwarf-4
	CMAKE_CXX_FLAGS	-g -Og -fno-omit-frame-pointer -DFUZZING_BUILD_MODE_UNSAFE_FOR_PRODUCTION -UNDEBUG -fsanitize=fuzzer-no-link -fsanitize=address,undefined -fsanitize-recover=address -fsanitize-address-use-after-scope -U_FORTIFY_SOURCE -gdwarf-4	-g -Og -fno-omit-frame-pointer -DFUZZING_BUILD_MODE_UNSAFE_FOR_PRODUCTION -UNDEBUG -fprofile-instr-generate -fcoverage-mapping -U_FORTIFY_SOURCE -mllvm -runtime-counter-relocation -gdwarf-4
Required for Fuzz Test only	target_compile_options	-I/home/e3-user/.local/share/cifuzz/include	-I/home/dte/.local/share/cifuzz/include
	CMAKE_EXE_LINKER_FLAGS TODO: Find a way to link this only to the fuzz target	-Wl,--wrap=__sanitizer_set_death_callback -fsanitize=fuzzer,address,undefined	-fsanitize=fuzzer -fprofile-instr-generate
	target_link_libraries	/home/e3-user/.local/share/cifuzz/src/dumper.cpp	/home/e3-user/.local/share/cifuzz/src/dumper.cpp

Explaination of all the flags provided by Code Intelligence

C/C++ Flags			
Required for	**Flag**	**Explaination**	**Status of implementation**

C/C++ Flags			
all - both	-g	Keep debug symbols	Removed from APfuzzing. This is added by the build script by default Confirmed in buildlog
all - both	-Og	Do optimizations which don't harm debuggin	Removed from APfuzzing, as this should be managed by the profile. https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/templates-e3comp/pull-requests/97/overview?commentId=1657202 Confirmed that it is not in the build log
all - both	-fno-omit-frame-pointer	Enable better stack frames for easier debugging	Removed from APfuzzing. This is added by the build script by default Confirmed in buildlog
all - both	-DFUZZING_BUILD_MODE_UNSAFE_FOR_PRODUCTION	Conventional macro to conditionally compile out fuzzer road blocks See https://llvm.org/docs/LibFuzzer.html#fuzzer-friendly-build-mode	In APfuzzing Confirmed in buildlog
all - both	-UNDEBUG	Ensure that asserts are enabled regardless of compilation mode	In APfuzzing Confirmed in buildlog
all - coverage	-fprofile-instr-generate	Generate instrumented code to collect execution counts	In APfuzzing Confirmed in buildlog

C/C++ Flags			
all - coverage	-fcoverage-mapping	Generate coverage mapping to enable code coverage analysis	In APfuzzing Confirmed in buildlog
all - both	-U_FORTIFY_SOURCE	Disable source fortification which is currently not supported in ASan builds	Removed from APfuzzing. Not shown in build log. But it is fine since source fortification is disabled anyway.
all - coverage	-mllvm -runtime-counter-relocation	Support LLVM's continuous mode on non-macOS platforms. This support is unstable in <=clang13 so this is only used with >=clang14	In APfuzzing Confirmed in buildlog
all - both	-gdwarf-4	Enable better debugging	In APfuzzing Confirmed in buildlog
all - libfuzzer	-fsanitize=fuzzer	Compile your executable with edge coverage and correct instrumentation for fuzzing.	In APfuzzing Confirmed in buildlog
all - libfuzzer	-fsanitize=address,undefined	Build with instrumentation for ASan and UBSan	Removed from APfuzzing. Added manually with build.sh script Confirmed in buildlog
all - libfuzzer	-fsanitize-recover=address	Support recovering from ASan findings	Removed from APfuzzing. Added manually with build.sh script Confirmed in buildlog

C/C++ Flags			
all - libfuzzer	-fsanitize-address-use-after-scope	Use additional error detectors for use-after-scope vulnerabilities	Removed from APfuzzing. Added manually with build.sh script Confirmed in buildlog
fuzz target	-I/home/e3-user/.local/share/cifuzz/include	Add the cifuzz include directory to the list of directories to be searched for header files	In APfuzzing Confirmed in buildlog

	Linker Flags		
Required for	**Flag**	**Explaination**	**Status of implementation**
all - libfuzzer	-fsanitize=fuzzer-no-link	Make sure that non fuzzing target are not linked against libfuzzer main (e.g. sometimes there are 2 main methods => avoid double main issue)	In APfuzzing Confirmed in buildlog
fuzz target - libfuzzer	-Wl --wrap=__sanitizer_set_death_callback	Redirect calls to _sanitizer_set_death_callback to our implemented wrap _sanitizer_set_death_callback (in dumper.c/.cpp) to modify the behavior of the original libfuzzer function	In APfuzzing Confirmed in buildlog
fuzz target - both	-fsanitize=fuzzer	Build with instrumentation for fuzzing	In APfuzzing Confirmed in buildlog to have both fuzzer and fuzzer-no-link during linking

	Linker Flags		
fuzz target - libfuzzer	-fsanitize=address	Build with instrumentation for ASAN	Removed from APfuzzing. Added manually with build.sh script Confirmed in buildlog
fuzz target - libfuzzer	-fsanitize=undefined	Build with instrumentation for USAN	Removed from APfuzzing. Added manually with build.sh script Confirmed in buildlog
fuzz target - coverage	-fprofile-instr-generate	Generate instrumented code to collect execution counts	In APfuzzing Confirmed in buildlog
fuzz target - both	/home/e3-user/.local/share/cifuzz/src/dumper.cpp	Path to the dumper of CI Fuzz which ensures that non-fatal sanitizer findings still have an input attached	In APfuzzing Confirmed in buildlog

4.3.2.3.6 How the sanitizer flags are set during build

When call the build.sh script from PMT[66] team, one can specify following sanitizer flags:

66 https://devstack.vwgroup.com/bitbucket/projects/E3INT/repos/docker_e3sdk_commonpart_build_bp1/browse/files/e3sdk-cli/commands/build.sh

Sanitizer flags in build.sh script

```
--sanitizer=<address|undefined|thread>  Select sanitizers (can give multiple, separated
by ':')"
--sanitizer_args=<string>               Specify a colon-separated list with additional
arguments for the sanitizer e.g. --sanitizer_args=-O1:-fno-omit-frame-pointer:-g
```

These arguments are then parsed and passed down to VwOsConanBase[67], which all the conanfile.py inherits from. In VwOsConanBase[68], the cmake flag *VWOS_CMAKE_SANITIZERS* is populated with sanitizers values. Subsequently, this flag *VWOS_CMAKE_SANITIZERS* is then parsed and processed in std.cmake[69] file in the docker container. As the result, respective sanitizer compile and link flags are set via *vwos_add_sanitizer_link_options* and *vwos_add_sanitizer_compile_options*.

4.3.3 Prepare your VM

Create Virtual Maschine (VM) in VirtualBox with an installation of Ubuntu 22.04 Linux

Change network settings for this VM for the network Adapter 1:

1. Connected to: Networkbridge
2. Name: Choose the hardware interface with that your host is connected to with the internet

Install latest available updates on this VM with:

- > sudo apt update && apt upgrade

  If your connected through VPN to the Cariad network apply the following settings before:

  > TODO: Figure out settings for connecting to web/apt servers when connected to Cariad VPN

Install Virtual Box Extensions for the Ubuntu 22.04 VM.

Also request access for JFROG repositories at the service desk https://devstack.vwgroup.com/jira/servicedesk/customer/portals for the following repositories (example request can be found here: https://devstack.vwgroup.com/jira/servicedesk/customer/portal/6/ESSD-79548):

- e3-e3sp-conan-release
- e3-e3sp-conan-snapshot
- e3-e3sp-conan-app-release
- e3-e3sp-conan-app-snapshot
- e3-e3sp-conan-vwos-production

67 https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/vwos_conan_base/browse
68 https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/vwos_conan_base/browse
69 https://devstack.vwgroup.com/bitbucket/projects/E3SPARCH/repos/e3_arch_compilerflags/browse/cmake/VWOS/std.cmake

- e3-e3sp-conan-vwos-test
- e3-e3sp-sdk
- e3-e3sp-deb-repo
- e3-e3sp-deb-repo-test
- e3-e3sp-rpm-repo
- e3-e3sp-rpm-repo-tes

Install docker:

- > sudo apt install docker.io
- (TODO: Maybe change with official docker.io documentation https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository)

Check if current user is member of group "docker"

- > groups (executing in terminal should list the groups of the current user)

If current user is not in group "docker" add user to "docker" group:

- > sudo usermod -aG docker $USER

May new group "docker" needs to be created:

- > sudo newgrp docker

Create ssh key and add public ssh key to bitbucket (in the profile settings of bitbucket).

Clone repo (e.g. E3_COMP_OTAM_COMMON) to location of your choice):

- > git clone https://devstack.vwgroup.com/bitbucket/scm/e3swcdep/e3_comp_otam_common.git .

If needed install pip:

- > sudo apt install pip

Install conan:

- > python3 -m pip install -v "conan==1.64.1"

Verify installation of conan:

- > conan --version

Configure conan by replacing the current conan remote.json settings file in ***~/.conan/remotes.json*** with:

- ```
  {
   "remotes": [
    {
     "name": "e3-release",
     "url": "https://jfrog.devstack.vwgroup.com/artifactory/api/conan/e3-e3sp-conan-release",
     "verify_ssl": true
    },
    {
     "name": "e3-snapshot",
     "url": "https://jfrog.devstack.vwgroup.com/artifactory/api/conan/e3-e3sp-conan-snapshot",
     "verify_ssl": true
    },
    {
     "name": "e3-app-release",
     "url": "https://jfrog.devstack.vwgroup.com/artifactory/api/conan/e3-e3sp-conan-app-release",
     "verify_ssl": true
    },
    {
     "name": "e3-app-snapshot",
     "url": "https://jfrog.devstack.vwgroup.com/artifactory/api/conan/e3-e3sp-conan-app-snapshot",
     "verify_ssl": true
    },
    {
     "name": "e3-vwos-production",
     "url": "https://jfrog.devstack.vwgroup.com/artifactory/api/conan/e3-e3sp-conan-vwos-production",
     "verify_ssl": true
    },
    {
     "name": "e3-vwos-test",
     "url": "https://jfrog.devstack.vwgroup.com/artifactory/api/conan/e3-e3sp-conan-vwos-test",
     "verify_ssl": true
    }
   ]
  }
  ```

- Setup conan user for your user to access the remote repositories configured before:
  - > conan user YOURUSER -r e3-release
  - > conan user YOURUSER -r e3-snapshot
  - > conan user YOURUSER -r e3-app-release
  - > conan user YOURUSER -r e3-app-snapshot
  - > conan user YOURUSER -r e3-vwos-production
  ```

    - > conan user YOURUSER -r e3-vwos-test

- Edit conan settings file in ***~/.conan/settings.yml*** and add to the entry ***os*** → ***Linux*** the following:
    - ***target: [local]***
- Edit conan settings file in ***~/.conan/settings.yml*** and add to the entry ***os*** → ***linux-musl*** the following:
    - ***target: [hcp3-sys]***

- Login to docker and conan as described here: Q&A and Troubleshooting CIFuzz (see page 439):
    - > docker login e3-e3sp-docker-3rdparty.docker.devstack.vwgroup.com[70]
    - > docker login docker-public-virtual.docker.devstack.vwgroup.com[71]
    - > docker login e3-e3sp-dockerdev.docker.devstack.vwgroup.com[72]

4.3.4 Setting up cifuzz environment

After following this guide, you will have cifuzz running in a docker container.

Using this container enables you to ...

1. Run fuzzing tests with ASAN/UBSAN support
2. Create code coverage reports to evaluate the fuzzing progress
3. Display and debug findings

4.3.4.1 (Optional) Building the docker image locally

If for some reasons, the docker image is not available on the artefactory. You can build the container locally like follows:

Building the docker image locally

```
1  git clone ssh://git@devstack.vwgroup.com:7999/e3int/
   docker_e3sdk_build_bp1.git
2  cd docker_docker_e3sdk_build_bp1
```

70 http://e3-e3sp-docker-3rdparty.docker.devstack.vwgroup.com
71 http://docker-public-virtual.docker.devstack.vwgroup.com
72 http://e3-e3sp-dockerdev.docker.devstack.vwgroup.com

```
# Build the image locally. Provide your CARIAD's credential if promted by the script
./build-image.sh
# Finally, tag the docker image accordingly
docker tag local/docker_e3sdk_build_bp1:99.99.0 e3-e3sp-dockerdev.docker.devstack.vwgroup.com/local/docker_e3sdk_build_bp1:99.99.0
```

4.3.4.2 Install cifuzz tool on your local machine

Eventhough we have prepared a docker image with cifuzz tools, it is helpful to have cifuzz tool installed on your local machine as well. Ultimately, you are going to use your IDE e.g., Visual Studio Code, to write code on your local machine and not our prepared docker image. In this case, having cifuzz tool on your local machine will help with:

- Auto Code Completion in the IDE
- Using cifuzz tool to create fuzz target template

In order to install cifuzz on your local machine, please follow the official guide from Code Intelligence[73]. And don't forget to include the cifuzz bin directory to your path. One way to do this in linux is like follows:

Building the docker image locally

```
echo "export PATH=\"/home/bi/.local/share/cifuzz/bin:\$PATH\"" >> ~/.profile
source ~/.profile
```

or install via curl:

install cifuzz via curl command

```
sh -c "$(curl -fsSL http://downloads.code-intelligence.com/assets/install-cifuzz.sh)"
 d436ba43-d764-4a6e-9110-5cf3e91be8d7 2.26.1
```

to enable auto-completion in VScode you might have to do the following:

Enable autopcomplete in VSCode

```
Add "/home/<insert user name>/.local/share/cifuzz/include" to your include path in
 VScode
```

73 https://docs.code-intelligence.com/ci-fuzz/ci-fuzz-overview

```
To see your include options in VSCode go to:
settings -> C_Cpp › Default: Include Path and add the line on the top
```

Alternatively, if you don't want to install cifuzz on your local machine, but still want to use the cifuzz tool to create fuzz target templates, we have prepared a script. You can download the latest version of the cifuzz.sh script[74] from our Security Team Repository. The script syntax is as follow:

```
./cifuzz.sh [arguments]
```

This shell script will forward all arguments (except for some specific options) to the docker container and execute the followings inside the docker container.

However, before using the script, you should change the name of the docker image you use to build accordingly in the cifuzz.sh[75] itself.

```
docker_image="<THE NAME OF THE DOCKER IMAGE WITH CIFUZZ TOOLING USED FOR BUILDING>"
```

After configuring the script accordingly and also pulling the docker image from the artefactory or building it locally, you should be able to execute following commands:

As an example, the result of "./cifuzz.sh help" should be:

./cifuzz.sh help

```
./cifuzz.sh help
Usage:
  cifuzz [command]

Available Commands:
  init        Set up a project for use with cifuzz
  create      Create a new fuzz test
  run         Build and run a fuzz test
  reload      Reload fuzz test metadata
  bundle      Bundles fuzz tests into an archive
  coverage    Generate a coverage report for a fuzz test
  finding     List and show findings
  integrate   Add integrations for the following tools: Git, CMake, VS Code
  help        Help about any command
  completion  Generate the autocompletion script for the specified shell
```

74 https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3-aux-security-repo/browse/DAST/cifuzz?at=refs%2Fheads%2Fdevelop

75 https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3-aux-security-repo/browse/DAST/cifuzz?at=refs%2Fheads%2Fdevelop

```
Flags:
  -C, --directory string   Change the directory before performing any operations
      --help               Show help for command
  -v, --verbose            Show more verbose output, can be helpful for debugging
problems

Use "cifuzz [command] --help" for more information about a command.
```

In addition to that, the script itself provides additional functionality:

Print the release tag & git hash of the cifuzz tool

```
./cifuzz.sh /version
v0.4.0-0-g77405c4

# Please state this version if you require support
```

Enter the cifuzz enabled container. Useful for debugging

```
e3sdk_scripts/shell.sh --docker --target fuzzing
e3-user@d9cd314ac596:/home/developer/cifuzz-0.4.0-clang15$
```

To know about further options, you can execute help.

```
$ ./cifuzz.sh /help
Usage: ./cifuzz.sh [options]
Options:
  run [fuzztarget name]                                      Run fuzzing on the fuzz
target defined inside fuzzing directory without .cpp ending
  /shell                                                     Login to the docker
container with a shell session. Use for debugging
  /debug [path to fuzztarget bin] [path to fuzz input]       Start a gdbserver inside
the docker container to debug with VSCode at localhost:55555
  /version                                                   Show the version of cifuzz
being used
  /help                                                      Print this help message
  *                                                          Pipe all arguments in ./
cifuzz.sh [arguments] and execute "cifuzz [args]" inside the docker container
```

4.3.4.3 Prepare your component folder for cifuzz

⚠️ Note that you can use both `cifuzz` and `cifuzz.sh`[76] the same way since the cifuzz.sh script created by us only forwards all the arguments specified into the docker container. And the directory you call the cifuzz.sh script from is mounted into the docker container, everything created inside that directory inside the docker container will also be visible in your local machine.

In this tutorial, we use our `cifuzz.sh`[77] option. In order to prepare your component repository folder for fuzzing, you must do the followings:

- Copy `cifuzz.sh`[78] into your component folder
- Call `cifuzz.sh`[79] /setup to setup your local directory:

```
./cifuzz.sh /setup
```

- Edit root CMakeLists.txt file like so:

Reorder addsubdirectory calls, call add_cifuzz_build_options

```
add_subdirectory(cmake)

if(VWOS_CMAKE_CIFUZZING)
    add_cifuzz_build_options()
endif()
...
add_subdirectory(src)
add_subdirectory(tests)
```

Our script "`cifuzz.sh`[80] /setup" command basically does the followings:

- Getting preconfigured cifuzz.yaml from our repo with following content:
  a. Set `build-system: other`

76 https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3-aux-security-repo/browse/DAST/cifuzz?at=refs%2Fheads%2Fdevelop
77 https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3-aux-security-repo/browse/DAST/cifuzz?at=refs%2Fheads%2Fdevelop
78 https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3-aux-security-repo/browse/DAST/cifuzz?at=refs%2Fheads%2Fdevelop
79 https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3-aux-security-repo/browse/DAST/cifuzz?at=refs%2Fheads%2Fdevelop
80 https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3-aux-security-repo/browse/DAST/cifuzz?at=refs%2Fheads%2Fdevelop

b. Set `use-sandbox: false`

c. Set `build-command: build-command: "echo Building Fuzz Target using custom build process"`

- Add APfuzzing.cmake and the new CMakeLists.txt to your cmake directory from the templates-e3comp[81] reporsitory. After that, your project folder should have following structure:

```
repo/
├─ cmake/
│  ├─ APfuzzing.cmake
│  ├─ CMakeLists.txt
├─ CMakeLists.txt
├─ .../
```

- Setup a directory for fuzz tests within your repository directory likes so:

```
repo/
├─ tests/
│  ├─ fuzzing/
│  │  ├─ CMakeLists.txt
│  │  ├─ FuzzTarget.cpp
```

- Edit `e3sdk_metadata` to add a target for fuzzing like so:

Add this target to the existing e3sdk_metadata

```
# TODO: change the profile fuzzing_x86_64 to a new one once all dependencies are built with clang
"fuzzing": {
    "ignore_target": "false",
    "build_container": "master/e3sdk-build-bp1:4.5.0",
    "conan_build_profile": "--profile fuzzing_x86_64",
    "blackDuckScaActive": "false"
}
```

⚠️ If you chose to build your docker container locally, then you should change the name of the container to `local/build-bp1-commonpart:99.99.0` as shown above.

- Adding the conan profile fuzzing_x86_64[82] for fuzzing to your local machine like so
  # TODO: change the profile fuzzing_x86_64 to a new one once all dependencies are built with clang

```
/$HOME/
├─ .conan
```

81 https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/templates-e3comp/browse/cmake

82 https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3-aux-security-repo/browse/DAST/cifuzz/fuzzing_x86_64?at=refs%2Fheads%2Fdevelop

```
│ ├─ profiles
│ │ ├─ fuzzing_x86_64
```

As examples for this setup, please check out our example chapter below. (see page 101)

Our script also has an update function " `cifuzz.sh`[83] /update" which basically does the followings:

- Updating the local cifuzz.sh script itself with the new version from e3-aux-security-repo repo
- Updating the APfuzzing.cmake file from repo
- Downloading the new conan profile for fuzzing and updated the one in "$HOME/.conan/profiles/"

4.4 Creation of fuzz target

1. With the provided script a fuzz-test-stub can be created

```
# cpp language selection
# -C, --directory string   Change the directory before performing any operations
# -o, --output string      File path of new fuzz test

# The naming convention for fuzz-targets is as follows : fuzz_<name of component,
function or class to fuzz in the target>

./cifuzz.sh create cpp -C tests/fuzzing/ -o <fuzz_target_name>.cpp
```

2. `CMakeLists.txt` must be created inside of tests/ `fuzzing/` with following content:

Example: fuzzing/CMakeLists.txt

```
cmake_minimum_required(VERSION 3.16)
# Note that this might vary for different project since each project names their
libraries differently
set(LINK_LIBS
    ${TARGET_NAME}_lib
    generatedlib
)
add_cifuzz_executable(<fuzz_target_name> <fuzz_target_src_files>)
target_link_libraries(<fuzz_target_name> LINK_PUBLIC ${LINK_LIBS})
```

3. A template fuzz-stub is created by the create command

83 https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3-aux-security-repo/browse/DAST/cifuzz?at=refs%2Fheads%2Fdevelop

fuzzing/my_fuzz_test1.cpp

```
#include <assert.h>

#include <cifuzz/cifuzz.h>
#include <fuzzer/FuzzedDataProvider.h>

FUZZ_TEST_SETUP() {
  // Perform any one-time setup required by the FUZZ_TEST function.
}

FUZZ_TEST(const uint8_t *data, size_t size) {
  // FuzzedDataProvider provides convenience methods that turn the raw fuzzer
  // data into common types. Use it to generate parameters for the function you
  // want to fuzz:
  //
  // FuzzedDataProvider fuzzed_data(data, size);
  // int my_int = fuzzed_data.ConsumeIntegral<int8_t>();
  // std::string my_string = fuzzed_data.ConsumeRandomLengthString();

  // Call the functions you want to test with the provided data and optionally
  // assert that the results are as expected:
  //
  // int res = DoSomething(my_int, my_string);
  // assert(res != -1);

  // If you want to know more about writing fuzz tests you can checkout the
  // example projects at https://github.com/CodeIntelligenceTesting/cifuzz/tree/main/examples
  // or have a look at our tutorial:
  // https://github.com/CodeIntelligenceTesting/cifuzz/blob/main/docs/How-To-Write-A-Fuzz-Test.md
}
```

The template defines two functions with documentation which makes them self-explaining.

`FUZZ_TEST_SETUP()`
... is just called once per run. Use it to initialize static content.

`FUZZ_TEST()`
... is called multiple times and is supplied with the generated fuzz data. The fuzzer starts with small and simple inputs and grows until the defined upper limit (default: 4kB).

Depending on the specification of the code under test it may be required to capture exceptions in this function.

As this function may be called millions of time during one run, care must be taken not to leak any resources during setup, execution and tear down.

4. When writing the fuzz-target the most basic way of giving input to the "code under test" is through function parameters. The fuzzing engine continuously mutates bytes and provides them to the `FUZZ_TEST()` function. This bytes can be converted into valid data types with a class named FuzzedDataProvider. It is initialized with data and size and provides methods to extract a set of different data types:

Examples FuzzedDataProvider

```
#include <assert.h>

#include <cifuzz/cifuzz.h>
#include <fuzzer/FuzzedDataProvider.h>

#include "AddrIPv6andPort.hpp"


FUZZ_TEST_SETUP() {
  // Perform any one-time setup required by the FUZZ_TEST function.
}

FUZZ_TEST(const uint8_t *data, size_t size) {
    FuzzedDataProvider data_provider(data, size);

    std::vector<uint8_t> addr = data_provider.ConsumeBytes<uint8_t>(16U);
    uint16_t port = data_provider.ConsumeIntegral<uint16_t>();

    vwg::e3p::rsp::AddrIPv6andPort addr_and_port(addr, port);
}
```

5. To build the fuzz-target the following command is used. After building, the libfuzzer binary can be found in

```
# --verbose flag is recommended to have more input on what is happening during the build process
# -j2 is also recommended, this makes the build happen in parallel/multithreaded

# TODO: change the profile fuzzing_x86_64 to a new one once all dependencies are built with clang
./build.sh --docker --target fuzzing --profile fuzzing_x86_64 --buildfolder=<path to build dir> --fuzzing --sanitizer=address:undefined --sanitizer-args=-fsanitize-recover=address:-fsanitize-address-use-after-scope
```

6. To run the fuzz-target the following command is used

```
./cifuzz.sh run <build dir>/bin/<fuzz_target_name> -e
ASAN_OPTIONS=strict_string_checks=1:detect_stack_use_after_return=1:che
ck_initialization_order=1:strict_init_order=1:abort_on_error=0:halt_on_
error=0
```

4.5 Maintaining a fuzz target

After a basic fuzz-test has been running for some time a test engineer should look over the results. With this information the fuzz-test can be improved to achieve more coverage or more branching.

4.5.1 Handling Fuzz Targets expecting different inputs

4.5.1.1 Injecting input via dynamic linking

> *"Certain tools (such as American Fuzzy Lop, for example) are unable to handle network binaries. Preeny includes two "de-socketing" modules.* `desock.so`[84] *neuters* `socket()`, `bind()`, `listen()`, *and* `accept()`, *making it return sockets that are, through hackish ways, synchronized to* `stdin` *and* `stdout`. `desock_dup.so`[85] *is a simpler version for programs that dup accepted sockets over file descriptors 0, 1, and 2."* [15]

No pilot done as of now.

4.5.1.2 Input via file

If the code under test reads input from a file (or preferably a file handle) we can create a temporary file with the input from the fuzzing engine and hand over the file name to the code under test.

Example when a file path should be provided as input

```
...

#define FUZZING_TEMP_BASEPATH   "/dev/shm"

//
// Some code omitted for clarity.
// See  https://github.com/google/security-research-pocs/blob/master/
autofuzz/fuzz_utils.cc
// for the original source
//

FUZZ_TEST_SETUP() {
```

84 http://desock.so

85 http://desock_dup.so

```
  // FileProc does not allow arbitrary paths. Whitelist the base path.
  // FileProc::addTrustedPath() is a static function and must only be executed once
  vwg::e3p::diag::lum::mfw::FileProc::addTrustedPath(FUZZING_TEMP_BASEPATH);
}

FUZZ_TEST(const uint8_t *data, size_t size) {
  // write the content of the given array into a temporary file
  // and return the name to that file
  char* pFilename = buf_to_file(data, size);

  if ( nullptr == pFilename ) {
    warn( "FAILED to create temp file");
    return;
  }

  Config config;
  ConfigParser configParser(config);

  std::string path(pFilename);
  configParser.loadConfig(path);

  // reset state and free all locally allocated resources
  delete_file(pFilename);
}
```

Repo: https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_lum/browse/fuzzing/fuzz_ConfigParser_loadConfig.cpp?at=custom%2Fcifuzz

In this example the fuzzing input is fed via istringstream to the function under test.

Examples File input - stream

```
...

FUZZ_TEST(const uint8_t *data, size_t size)
{
  std::string content(&data[0], &data[size]); // string from array
  std::istringstream isstream{content};       // istringstream from string
  PersistencyParser persistencyParser;

  try {
    persistencyParser.parse(isstream);
  } catch ( mfw::FileCorruptException e ) {
    // do nothing, this is expected
  }
}
```

Repo: https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_lum/browse/fuzzing/fuzz_PersistencyParser.cpp?at=custom%2Fcifuzz

4.5.2 Improve by adding a switch case

When fuzzing multiple members of a class it is recommended to use a switch case like so:

switch case example

```
// taken from https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_tm_base/
browse/tests/fuzzing/fuzz_Time_Diff_Calculation.cpp?at=refs%2Fheads%2Fcustom%2Ffuzzing-
setup

FUZZ_TEST(const uint8_t *data, size_t size) {

  FuzzedDataProvider fdp(data, size);

  const tmbase::Internal_Abs_Time_Type_Reduced abs_time_red_0 =
{fdp.ConsumeIntegral<uint32>(), fdp.ConsumeIntegral<uint32>()};
  const tmbase::Internal_Abs_Time_Type_Reduced abs_time_red_1 =
{fdp.ConsumeIntegral<uint32>(), fdp.ConsumeIntegral<uint32>()};
  const tmbase::Internal_Abs_Time_Type offset_time{fdp.ConsumeIntegral<sint64>(),
fdp.ConsumeIntegral<uint32>()};
  const tmbase::Internal_Abs_Time_Type abs_time_0{fdp.ConsumeIntegral<sint64>(),
fdp.ConsumeIntegral<uint32>()};

  try
  {
    switch (fdp.ConsumeIntegralInRange<int>(0, 4))
    {
    case 0:
      calc_abs_difference_abs_times(abs_time_red_0, abs_time_red_1);
      break;
    case 1:
      calc_sum_abs_time_and_offset(abs_time_red_0, offset_time);
      break;
    case 2:
      calc_sum_abs_time_and_offset(abs_time_0, offset_time);
      break;
    case 3:
      round_abs_time_to_seconds(abs_time_0);
      break;
    case 4:
      round_abs_time_to_seconds(abs_time_red_0);
      break;

    default:
      break;
    }
  }
  catch(const std::exception& e)
  {
    return; // We expect exceptions
  }
}
```

4.5.3 Improve by setting cifuzz runtime flag

The most simple way to immediately improve the code coverage is to set the "use_value_profile" runtime cifuzz flag. This is a LLVM libFuzzer flag which collect value profiles for the parameters of compare instruction e.g., if/else and treat some new values as new coverage. In short, with this flag set to 1, we will achieve the followings:

- Every matching bit in the CMP instructions (e.g., if/else statement) will be treated as new coverage. (see llvm doc[86])

Setting -use_value_profile flag

```
cifuzz run build/bin/fuzz_TLRMessage --build-command "echo hi" --engine-arg="-
use_value_profile=1"
```

This will immediately help the code coverage to CMP instruction if the fuzzer is stuck. However, this does come with a trade-off, since it **reduces the overall fuzzing performance**.

4.5.4 Improve fuzzing via dictionary

Command to use a dictionary

```
cifuzz run <build dir>/bin/<fuzz_target_name> --dict="dictionary.txt" [7]
```

Dictionary example [8]

```
# Adds "blah" (w/o quotes) to the dictionary.
kw1="blah"
# Use \\ for backslash and \" for quotes.
kw2="\"ac\\dc\""
# Use \xAB for hex values
kw3="\xF7\xF8"
# the name of the keyword followed by '=' may be omitted:
"foo\x0Abar"
```

If stuck behind a comparison this type of improvement can also work to bypass the block. The fuzzer will take the words in a dictionary in consideration when generating inputs. This approach can improve

86 https://llvm.org/docs/LibFuzzer.html#value-profile

the branching and coverage of a target. In addition to this it does not slowdown the execution nor will the corpus increase its size by several times.

An example for a block of this type: [6]

Block at comparison

```
...
  if (my_string.compare("Guess this String!") == 0) {
    if (my_int == 0xE26DB107) {
      ... // to get to this block, Value using a dictionary is a valid approach.
    }
  }
...
```

4.5.5 Improve fuzzing via Value Profile

Value Profile command

```
cifuzz run <build dir>/bin/<fuzz_target_name> --engine-arg="-use_value_profile=1" [7]
```

If stuck behind a comparison this type of improvement can work to bypass the block. The fuzzer will collect value profiles for the parameters of compare instructions and treat new values as new coverage. This approach has two downsides. First, the extra instrumentation may bring up to 2x additional slowdown. Second, the corpus may grow by several times. [5]

An example for a block of this type: [6]

Block at comparison

```
...
  if (my_string.compare("Guess this String!") == 0) {
    if (my_int == 0xE26DB107) {
      ... // to get to this block, Value Profile is a valid approach.
    }
  }
...
```

4.5.6 Improve fuzzing via preparing corpus

By providing a directory filled with structured data (seeds) an engineer can give the fuzzer a starting point to generate inputs that will provide a better result.

Example of a seed-corpus

```
directory/with/seed-corpus/

├── seed1.png

├── seed2.png

└── seed3.png

# The seeds can be any type of data one wants the fuzzer to use. Also data that is
malformed should be provided.
```

Command to specify seed-corpus

```
cifuzz run -s <directory/with/seed-corpus/> <build dir>/bin/<fuzz_target_name>
```

4.5.7 Improve fuzzing via Custom Mutator

When a fuzz-target requires structured input a custom mutator can be written to achieve structured inputs. This approach can decrease execution speeds and uses extra time for an engineer to write. Writing a good seed corpus can create a similar benefit without the here mentioned drawbacks.

A Custom mutator must be defined in the *LLVMFuzzerCustomMutator(...)* function. The fuzzer will use this function to modify the data in it a structure can be given to the data.

Mutator example \[9\]

```
#include <cstdint>
#include <cstdlib>
#include "png.h" // Include the PNG library header

extern "C" size_t LLVMFuzzerCustomMutator(uint8_t *data, size_t size, size_t max_size,
unsigned int seed) {
  // Initialize the PNG structures with custom error handling
```

```
  png_structp png_ptr = png_create_read_struct(PNG_LIBPNG_VER_STRING, nullptr, nullptr,
nullptr);
  png_infop info_ptr = png_create_info_struct(png_ptr);

  // Set up the custom read function to read from the data buffer
  png_set_read_fn(png_ptr, &data, custom_read_function);

  // Read the PNG file header and validate the signature
  if (setjmp(png_jmpbuf(png_ptr))) {
    // Invalid PNG signature, clean up and return
    png_destroy_read_struct(&png_ptr, &info_ptr, nullptr);
    return size;
  }
  png_read_info(png_ptr, info_ptr);

  // Mutate the PNG file intelligently
  mutate_IHDR(png_ptr, info_ptr);
  // Add more mutation functions as needed

  // Write the mutated PNG data to a new buffer
  uint8_t *new_data = new uint8_t[max_size];
  png_set_write_fn(png_ptr, &new_data, custom_write_function, custom_flush_function);
  png_write_png(png_ptr, info_ptr, PNG_TRANSFORM_IDENTITY, nullptr);

  // Clean up
  png_destroy_read_struct(&png_ptr, &info_ptr, nullptr);

  // Copy the mutated data back to the input buffer, respecting the max_size constraint
  size_t new_size = std::min(static_cast<size_t>(max_size),
get_new_data_size(new_data));
  memcpy(data, new_data, new_size);
  delete[] new_data;

  return new_size;
}
```

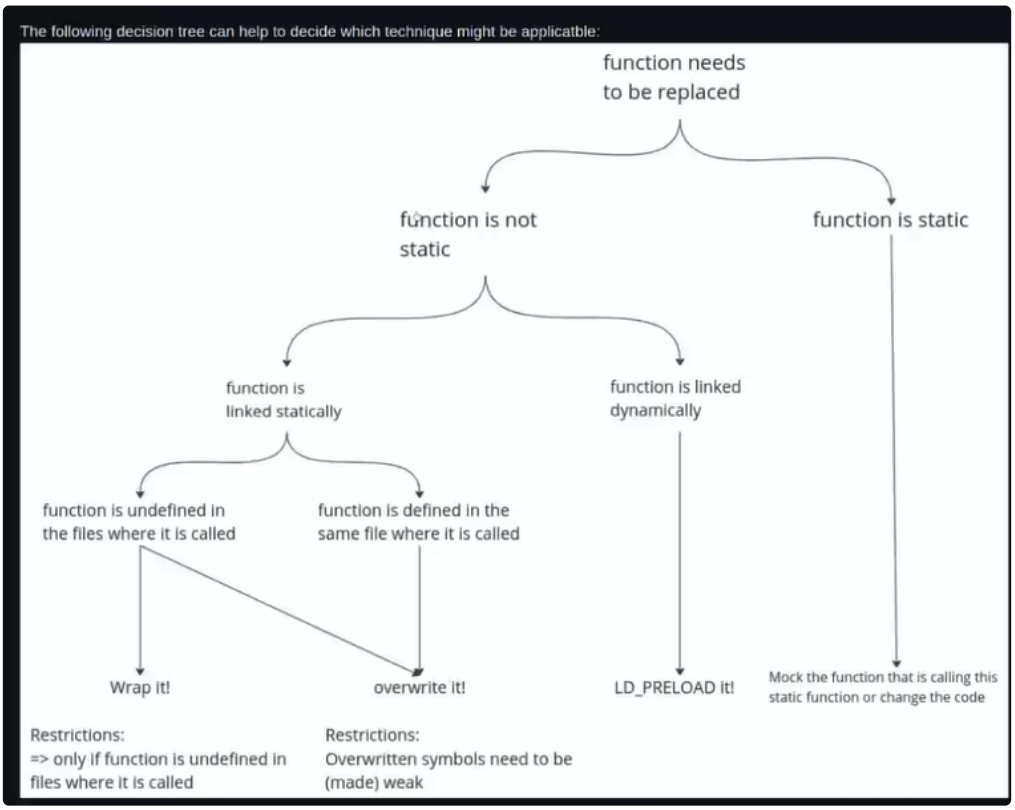
4.5.8 Mocking

Mocking is essential for not only unit testing but also for fuzz testing.

By mocking a function, testers can simulate its behavior without actually executing the real code. This is especially useful when dealing with functions that involve external dependencies, such as databases, APIs, or network services. Mocking these dependencies ensures that the fuzz testing process remains isolated and doesn't affect external systems. Additionally, mocking functions provides a way to focus on specific parts of the codebase without running the entire application.

However, depends on how the function you want to mock is implemented and linked, there are different strategies to mock them. The following figure shows which strategy to use in which situation:

TODO: Beautify this graph



4.5.8.1 Using GMock

Sometimes the functions can be mocked with GMock, similar to the situation in unit tests, a prerequisite is the function we want ot mock has a template calss we can use for GMock to work.

GMock example

```
...

FUZZ_TEST(const uint8_t *data, size_t size) {
  FuzzedDataProvider fuzzed_data(data, size);
  ::ara::stdtypes::String rand_kServiceId = fuzzed_data.ConsumeRandomLengthString();
  ::ara::stdtypes::String rand_kOperationId = fuzzed_data.ConsumeRandomLengthString();
  ::ara::stdtypes::String randURL = fuzzed_data.ConsumeRandomLengthString();
  // Create the mock object every fuzz iteration
  ServiceMock otaComMock;
  // Set the fuzz object to default value every fuzz iteration
  otaComMock.proxy.initDefault();
  vwg::otamaster::helpers::OTACommunicationCommHelper helper;

  // promise will never be set
  ac::core::Promise<Methods::getServiceListUrl::Output> returnPromise;
  returnPromise.set_value({randURL});
  EXPECT_CALL(otaComMock.getServiceListUrl,
```

```
18
call(otaComMock.getServiceListUrl.makeArgsTuple(rand_kServiceId,
rand_kOperationId)))
19          .WillOnce([&returnPromise](auto const&) {
20              // invalid future
21              auto future = returnPromise.get_future();
22              future.GetResult();
23              // next access will raise ara::core::FutureException(
24              // ara::core::future_errc::promise_already_satisfied)
25              return future;
26          });
27
28    helper.getServiceListUrl(rand_kServiceId, rand_kOperationId);
29  }
30
31  // For more details visit: https://devstack.vwgroup.com/bitbucket/projects/
E3SWCDEP/repos/e3_comp_otam_common/browse/tests/fuzzing/
fuzz_OTACommunicationCommHelper.cpp?
at=refs%2Fheads%2Fcustom%2Fcifuzz_2024_01
```

4.5.8.2 Wrapping symbols via the linker

GMock can not be used to mock free functions like socket(), recv(), or send() [14]. For following situation, this following approach should be used:

Requirements	Mocking Solution
Function is not static Function is linked statically Function is not defined in the files where it is called	Wrap the function

This can be achived with some help from the linker:

Examples Linker function wrapping (for C language)

```
1   ...
2
3   FuzzedDataProvider* DP = nullptr;
4
5   extern "C" ssize_t __wrap_recv(int sockfd, void* buf, size_t len, int flags)
    {
6       // will be entered when "recv" is called. Advanced usages would for
    example
7       // evaluate sockfd to select which socket to send the fuzzing input
    into.
8       // The real recv can be called via __real_recv(...)
9       auto output_buf = static_cast<uint8_t*>(buf);
```

```
    auto bytes = DP->ConsumeBytes<uint8_t>(len);
    ssize_t real_length = bytes.size();

    std::copy(bytes.begin(), bytes.end(), output_buf);
    return real_length;
}

int receiveDataFromSocket() {
    int sockfd = 460;
    char buffer[1024];

    // This is where the input is coming in:
    ssize_t received = recv(sockfd, buffer, 1024, 0);

    if(received <= 0){
        return 1;
    }

    // Some if blocks for the fuzzing engine to discover:
    if(received >= 5){
        if(buffer[0] == 'X')
            if(buffer[1] == 'y')
                if(buffer[2] == 'Z'){
                    return 0;
                }
    }
    return 2;
}

...

FUZZ_TEST(const uint8_t *data, size_t size) {
    FuzzedDataProvider data_provider(data, size);
    DP = &data_provider; // Make data_provider available to other functions

    auto ret = functionBeingFuzzed(); // Has no input parameters, as input is received from socket
    std::cout << ret << " ";
}
```

Linker options have to be added to CMakeLists.txt:

CMakeFile.txt

```
add_cifuzz_executable(wraptest, wraptest.cpp)
target_link_libraries(wraptest PUBLIC ${LINK_LIBS})
target_link_options(wraptest PUBLIC "LINKER:--wrap,recv")
```

Given the wrap-recv example from above, the linker performs some renaming:

- The object file wraptest.o requires a definition of symbol “recv”, which will be provided by e.g. libc.so[87].
- With the wrap-recv option the linker adds the prefix “__wrap_” to the required symbol, and it also adds the prefix “__real_” to the provided symbol.

Without any glue-code linking will fail, as the required symbol is not provided.
You may now implement that glue code and define the “__wrap_”-variant to your needs.
You may also call the original function under its new name:

Example: wrapped function

```
// prototype is required as compiler is unaware of the wrapping
extern "C" ssize_t __real_recv(int sockfd, void* buf, size_t len, int flags);

extern "C" ssize_t __wrap_recv(int sockfd, void* buf, size_t len, int flags) {
  // do some stuff here
  if ( len % 2 == 0 ) {
    return 0;
  }
  return __real_recv(sockfd, buf, len, flags); // call original function
}
```

Note that this naming approach only works for C language. In order to wrap functions in C++, you must use the mangled names ("*the encoding of function and variable names into unique names so that linkers can separate common names in the language*"). To find the mangeled name of a specifc function, execute:

Command to find the mangeled name

```
$ nm [path to .o/.a file] | less
$ find . -iname "[pattern]*.o" // this can be used to find the .o file
```

Further info:

- Using LD, the GNU linker - Options[88], Ctrl+F "wrap"
- Full size example: see commits associated with E3-102322[89] using wrap on e3_comp_socks_proxy

87 http://libc.so
88 https://ftp.gnu.org/old-gnu/Manuals/ld-2.9.1/html_node/ld_3.html
89 https://devstack.vwgroup.com/bitbucket/plugins/servlet/jira-integration/issues/E3-102322

4.5.8.3 Overwrite and weaken the function

In some cases, it is not possbile to wrap the function as described above. This is because, the function we want to wrap is defined in the same file where it is called. Example:

source.cpp Example: The function is defined in the same file where it is called

```
...
// We want to mock this function
void to_mock();

// This function is to be fuzzed
void function_to_fuzz(std::string argument){
    // The function to fuzz calls the function to_mock in the same file
    to_mock();
}
...
_Z7to_mockv
```

In this case, wrapping the function *to_mock* is not possible, since the to_mock function definition is in the same file where it is called. We need to overwrite the function *to_mock()* with our mocked version instead.

Requirements	Mocking Solution
Function is not static Function is linked statically Function is defined in the files where it is called	Overwrite and weaken the symbol of the original function

First, we write our own version of the fucntion to_mock in our fuzz target file. Make sure that our definition of the function to_mock has the **exact same function header** definition as the original one.

my_fuzz_test1.cpp

```
1   void to_mock(){
2       // our own definition
3   }
4
5   FUZZ_TEST_SETUP()
6   {
7   ...
8   }
9
10  // This function is to be fuzzed
11  FUZZ_TEST(const uint8_t* data, size_t size)
12  {
13  ...
```

```
}
...
_Z7to_mockv
```

Next, we build the project . It is fine if this step fails. It is only important that the file with the original *to_mock* function is compiled. Now look in the build directory for any .a, .o or .so file with the same name of the source file containing the *to_mock* function.

Find the compiled file of the source file

```
$ find ./build -iname "source*"
> build/src/CMakeFiles/project.dir/helpers/source.cpp.o
```

Now, we need to find the **mangle** name of the function *to_mock* in the compiled source file we just found. To do so, execute and look for the mangle name like follow:

Use nm command to look for the mangled name of the function

```
$ nm build/src/CMakeFiles/project.dir/helpers/source.cpp.o
> ...
> 0000000000000000 T _Z7to_mockv
> ...
```

In this case, *_Z7to_mockv* is the mangled name of the function *to_mock* we are looking for.
Next, we need to modify the CMakeLists.txt file where our fuzz target is defined like follow to weaken the original definition of the function *to_mock*

tests/fuzzing/CMakeLists.txt

```
cmake_minimum_required(VERSION 3.16)

set(TARGET_NAME example)

# Define the fuzz target, source files and link any library to it
add_cifuzz_executable(my_fuzz_test1, my_fuzz_test1.cpp ...)
target_link_libraries(my_fuzz_test1 LINK_PUBLIC ...)

# Weaken the original definition of function to_mock
add_custom_command(TARGET my_fuzz_test1 PRE_BUILD COMMAND objcopy
    --globalize-symbol=_Z7to_mockv
    --weaken-symbol=_Z7to_mockv
    $<TARGET_FILE:${TARGET_NAME}>
    COMMENT "Enabling wrapping static function to_mock")
```

Finally, build the fuzz target and enjoy.

4.5.9 Debugging a fuzz target

4.5.9.1 Analyze build problems

When a fuzz-target generates errors while building. An engineer can increase the verbosity of outputs by:

Verbose options

```
# -v : This flag activates verbose mode
./cifuzz.sh run <build dir>/bin/<fuzz_target_name> -v

# this variable can be set on in a cmake file
set(CMAKE_VERBOSE_MAKEFILE ON)
```

4.5.9.2 Use coverage information to find coverage blockers

At some stage the fuzz-run will not reach any new edges and coverage will stop growing.

CIFuzz progress indication

```
# 00:13:36 - still progress: the last edge was found 8 seconds ago
paths: 3947 - last new Path: 8s - exec/s: 753 (13m36s)

# 01:23:20 - likely saturated: the last edge was found 7 minutes ago
paths: 4814 - last new Path: 7m13s - exec/s: 697 (1h23m20s)
```

When creating a coverage report the existing corpus is used by CIFuzz to execute all known branches. The number of branches is not related to the number of paths.

Coverage report commands

```
# TODO: change the profile fuzzing_x86_64 to a new one once all dependencies are built
with clang
# Before generating a report, we must build the fuzz target with coverage flags first
via
./build.sh --docker --target fuzzing --coverage --profile fuzzing_x86_64 --
buildfolder=<path to coverage build dir> --verbose --fuzzing -j<#threads for parallel
building>

# LCOV format report
```

```
# =============
./cifuzz.sh coverage --add-corpus <path to corpus directory> <path to coverage
libfuzzer binary file> --output=<path to output lcov file>.lcov --format=lcov
# Example:
# ./cifuzz.sh coverage --add-corpus .cifuzz-corpus/build/bin/fuzz_BucketHandler/ build-
cov/bin/fuzz_BucketHandler --output=tests/fuzzing/fuzz_BucketHandler.lcov --format=lcov

# HTML detailed report
# ===============
./cifuzz.sh coverage --add-corpus <path to corpus directory> <path to coverage
libfuzzer binary file> --output=<path to output lcov file>.html
# Example:
# ./cifuzz.sh coverage --add-corpus .cifuzz-corpus/build/bin/fuzz_BucketHandler/ build-
cov/bin/fuzz_BucketHandler --output=tests/fuzzing/fuzz_BucketHandler_cov.html
```

LCOV - code coverage report

			Hit	Total	Coverage
Current view:	top level - src/FileProcessing/Config				
Test:	fuzz_ManifestReader.lcov	**Lines:**	189	420	45.0 %
Date:	2022-12-15 20:24:52	**Functions:**	40	77	51.9 %
		Branches:	43	238	18.1 %

Filename	Line Coverage			Functions		Branches	
Config.cpp		52.5 %	107 / 204	11.1 %	1 / 9	0.0 %	0 / 36
Config.hpp		40.9 %	27 / 66	48.0 %	12 / 25	22.9 %	11 / 48
ConfigItem.cpp		100.0 %	4 / 4	100.0 %	2 / 2	-	0 / 0
ConfigItem.hpp		73.1 %	19 / 26	77.8 %	7 / 9	16.7 %	1 / 6
ConfigItem.ipp		36.0 %	9 / 25	33.3 %	5 / 15	20.8 %	25 / 120
ConfigItemCheck.cpp		5.9 %	4 / 68	40.0 %	2 / 5	0.0 %	0 / 12
ConfigItemCheck.hpp		100.0 %	6 / 6	100.0 %	4 / 4	-	0 / 0
ConfigItemCheck.ipp		61.9 %	13 / 21	87.5 %	7 / 8	37.5 %	6 / 16

Generated by: LCOV version 1.14

The coverage report is to be inspected and a desision is to be made.

If the possibility to improve coverage is given it is to be improved using one or more of the methods mentioned from point 4.5.1 to 4.5.4.

4.5.9.3 How to debug a fuzz target (with gdb or VSCode)

4.5.9.3.1 Debug process with no script

Fuzz-targets are executed in a docker container, therefore debuging with gdb or a gdbserverhas to happen within the docker container.

```
# Enter the docker container
e3sdk_scripts/shell.sh --docker --target fuzzing
```

```
# Run the target with gdb.
# NO_CIFUZZ is used to tell Cifuzz not to start fuzzing, instead it will give gdb
access to the fuzz-target.
NO_CIFUZZ=1 gdb build/bin/<fuzz_target_name> .cifuzz-findings/<name_of_the_finding>/
crashing-input
```

If a gdb-server setup exists it can be adjusted to be used.

4.5.9.3.2 Debug process with script

A gdb-server can be spawned inside the container using cifuzz.sh, this server will send all debugging information to *localhost:55555*

⚠️ Note that sometimes you have to adjust the hardcoded docker image name inside the cifuzz.sh script first.

Commands for debugging with VSCode and gdb-server

```
$ ./cifuzz.sh /debug <path/to/fuzz-target/bin/> <path/to/fuzz/input/>
# For example
$ ./cifuzz.sh /debug <build-dir>/bin/<fuzz_target_name> tests/fuzzing/
<fuzz_target_name>/<fuzz_target_finding>
```

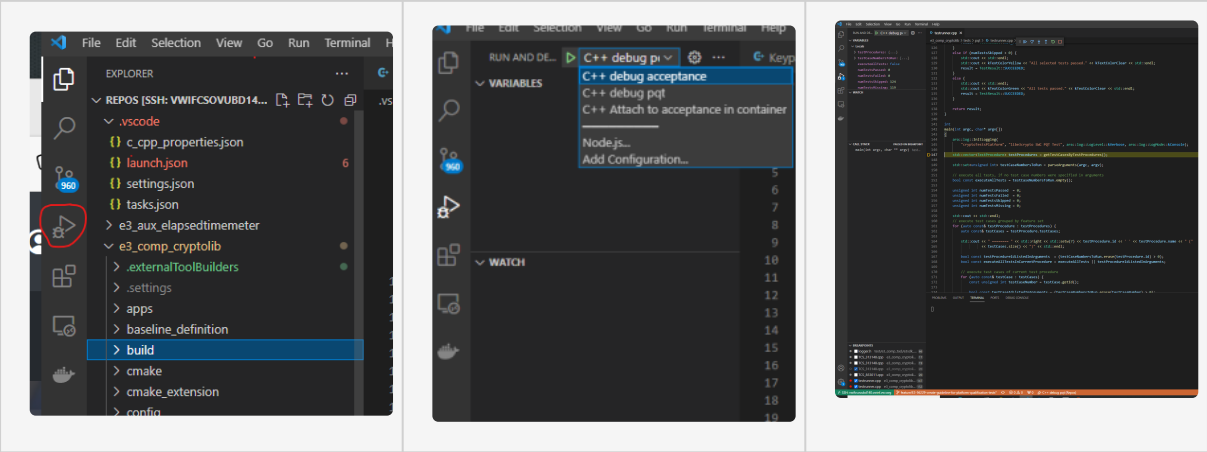
The following file is to be created under .vscode/ in the root repository to receive debugging information from the gdb-server:

launch.json

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "(gdb) Remote debug cifuzz",
            "type": "cppdbg",
            "request": "launch",
            "program": "/path to the binary executable e.g ${fileWorkspaceFolder}/
build/bin/fuzz_DiagComChannel",
            "args": [],
            "stopAtEntry": true,
            "cwd": "${fileDirname}",
            "environment": [],
            "externalConsole": false,
            "filterStderr": false,
            "filterStdout": false,
```

```
            "preLaunchTask": "",
            "miDebuggerServerAddress": "localhost:55555",
            "MIMode": "gdb",
            "setupCommands": [
                {
                    "description": "Enable pretty-printing for gdb",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                }
            ]
        }
    ]
}
```

After this, following options should be visible under debug configurations of VSCode, and the VSCode debugger interface should be usable.



4.5.9.4 Debug a finding

Findings Commands

```
$ ./cifuzz.sh findings                  # Display list of findings
$ ./cifuzz.sh findings <finding name>   # Display detailed information about a finding
```

After reading information about a finding using the commands above. It can be useful to reproduce the finding in debug mode.

Re-run finding commands

```
# Enter shell
$e3sdk_scripts/shell.sh --docker --target fuzzing

# Ensure that ubsan finding stops gdb but does not kill the process
$ export UBSAN_OPTIONS='halt_on_error=1:abort_on_error=1'

# Skip CI fuzz wrapper
$ export NO_CIFUZZ=1

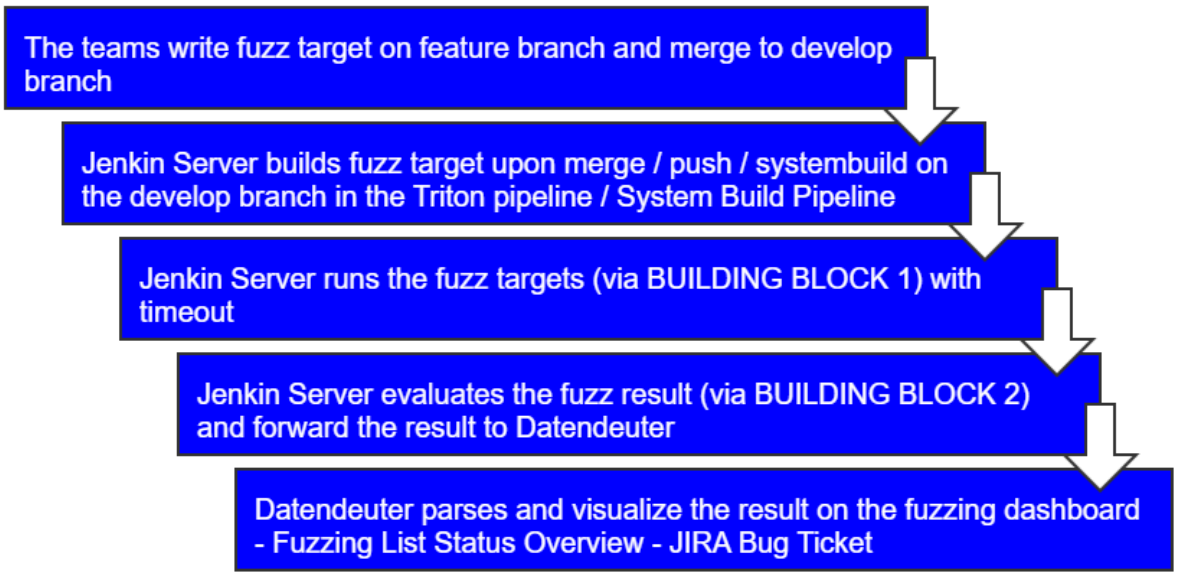
# Execute the binary with a specific input:
$ NO_CIFUZZ=1 <build-dir>/bin/<fuzz_target_name> .cifuzz-findings/predictive_barnacle/
<fuzz_target_finding>

# Start gbd and run finding
# Optionally the argument "-ex 'break LLVMFuzzerTestOneInput'" will set breakpoint at
the start of the fuzz target
$ gdb -ex run --args <build-dir>/bin/<fuzz_target_name> .cifuzz-findings/
hopeful_possum/<fuzz_target_finding>

# One-liner
$ UBSAN_OPTIONS='halt_on_error=1:abort_on_error=1' NO_CIFUZZ=1 gdb -ex 'break
LLVMFuzzerTestOneInput' -ex run --args <build-dir>/bin/<fuzz_target_name> $(find -type
 f -iname <fuzz_target_finding>)
```

4.6 Fuzzing on CICD pipeline

The following graph shows the work flow of fuzzing with the CICD Pipeline



1. First, the team writes their fuzz target, commits, pushes these to their repository

2. Now, there are 2 ways the CICD ppeline performs fuzzing:
   a. First option, upon merging the changes to develop or master branch, the TRITON CICD pipeline will build the fuzz targets' binaries and the coverage binaries and further required resources to start fuzzing.
   b. Second option, depending on how the System Build pipeline is configured, fuzzing build will be triggered either daily or weekly to run fuzzing. Please contact CICD team to get the current configuration state of the System Build Pipeline.
   Upon triggering, the System Build pipeline will build the fuzz targets' binaries and the coverage binaries and further required resources to start fuzzing.
3. After building the following required resources
   a) the different fuzzing binary files in (default) `{repo_root_dir}/build/bin/`
   b) the different fuzzing coverage binary in (default) `{repo_root_dir}/build-cov/bin/`
   c) the doxygen fuzzing documentation file in (default) `{repo_root_dir}/build/doc/doxygen/xml/e3fuzz.xml`
   the CICD pipeline will runs the fuzz targets via **BUILDING BLOCK 1: tools-bb-fuzzing-src** with a specific timeout set by the CICD team (please contact the CICD team to find out how long this timeout is currently set to).
4. After executing the **BUILDING BLOCK 1: tools-bb-fuzzing-src**, following result is generated:
   a) one *.lcov file for each fuzz target in (default) `{repo_root_dir}/tests/fuzzing/lcov/`
   b) findings documentation generated by cifuzz tool (default) in `{repo_root_dir}/.cifuzz-findings/`
   c) fuzzing result report file in (default) `{repo_root_dir}/fuzzing_result.json`
   the CICD pipeline will run the BUILDING BLOCK 2: tools-bb-fuzzing-report-src to analyze the fuzzing result, parses this and gather all information including fuzz target's code coverage, findings, stacktrace, finding's input to forward to Datendeuter to be visualized and reported.
5. After executing the **BUILDING BLOCK 2: tools-bb-fuzzing-report-src**, following result is generated:
   a) Fuzzing result zip file to be uploaded to artefactory (default) `{repo_root_dir}/fuzzing_report.tar.gz`. This zip file contains:
   - Input files for findings
   - Findings logs including stacktrace
   - `report.json` with all information about the fuzz run including code coverage, time executed, etc.

4.6.1 Building block 1: tools-bb-fuzzing-src

Following chapter gives details about the first building block. For further information, please refer to the documentation on Github.

Repository	https://git.hub.vwgroup.com/CARIAD/tools-bb-fuzzing-src

Functionality	**This building block takes a list of prebuilt fuzzing binaries and fuzzing coverage binaries and run the fuzz tests for a specific amount of time (specifiedable).** **Before running the fuzz tests, this building block makes sure to pull the corpus from the artefactory so it can continue fuzzing where it left off the last time. After executing the fuzz tests, this building block uploads the new corpus to the artefactory to save the progress. Also, this building block generates:** ▪ **LCOV coverage files for each fuzz tests** ▪ **Findings documentations** ▪ **Overall fuzzing run result** **For more detailed and up-to-date information, please check the detailed README documentation on the respective repository.**

4.6.2 Building block 2: tools-bb-fuzzing-report-src

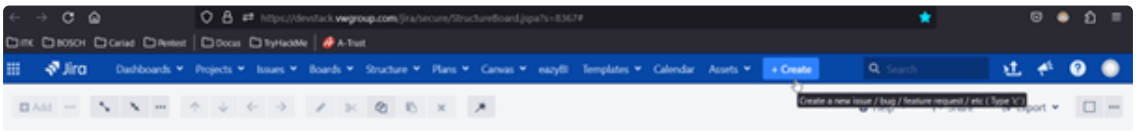
Repository	https://git.hub.vwgroup.com/CARIAD/tools-bb-fuzzing-report-src
Functionality	**This building block takes result from the first building block 1: tools-bb-fuzzing-src and create a report about the fuzzing effort for each fuzz target. This includes information such as:** ▪ **Findings, crashing inputs, CVSS Score and stack trace** ▪ **Code coverage** ▪ **Amount of time fuzzing was running** **This building block generates a zip file report to be uploaded to the artefactory. This zip file can then be parsed and visualized by Datendeuter. In case a finding is identified, Datendeuter would also have all the necessary information to automatically create a Jira bug ticket.** **For more detailed and up-to-date information, please check the detailed README documentation on the respective repository.**

4.7 Reporting fuzzing-findings / bugs

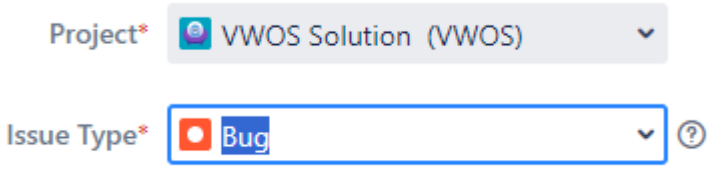
4.7.1 Manually creating a Jira bug ticket

4.7.1.1 VWOS Solution (VWOS)

1. Start the Ticket creation process in **Jira**.
   See relevant image



2. Select the **right Project** and the **Bug** issue type.
   See relevant image



3. Fill out the bug ticket with the following information. Make sure to fill out all the red star (*) fields.
   See the table of JIRA fields

	Jira Field	Expected values
General Details*	Project*	VWOS Solution (VWOS)
	Issue Type*	Bug
	Summary*	<repo_name> <bug tilte> in <crashed_function> e.g. e3_comp_tlrlib out of memory in vwg::e3p::tlr::createTLRBlobMessage

	Jira Field	Expected values

	Description*	**General information**
		Contact Person : <The person who identified this bug>
		DevTeam : <The team responsible for the repository at the time of fuzzing trigger (SBOM)>
		Used Component Version(s)
		<Repo name>
		<Fuzzing Branch>
		<Current commit ID from the Fuzzing Branch>
		Environment
		Compiled for x86_64, instrumented for fuzzing with cifuzz (libfuzzer).
		(or dependent on the CISence server environment. But this will stay the same for all fuzzing bug)
		Expected behaviour
		No <summary of the bug>
		e.g. No out of memory in vwg::e3p::tlr::createTLRUTF8StringMessage src/Messages/TLRMessageCreator.hpp
		Observed behaviour
		<summary of the bug> e.g., <out of memory in vwg::e3p::tlr::createTLRUTF8StringMessage src/Messages/TLRMessageCreator.hpp>
		Steps to reproduce
		Step to reproduce
		`// Clone the repository`
		`git clone <link to git repo>`
		`// Switch to our fuzzing branch`
		`git checkout <fuzzing branch>`
		`# TODO: change the profile fuzzing_x86_64 to a new one once all dependencies are built with clang`
		`// Build the fuzzing binary`
		`./build.sh --docker --target fuzzing --profile fuzzing_x86_64 --buildfolder=build --verbose --fuzzing -j2`
		`// Login to the Docker shell`
		`./cifuzz.sh /shell`
		`// Run the binary with the crashing input`
		`NO_CIFUZZ=1 build/bin/<fuzz target binary> <path to crashing-input>`
		`// View the finding via`

	Jira Field	Expected values
| | | ```
./cifuzz.sh finding <name of finding>

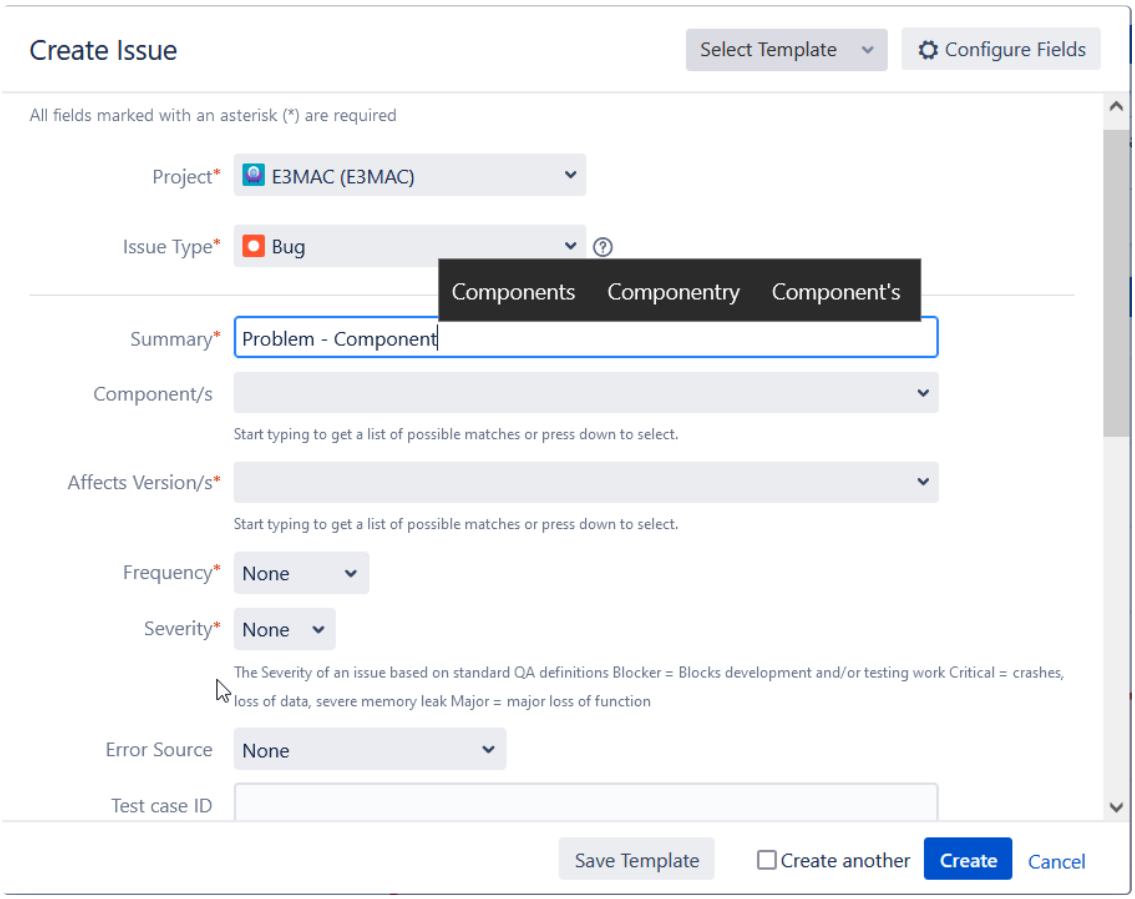
// Launch a GDB Debug server
./cifuzz.sh /debug build/bin/fuzz_DiagComChannel

// Follow these steps to debug with VSCode
https://devstack.vwgroup.com/confluence/display/E3SEC/CI-Fuzz+Proof+of+Concept#CIFuzzProofofConcept-Debugprocesswithscript
```<br><br>**Healing**<br>n/A, what the step to do to bring it to run.<br>**Customers Impact**<br>n/A<br>**Traces**<br><Attach logfile><br><Attach crash input file> |
	Priority	<Depends on the CVSS score for severity of the bug>. As rule of thumb: • Critical 9.0-10.0 => Highest • High 7.0 -8.9 => High • Medium 4.0 - 6.9 => Low • Low 0.1 - 3.9 => Lowest
	Severity *	<Depends on the CVSS score for severity of the bug>. As rule of thumb: • Manually set if related to safety => Blocker • 9.0-10.0 => Critical • 7.0 -8.9 => Major • 4.0 - 6.9 => Minor • 0.1 - 3.9 => Trivial • 0.0 => None
	Assignee	Manually set by the team after analysis of the bug
	Affected Version /s*	Current PI
	Attachment	<Attach logfile> <Attach crash input file>
```

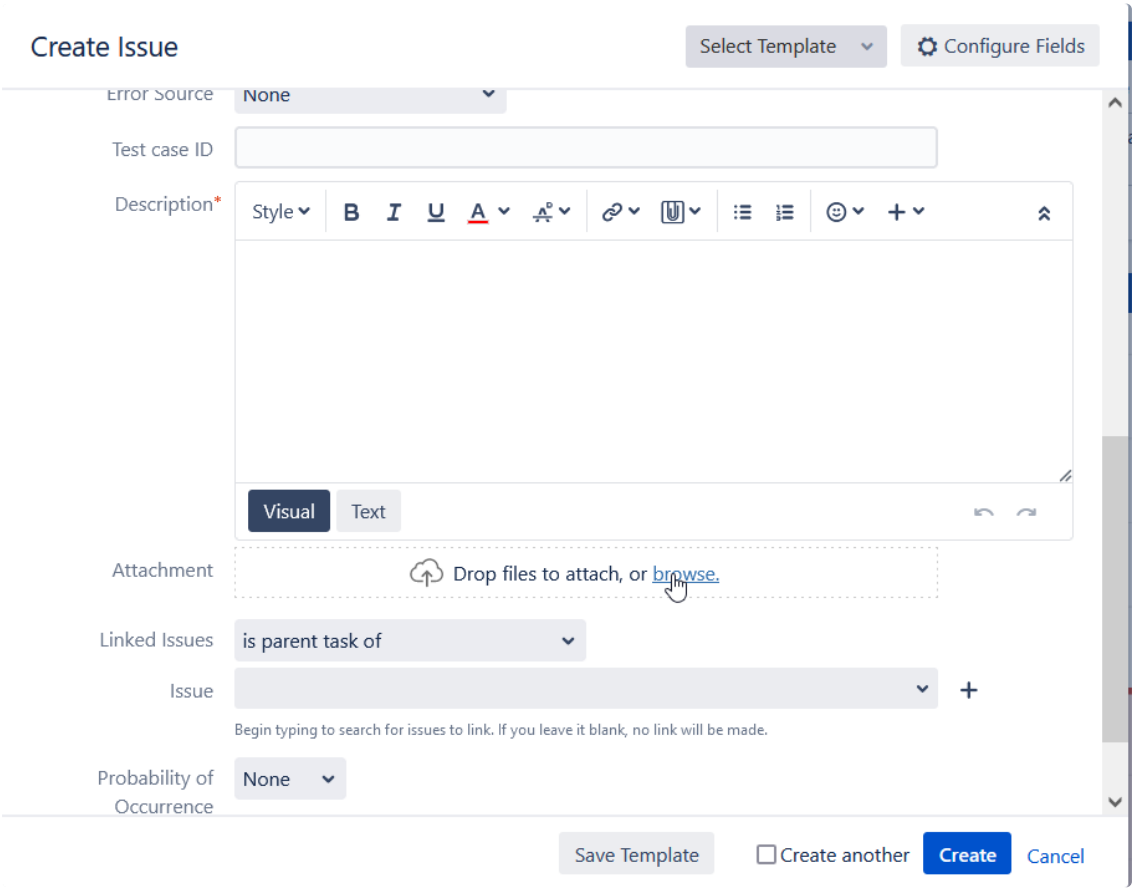
	Jira Field	Expected values
	Teams*	The team responsible for the repository at the time of fuzzing trigger (SBOM)
	Master Branch*	Choose the correct cluster. Mostly Cluster 6 if it is develop branch
Impact Analysis*	Affected Teams	The team that will have to fix the bug.
	Error Source*	Fuzzing
	Test case ID*	Fuzzing
	Security Relevant	Yes
	Scope	Choose the scope
	Probability of Occurence*	always

4.7.1.2 E3MAC

1. Start the Ticket creation process in **Jira**. Select the **right Project** and the **Bug** issue type. In the summary field, give a name to the bug. It is sufitient to write **what** is wrong and **where** as a name.
   See relevant image



2. Fill in all **red star fields**. In the Description box fill in as much information as you can/have info on. Using the Attachments field, **attach any files that help replicate the bug**.
   See relevant image



4.7.2 Via CICD

- Via CICD Dang, Quang Anh (EXTERN: ITK)
  Jenkins monitors and report
  Automatic JIRA bug ticket creation

5 Practical Examples

In this section, we present a list of current examples that we have in the course of onboarding different teams to fuzzing:

POC Use Case	Team	Repo & Branch	Link to fuzz target	Build process
Generic new build process	UDSM	e3_app_CSS:custom/cifuzz[90]	fuzz_BucketHandler.cpp[91]	new
Generic new build process	TLR	e3_comp_tlrlib:custom/cifuzz_cmake_new_buildprocess[92]	fuzz_TLRMessage.cpp[93]	new
File fuzzing	TMBase	e3_tm_base:Fcustom/fuzzing-setup[94]	fuzz_Time_Diff_Calculation.cpp[95]	
Wrapper		TODO		
Gtest		TODO		

90 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_css/browse?at=refs%2Fheads%2Fcustom%2Fcifuzz
91 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_css/browse/tests/fuzzing/fuzz_BucketHandler.cpp?at=refs%2Fheads%2Fcustom%2Fcifuzz
92 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_tlrlib/browse?at=custom%2Fcifuzz_cmake_new_buildprocess
93 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_tlrlib/browse/tests/fuzzing/fuzz_TLRMessage.cpp?at=refs%2Fheads%2Fcustom%2Fcifuzz_cmake_new_buildprocess
94 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_tm_base/browse?at=refs%2Fheads%2Fcustom%2Ffuzzing-setup
95 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_tm_base/browse/tests/fuzzing/fuzz_Time_Diff_Calculation.cpp?at=refs%2Fheads%2Fcustom%2Ffuzzing-setup

6 Tips, Tricks and Best Practices

How to generate coverage information (LCOV) and show them in Visual Studio Code (VSCode) with Coverage Gutters?

1. Install the Coverage Gutters[96] extension in VScode
2. Build and generate coverage file in the project root folder

Generate coverage LCOV file

```
# TODO: change the profile fuzzing_x86_64 to a new one once all dependencies are
built with clang
./build.sh --docker --target fuzzing --coverage --profile fuzzing_x86_64 --
buildfolder=build-cov --verbose --fuzzing -j2
./cifuzz.sh coverage --add-corpus .cifuzz-corpus/build/bin/<name of fuzz target>/
build-cov/bin/<name of fuzz target> --output=lcov.info --format=lcov
```

3. Go to the source file we want to display the coverage information and click "Coverage Gutters: Watch"

4. The coverage information is then displayed in colors.

How to manipulate #ifdefine block to achieve the suitable testing scenario similar to unit test?

In the code #ifndefine #if defined(...) #if UNIT_TEST=1 etc. are often used to manipulate code in different scenarios such as unit test and production code. Since our fuzz test setup is similar to a unit test setting, we often want to reuse mocks or have attributes of classes public for access from outside. To do so, you can use:

Code flow manipulation with #define

```
#define UNIT_TEST=1

...

#undef UNIT_TEST
```

96 https://marketplace.visualstudio.com/items?itemName=ryanluker.vscode-coverage-gutters

to manipulate the code flow.

7 Download

- Our latest cifuzz.sh script[97]

97 https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3-aux-security-repo/browse/DAST/cifuzz?at=refs%2Fheads%2Fdevelop

8 Abbreviation & Terminology

9 References

Reference	Document
1	https://www.microsoft.com/en-us/securityengineering/sdl/practices
2	https://www.wired.com/2016/06/hacker-lexicon-fuzzing
3	https://kunnamon.io/tbone/tbone-v1.0-redacted.pdf
4	https://[98]en.wikipedia.org/wiki/Fuzzing[99]
5	https://llvm.org/docs/LibFuzzer.html#value-profile
6	https://github.com/CodeIntelligenceTesting/tutorials/blob/workshop/Dictionary/dictionary_example.cpp
7	https://github.com/CodeIntelligenceTesting/tutorials/tree/workshop/Dictionary
8	https://llvm.org/docs/LibFuzzer.html#dictionaries
9	https://github.com/CodeIntelligenceTesting/tutorials/tree/workshop/CustomMutators
10	https://mboehme.github.io/paper/IEEESoftware20.pdf
11	https://github.com/google/fuzzing/blob/master/docs/split-inputs.md
12	https://github.com/google/fuzzing/blob/master/docs/good-fuzz-target.md[100]
13	https://github.com/google/fuzzing/blob/master/docs/why-fuzz.md[101]
14	https://google.github.io/googletest/gmock_cook_book.html#mocking-free-functions[102].

98 https://en.wikipedia.org/wiki/Fuzzing
99 https://en.wikipedia.org/wiki/Fuzzing
100 https://github.com/google/fuzzing/blob/master/docs/why-fuzz.md
101 https://google.github.io/googletest/gmock_cook_book.html#mocking-free-functions
102 https://github.com/zardus/preeny#de-socketing

Reference	Document
15	https://github.com/zardus/preeny#de-socketing
16	https://pubs.opengroup.org/onlinepubs/9699919799/functions/fmemopen.html

10 TODO

Old doc file:

https://volkswagengroup.sharepoint.com/:w:/r/sites/E3SW-PAC-Security-TrainingTest/Shared%20Documents/General/Fuzzing/CI-Fuzz_PoC--Arbeitsstand/20210702_VWOS-CI-Fuzz-ProofOfConcept_HSAT_V08F.docx?d=w0746cb68fe4442dd86593715d55dcf36&csf=1&web=1&e=flRkRx

TODO:

- Eventually migrate to XC.4 Sec Management/3 Guidelines | Security management
- Finish by end of this PI 23 Jun 2023

11 (Deprecated) Archive

This page contains deprecated information which is not suitable for the current state of fuzzing. However, some of these information might still be relevant in the future.

11.1 Working with CI-Sense Server

When working on fuzzing a project it is not realistic to continuously execute all test locally. For this reason, a server is in the process of being set up.

11.1.1 Server Information

Server address: https://cifuzz.qs.hub.vwgroup.com

Status: **IN PROGRESS** (But usable)

This server is currently under construction. It is possible to run a server locally. For information on how to please contact Code-Intelligence to get the server bundle and follow instructions here https://docs.code-intelligence.com/ci-fuzz-installation to install the server.

11.1.2 Creating and uploading a bundle to the server

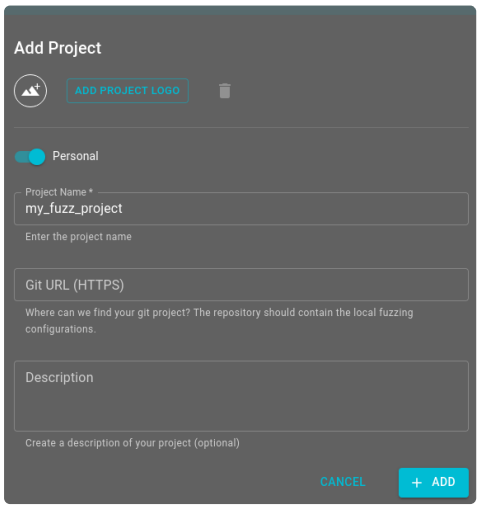
TODO: Dang, Quang Anh (EXTERN: ITK) Update the information here. Copy both binaries for normal libfuzzer and coverage binary to the same directory, then cifuzz bundle

Creating fuzz-bundle with cifuzz

```
# -v (optional) see verbose output for more information in case bundle creation fails.
# --build-jobs=$(nproc) (optional) multithreading to speed up build process.

./cifuzz.sh bundle <fuzz_target_name> -o <fuzz_target_name>_bundle.tar.gz --build-jobs=$(nproc) -v
```

The command above will create a tar.gz file containing dependencies and 2 binaries (one to execute the fuzz test and the other one to provide code coverage information).

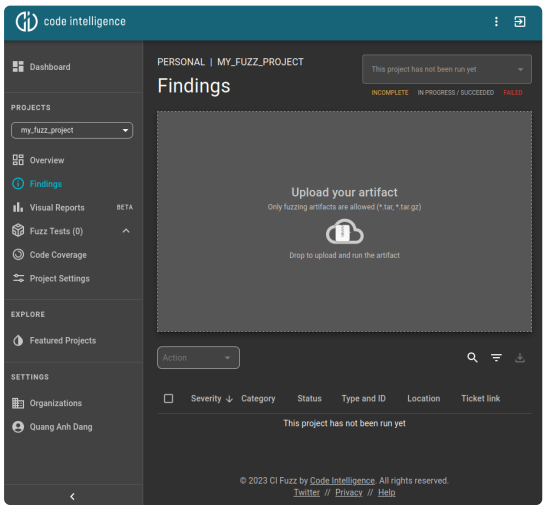
After creating a bundle, proceed on to the server and create a new project. On the main page, go to "Select Project" and choose "Add project". Then, provide project name and project GIT Repo URL and description if needed:



After creating a fuzz project, there are 3 ways to upload a bundle to the server:

11.1.2.1 Drag and drop

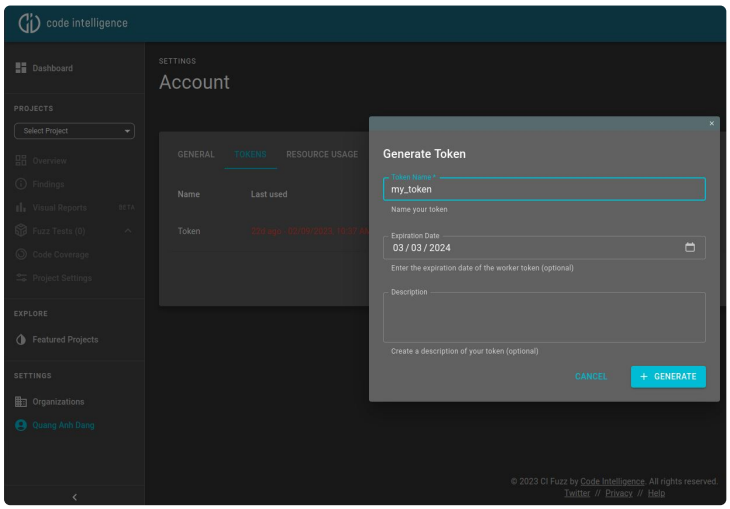
On the server page, choose a project on the left side panel and drag and drop the <fuzz_target_name>_bundle.tar.gz file previously generated on to the project area.



After wait for the upload to finish, the fuzz-test will start by itself.

11.1.2.2 Cifuzz remote-run

A bundle can be uploaded to the server via command-line. This requires an access token. To generate a token, on the server webpage at the bottom of the left panel, click on your name, "TOKENS" and "GENERATE TOKEN". Note down the token, this cannot be accessed again.



After a token is created the following command can be used to upload a fuzz-bundle

Upload the bundle to cifuzz server via cifuzz remote-run

```
$ ./cifuzz.sh remote-run --bundle <fuzz_target_name>_bundle.tar.gz --server
<server_link> --project <project_name>
You need to authenticate to a CI Fuzz Server instance to use this command.
Log in now? [Y/n]: Y
You need an API access token which can be generated here:
...
Open browser to generate a new token? [Y/n]: n
Paste your access token:
✓ Successfully authenticated with ...
Your API access token is stored in ...
Uploading...
Upload complete
✓ Successfully started fuzzing run. To view findings and coverage, open:
  ...
```

Info

The <server_link> passed to the --server parameter must not contain a trailing backslash '/'.

Info

The project name can be found in the URL when accessing the project on the cifuzz-server webpage

11.1.3 CICTL tool

CICTL provides an useful API to access and interact with the cifuzz-server. It is useful for automation of the fuzzing process via Jenkins or Github. Naturally, the fuzz-target can be uploaded to the server with this tool.

Login to the docker container

```
# The container has the cictl tool prepared.
e3sdk_scripts/shell.sh --docker --target fuzzing
```

the following script can be used as an abstraction layer for the CICTL tool.

Steps to upload our fuzz target to the server via cictl tool

```
#!/bin/bash

set -ex

# Minimum severity for findings that causes the pipeline to fail.
# Findings with lower severity are still reported but do not fail
# the pipeline.
# Possible values: 'LOW', 'MEDIUM', 'HIGH', 'CRITICAL'
MIN_FINDINGS_SEVERITY=MEDIUM

# Define some parameters. Note that the server address should not contain https:// nor
a backslash in the end
FUZZING_SERVER=<server_link>
CICTL_CMD="cictl --server ${FUZZING_SERVER}"
BUNDLE_LOCATION=./<fuzz_target_name>_bundle.tar.gz
PROJECT=projects/<project_name>
WEB_APP_ADDRESS=<web_app_URL>
TIMEOUT=300
CI_FUZZ_API_TOKEN=[[Insert your Access TOKEN here]]

# Then, login to the server
echo "${CI_FUZZ_API_TOKEN}" | ${CICTL_CMD} login --quiet

# Then, upload the bundle to the server
ARTIFACT_NAME=$(${CICTL_CMD} import artifact ${BUNDLE_LOCATION} --project-name "$
{PROJECT}")

# Finally, start campaign run via
CAMPAIGN_RUN_NAME=$(${CICTL_CMD} start "${ARTIFACT_NAME}")

# Observe Campaign Run with timeout, watching for findings
${CICTL_CMD} monitor_campaign_run \
  --dashboard_address="${WEB_APP_ADDRESS}" \
  --duration="${TIMEOUT}" \
```

```
  --min_findings_severity="${MIN_FINDINGS_SEVERITY}" \
  ${CAMPAIGN_RUN_NAME}
```

TODO:

- Pre-CICD Manually
- Post-CICD Automatically
  Process e.g commit, Jenkins builds, jenkin uploads and monitor
- Do a remote run on CICD server
- Extracting code coverage information
- Start/Stop fuzz campaign

12 (Deprecated) CI Fuzz Integration/Training

Please visit our new updated tutorial page: CI-Fuzz Proof of Concept (see page 22)

12.1 Agenda, Purpose and Training Materials

12.1.1 DAST in this case Fuzzing on Architecture level (Draft)

DAST (Fuzzing) is intended to find faulty behavior of a component based on malformed input. By randomizing the input DAST triggers arbitrary states in the evaluation of input by a component and thus can expose problems like out-of-bounds access or overflows.

However, certain bugs can only be found, if the input is only partially random. Example:

- A component takes a URL for accessing a server resource. An implementation of DAST now could provide arbitrary byte arrays as input to check the components behavior.
- Because of a simple check implemented by the programmer, most of the input is actually discarded, because the input does not start with https://. (see page 114)
- They might be however problems with URL-like string because escape characters and similar special characters:
  A test-pattern might look like https://[]<random ascii characters>/ (see page 114)**+** --> Thus problems like interpretation of "%s" or "\n" within the component might be found which were not found by completely randomizing the input because there was a simple check implemented.

DAST Questionnaire by steps

- First step :
  - Looking at the components inside the platform generate a picture here and list components see stack picture flow of interaction
  - SWAD Data from outside
  - Where will be the data inside terminated
  - Is it possible to look for services who have the need of using root right
  - If an SRA on ICP level exist to look for that once with lower effort

  Second step
  - e.g. CS Proxy as gate to influence the rest (http handling) see ticket ~~E3-78634~~
  - e.g. LUM see comment see what happens in the CS proxy and transfer to LUM behavior
  - e.g. open source libraries and their usage inside the platform, look up if they are fuzz tested already before, CVE, trust level of the
  - zip file access handling, botan .

  Third step

  How to make a concept out of it this for other use cases?

  Make a decision flow diagram with question to give a guide : flow points as guideline
  - external interface

- internal were are get the data's terminated
- dynamics data action

TODOÄnderungen anzeigen[103]

- Fuzzing presentation video

12.2 Introduction to Fuzzing

12.2.1 What is fuzzing

Wikipedia definition
"… fuzzing or fuzz testing is an automated software testing technique that involves providing invalid, unexpected, or random data as inputs to a computer program.

The program is then monitored for exceptions such as crashes, failing built-in code assertions, or potential memory leaks."

Source: https://[104]en.wikipedia.org/wiki/Fuzzing[105]

Security Goal

Minimize risk that software contains exploitable vulnerability.

UNECE / ISO21434

Does not require fuzz testing, but suggests it as a method for searching for vulnerabilities along with pen testing.

Fuzzing in the secure software development life-cycle (SSDL):

- use available interfaces to inject tainted data
- find vulnerabilities: crashes, memory, timing
- monitor software under test
- collect & report issues
- ensure reproducibility

103 https://devstack.vwgroup.com/confluence/pages/resumedraft.action?draftId=860960837&draftShareId=89eefcd5-00eb-4908-b4d3-dfb20509d24b
104 https://en.wikipedia.org/wiki/Fuzzing
105 https://en.wikipedia.org/wiki/Fuzzing

12.2.2 Motivation

„Today, if you're a hacker trying to crack a system, the first thing you do is fuzz test it.“
Barton Miller, https://www.wired.com/2016/06/hacker-lexicon-fuzzing

Attackers use fuzzing to develop zero-day-exploits!

TBONE - A zero-click exploit for Tesla MCUs

https://kunnamon.io/tbone/tbone-v1.0-redacted.pdf

Attackers steps:

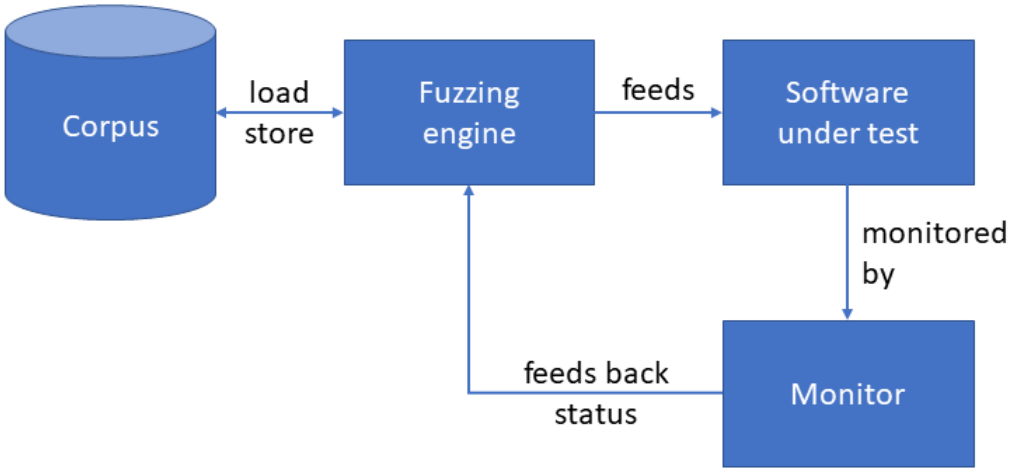
- Reviewed code.
- Found bad smells in ConnMan component
- Fuzzed ConnMan
- Found vulnerability.
- Exploited vulnerability.

Additionally to test techniques, like unit testing, static analysis, component & integration testing, negative testing is used.
Fuzzing extends negative testing by generating test vectors and can therefor reduce the risk that a component contains a vulnerability.

12.2.3 Overview Fuzzing

Modern fuzzing techniques use guided mutations to discover unique software paths based on information gathered by source code instrumentation.

A fuzzing setup consists of four protagonists connected in a closed loop. Their responsibilities and some of their attributes are summarized in the following image:



General setup of feedback-based fuzzing

Corpus

- Stores input vectors
- Grows over time as progress is made
- In coverage-guided / feedback-based fuzzing, each program state reached leads to at least one entry in the corpus
- Requires
  - Persistence
  - Maintenance, like deduplication

Fuzzing engine

- Reads from corpus
- Mutates promising data sets based on statistics and prediction algorithms
- Persists data sets reaching new program state into corpus
- Controls Software under test, e.g. spawns fresh instance
- Feeds new input into Software under test
- Processes the feedback from the monitor / instrumentation, e.g. byte comparisons, to be able to predict good new inputs

SW under test

- The unit under test: a function, module, component or system.
- Processes given input
- Optionally instrumented to provide best information to monitor

Monitor

- Observes status of software under test
- Feeds back this status to the Fuzzing engine

12.2.4 Introduction CI Fuzz

CI Fuzz is a fuzzing middle-ware by the company Code Intelligence[106], Bonn.

The tool's purpose is to simplify the integration of fuzz testing into the developer's workflow. A component developer shall be enabled to create and maintain fuzz tests as easy as writing unit tests.

The tools consists of two applications: a client-side and a server side installation

12.2.4.1 Command line interface tools (CI Fuzz CLI)

Documentation: CI Fuzz CLI - Code Intelligence Docs (code-intelligence.com)[107]

These are a collection of command line tools bundled in a single executable which help during typical development tasks, e.g.

- setup of a new test
- building and fuzzing a target
- building and coverage reporting of a target
- bundling and deploying a target to the fuzzing server

Currently the CI Fuzz CLI produces two different targets:

- one target for in-process fuzzing
  - contains the fuzzer (clang libfuzzer)
  - instrumented for fuzzing to provide feedback
  - instrumented with sanitizers to detect vulnerabilities:
    ASAN - address sanitizer
    UBSAN - undefined behavior sanitizer
  - it is not possible to use different fuzzing engine or change sanitizer
- one target for coverage generation
  - line, function & branch coverage
  - not required for fuzzing
  - used for reporting and debugging

12.2.4.2 CI Fuzz

Documentation: CI Fuzz - Code Intelligence Docs (code-intelligence.com)[108]

12.2.4.2.1 Continuous integration builder (CI-BUILD & CICTRL)

The CI/CD integration provides ...

- building of fuzz tests for fuzzing and coverage reporting

106 https://www.code-intelligence.com/
107 https://docs.code-intelligence.com/cifuzz-cli
108 https://docs.code-intelligence.com/ci-fuzz

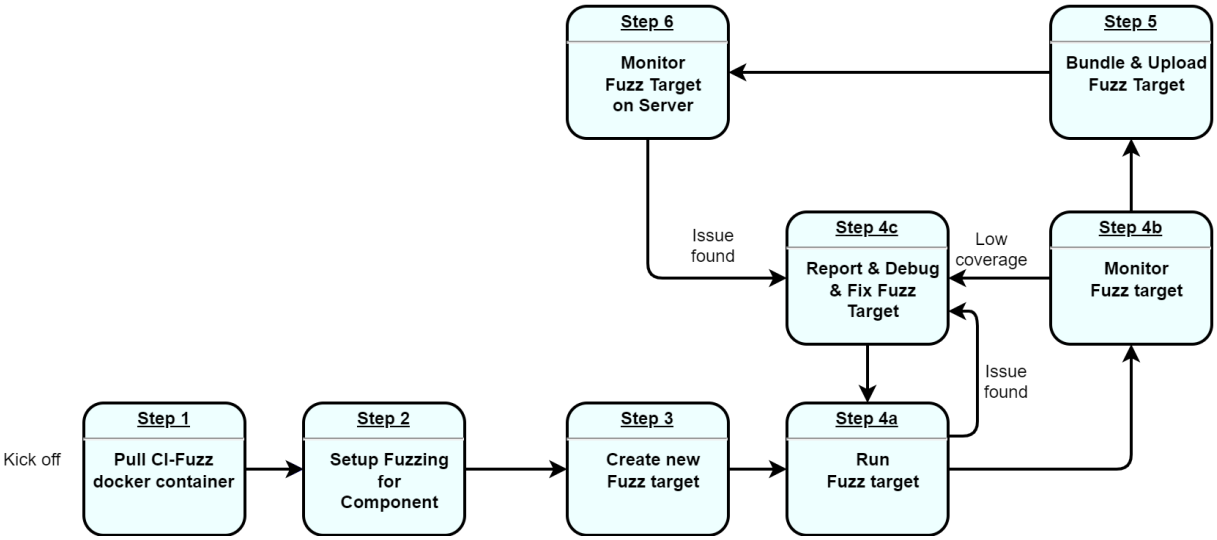
- deploying to and triggering of Fuzzing Server
- collecting reports and findings from Fuzzing Server
- reporting pipeline state to CI/CD

12.2.4.2.2 Fuzzing Server

The fuzzing server orchestrates the fuzzing runs:

- execute fuzzing and coverage targets on trigger
- report findings to CI/CD server
- report findings and coverage to web frontend

12.2.5 Fuzzing Workflow



Phase	Step	Description	Output
Initial setup Container	1	Create CI Fuzz enabled build container from your component's build container. If the component's build container is updated, this step needs to be redone. This step will become obsolete once fuzz enabled containers are provided by SDK.	CI Fuzz enabled Docker container available

Phase	Step	Description	Output
Initial setup Project	2	Adapt component's makefiles to enable fuzzing.	Component is on-boarded for fuzzing
Fuzz target Development	3	Select functionality to fuzz test. Scope: function, class or sub-component Trigger: single API, multiple APIs State: stateful or stateless? Setup preconditions Craft input vector for each call: static, fully based on fuzz input or partial	Fuzz target can be built
Fuzz target Maintenance	4a	Fuzz target is running until ... • no new edges are found (→ 4b) • time limit is reached (→ 4b) • one or multiple issue are found (→ 4c)	Corpus data, if some edges were found. Fuzzing input for issue
	4b	Replay corpus data to create coverage information Coverage information can be used to identify blockers and fine-tune the fuzz target setup.	Coverage report available
	4c	Replay and analyze finding Remove blockers Fix "unit under test" or fuzz target.	Patched vulnerability (prod. code) or fuzz target (test code)
Monitoring	5	Create self contained bundle of executables and dependencies to run on server. Upload to server project Schedule server runs	Fuzz target running on server
	6	Check how fuzz target performs over time	

12.3 Dockerized CI Fuzz

After following this guide, you will have cifuzz running in a docker container.

Using this container enables you to ...

1. Run fuzzing tests with ASAN/UBSAN support
2. Create code coverage reports to evaluate the fuzzing progress
3. Display and debug findings

Before starting the process, you should download our cifuzz.sh[109] script and put it in the root of your project repository. Alternatively, you can also download the latest version of the cifuzz.sh script[110] from our Security Team Repository.

12.3.1 Pulling the cifuzz docker image from artefactory

Execute following command to pull the docker cifuzz image from artefactory. Please replace the [version] with the oldest version in the change log on our docker image repository[111] e.g 1.0.0. You can see the current build status of the docker images on the jenkins server[112].

```
docker pull e3-e3sp-dockerdev.docker.devstack.vwgroup.com/master/
e3_docker_ci_fuzz:1.0.2
```

After pulling the image, we need to change the name of the image to cifuzz:latest:

```
docker tag  e3-e3sp-dockerdev.docker.devstack.vwgroup.com/master/
e3_docker_ci_fuzz:1.0.2 cifuzz:latest
```

After setting up the docker image, you should be able to execute

```
./cifuzz.sh [arguments]
```

This shell script will forward all arguments (except for some specific options) to the docker container and execute the followings inside the docker container.

```
cifuzz [arguments]
```

As an example, the result of "./cifuzz.sh help" should be:

109 https://devstack.vwgroup.com/confluence/download/attachments/860960818/cifuzz.sh?api=v2&modificationDate=1681302056738&version=1
110 https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3-aux-security-repo/browse/DAST/cifuzz?at=refs%2Fheads%2Fdevelop
111 https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_docker_ci_fuzz/browse/changelog.md
112 https://ioi-toolchain.vwgroup.com/jenkins/job/E3SP_SDK/job/Dockers/job/e3_docker_ci_fuzz/job/hotfix%252Ffix-master/1/

```
./cifuzz.sh help
Usage:
  cifuzz [command]

Available Commands:
  init        Set up a project for use with cifuzz
  create      Create a new fuzz test
  run         Build and run a fuzz test
  reload      Reload fuzz test metadata
  bundle      Bundles fuzz tests into an archive
  coverage    Generate a coverage report for a fuzz test
  finding     List and show findings
  integrate   Add integrations for the following tools: Git, CMake, VS Code
  help        Help about any command
  completion  Generate the autocompletion script for the specified shell

Flags:
  -C, --directory string   Change the directory before performing any operations
      --help               Show help for command
  -v, --verbose            Show more verbose output, can be helpful for debugging
problems

Use "cifuzz [command] --help" for more information about a command.
```

In addition to that, the script itself provides additional functionality:

Print the release tag & git hash of the cifuzz tool

```
./cifuzz.sh /version
v0.4.0-0-g77405c4

# Please state this version if you require support
```

Enter the cifuzz enabled container. Useful for debugging

```
./cifuzz.sh /shell
e3-user@d9cd314ac596:/home/developer/cifuzz-0.4.0-clang15$
```

To know about further options, you can execute help.

```
$ ./cifuzz.sh /help
Usage: ./cifuzz.sh [options]
Options:
```

```
  run [fuzztarget name]                                      Run fuzzing on the fuzz
target defined inside fuzzing directory without .cpp ending
  /shell                                                     Login to the docker
container with a shell session. Use for debugging
  /debug [path to fuzztarget bin] [path to fuzz input]       Start a gdbserver inside
the docker container to debug with VSCode at localhost:55555
  /version                                                   Show the version of cifuzz
being used
  /help                                                      Print this help message
  *                                                          Pipe all arguments in ./
cifuzz.sh [arguments] and execute "cifuzz [args]" inside the docker container
```

12.3.2 Prepare your component folder for cifuzz

1. Copy `cifuzz.sh` into your component folder
2. Create a cifuzz.yaml file inside your component folder:

   ```
   ./cifuzz.sh init
   ```

3. Edit `cifuzz.yaml` (usually just uncomment the lines):
   a. Set `build-system` to `cmake`.
   b. Set `use-sandbox` to `false`.
4. Edit your top-level `CMakeLists.txt`:
   a. The CI Fuzz cmake integration requires a minimum cmake version 3.16.
      Please ensure that the top of the file specifies:

      ```
      cmake_minimum_required(VERSION 3.16)
      ```

   b. Add the following lines before any target definition / subdirectory import:

      ```
      if(DEFINED ENV{CIFUZZ})
          find_package(cifuzz)
          enable_fuzz_testing()

          file(REMOVE_RECURSE /home/e3-user/.conan/profiles)
          execute_process(COMMAND "conan" config install /opt/e3sdk/conan_settings)
          execute_process(COMMAND "conan" install "${CMAKE_SOURCE_DIR}" --profile
      fuzzing-x86_64)
          file(TOUCH /home/e3-user/.conan/profiles/default)

          set(CONAN_DISABLE_CHECK_COMPILER ON)
          list(APPEND CMAKE_MODULE_PATH ${CMAKE_BINARY_DIR})
      ```

```
    # Enforce DWARF V4 debug symbols to enable debugging support with gdb
    # See: https://github.com/llvm/llvm-project/issues/53439
    add_compile_options(-gdwarf-4)
endif()
```

This block will on be active if an environment variable `CIFUZZ` is defined, so it should not change your usual build environment. If you run cifuzz, though, this environment variable is set automatically.

c. Add a subdirectory `fuzzing`:

```
if(DEFINED ENV{CIFUZZ})
    add_subdirectory(fuzzing)
endif()
```

5. Create a subdirectory for the fuzzing tests:

```
mkdir fuzzing
```

6. Create a fuzz test stub:

```
# cpp language selection
# -C, --directory string   Change the directory before performing any operations
# -o, --output string      File path of new fuzz test

./cifuzz.sh create cpp -C fuzzing -o my_fuzz_test_1.cpp
```

7. Create a `CMakeLists.txt` inside of `fuzzing/`.

Example: fuzzing/CMakeLists.txt

```
cmake_minimum_required(VERSION 3.16)
set(LINK_LIBS
    ${TARGET_NAME}_lib
    generatedlib
)

add_fuzz_test(my_fuzz_test_1 my_fuzz_test_1.cpp)
target_link_libraries(my_fuzz_test_1 LINK_PUBLIC ${LINK_LIBS})
```

12.3.3 Implement a fuzzing test in the stub file

Let's take a closer look at the generated fuzz test stub:

Click to expand fuzzing/my_fuzz_test1.cpp

fuzzing/my_fuzz_test1.cpp

```cpp
#include <assert.h>

#include <cifuzz/cifuzz.h>
#include <fuzzer/FuzzedDataProvider.h>

FUZZ_TEST_SETUP() {
  // Perform any one-time setup required by the FUZZ_TEST function.
}

FUZZ_TEST(const uint8_t *data, size_t size) {
  // FuzzedDataProvider provides convenience methods that turn the raw fuzzer
  // data into common types. Use it to generate parameters for the function you
  // want to fuzz:
  //
  // FuzzedDataProvider fuzzed_data(data, size);
  // int my_int = fuzzed_data.ConsumeIntegral<int8_t>();
  // std::string my_string = fuzzed_data.ConsumeRandomLengthString();

  // Call the functions you want to test with the provided data and optionally
  // assert that the results are as expected:
  //
  // int res = DoSomething(my_int, my_string);
  // assert(res != -1);

  // If you want to know more about writing fuzz tests you can checkout the
  // example projects at https://github.com/CodeIntelligenceTesting/cifuzz/tree/main/examples
  // or have a look at our tutorial:
  // https://github.com/CodeIntelligenceTesting/cifuzz/blob/main/docs/How-To-Write-A-Fuzz-Test.md
}
```

The skeleton defines two functions with documentation which makes them self-explaining.

`FUZZ_TEST_SETUP()`
... is just called once per run. Use it to initialize static content.

`FUZZ_TEST()`
... is called multiple times and is supplied with the generated fuzz data. Generally the fuzzer starts easy with short inputs, which it gradually increased up to the defined upper limit (default: 4kB).

Depending on the specification of your code under test it may be required to capture exceptions here.

As this function may be called millions of time during one run, take care not to leak any resources during setup, execution and tear down.

12.3.4 Run you first fuzz test

```
./cifuzz.sh run my_fuzz_test_1 -v
```

⚠️ It is recommended to run with -v verbose option to see the progress of the build and also receive the prompt for the credentials for the artifactory.

12.4 Fuzzing your E^3 project with CI Fuzz

12.4.1 Creation of a fuzz target

12.4.1.1 Selection

You cannot fuzz each and every function of your component as this will not scale well. Instead focus on the code sections which process untrusted (aka tainted) data. Untrusted data is data received across a trust boundary, e.g. user supplied input, or data retrieved from the file-system or via network socket.

So first of all you need to figure out which functionality is exposed to taint. These sections must then be hardened by fuzzing.

If your component is a library, consider fuzzing all of its public interface.

Identifying input

Data/input received and processed by your component may be network sockets, inter-process-communication, data from the file system, external interfaces, command line arguments, system environment variables etc.

When in doubt ask your system security architect about trust boundaries to your component.

Identifying structure

What kind of data does your component consume?

- Highly structured data, e.g. text file containing JSON or XML, binary data like ARA com serialized content
- Complex objects

- simple input

Identifying state

Most component contain state which cannot be reached by a single call.

Thus multiple calls of single or multiple calls will be required for a single run.

For further information please refer to https://github.com/google/fuzzing/blob/master/docs/structure-aware-fuzzing.md#fuzzing-stateful-apis

12.4.1.2 Setup / Teardown

Setting up your function block for fuzz testing is similar to setting it up for unit testing:

- preconditions must be setup
- the component must be in defined state
- you need a path to feed test data into your component / unit under test

If you require resource allocations to setup these preconditions, ensure that they are de-allocated on function exit as the fuzz test function may be called millions of times during one fuzz run.

12.4.1.3 Handling input

12.4.1.3.1 Simple Function parameters

The probably most common way of giving input to the "code under test" are function parameters. The fuzzing engine will give us a char-array `data` of size `size`. Sometimes this is also the expected type of the code under test. If it's not, we have to do a conversion. For this task, cifuzz comes with a class named FuzzedDataProvider. It is initialized with data and size and provides methods to extract a set of different data types:

Examples FuzzedDataProvider

```
#include <assert.h>

#include <cifuzz/cifuzz.h>
#include <fuzzer/FuzzedDataProvider.h>

#include "AddrIPv6andPort.hpp"


FUZZ_TEST_SETUP() {
  // Perform any one-time setup required by the FUZZ_TEST function.
}

FUZZ_TEST(const uint8_t *data, size_t size) {
    FuzzedDataProvider data_provider(data, size);

    std::vector<uint8_t> addr = data_provider.ConsumeBytes<uint8_t>(16U);
```

```
    uint16_t port = data_provider.ConsumeIntegral<uint16_t>();

    vwg::e3p::rsp::AddrIPv6andPort addr_and_port(addr, port);
}
```

12.4.1.3.2 Using GMock

Sometimes input comes from other classes that can be mocked with GMock, similar to the situation in unit tests:

TODO

Examples GMock

```
// TODO
```

12.4.1.3.3 Wrapping symbols via the linker

GMock can not be used to mock free functions like socket(), recv(), or send(), see https://google.github.io/googletest/gmock_cook_book.html#mocking-free-functions.

But this can be done with some help from the linker:

Examples Linker function wrapping (for C language)

```
#include <iostream>
#include <assert.h>
#include <sys/socket.h>

#include <cifuzz/cifuzz.h>
#include <fuzzer/FuzzedDataProvider.h>

FuzzedDataProvider* DP = nullptr;

extern "C" ssize_t __wrap_recv(int sockfd, void* buf, size_t len, int flags)
{
    // will be entered when "recv" is called. Advanced usages would for example
    // evaluate sockfd to select which socket to send the fuzzing input into.
    // The real recv can be called via __real_recv(...)
    auto output_buf = static_cast<uint8_t*>(buf);
    auto bytes = DP->ConsumeBytes<uint8_t>(len);
    ssize_t real_length = bytes.size();

    std::copy(bytes.begin(), bytes.end(), output_buf);
    return real_length;
}
```

```

int receiveDataFromSocket() {
    int sockfd = 460;
    char buffer[1024];

    // This is where the input is coming in:
    ssize_t received = recv(sockfd, buffer, 1024, 0);

    if(received <= 0){
        return 1;
    }

    // Some if blocks for the fuzzing engine to discover:
    if(received >= 5){
        if(buffer[0] == 'X')
            if(buffer[1] == 'y')
                if(buffer[2] == 'Z'){
                    return 0;
                }
    }
    return 2;
}

FUZZ_TEST_SETUP() {
  // Perform any one-time setup required by the FUZZ_TEST function.
}

FUZZ_TEST(const uint8_t *data, size_t size) {
    FuzzedDataProvider data_provider(data, size);
    DP = &data_provider; // Make data_provider available to other functions

    auto ret = receiveDataFromSocket(); // <- Has no input parameters, as input is received from socket
    std::cout << ret << " ";
}
```

Add linker options to CMakeLists.txt:

CMakeFile.txt

```
add_fuzz_test(wraptest wraptest.cpp)
target_link_libraries(wraptest PUBLIC ${LINK_LIBS})
target_link_options(wraptest PUBLIC "LINKER:--wrap,recv")
```

Given the wrap-recv example from above, the linker performs some renaming:

- The object file wraptest.o requires a definition of symbol “recv”, which will be provided by e.g. libc.so.
- With the wrap-recv option the linker adds the prefix “__wrap_” to the required symbol, and it also adds the prefix “__real_” to the provided symbol.

Without any glue-code linking will fail, as the required symbol is not provided.
You may now implement that glue code and define the "__wrap_"-variant to your needs.
You may also call the original function under its new name:

Example: wrapped function

```
// prototype is required as compiler is unaware of the wrapping
extern "C" ssize_t __real_recv(int sockfd, void* buf, size_t len, int flags);

extern "C" ssize_t __wrap_recv(int sockfd, void* buf, size_t len, int flags) {
  // do some stuff here
  if ( len % 2 == 0 ) {
    return 0;
  }
  return __real_recv(sockfd, buf, len, flags); // call original function
}
```

Note that this naming approach only works for C language. In order to wrap functions in C++, you must use the mangled names ("*the encoding of function and variable names into unique names so that linkers can separate common names in the language*"). To find the mangeled name of a specifc function, execute:

Command to find the mangeled name

```
$ nm [path to .o/.a file] | less
$ find . -iname "[pattern]*.o" // this can be used to find the .o file
```

Further info: Using LD, the GNU linker - Options[113], Ctrl+F "wrap"

Full size example: see commits associated with E3-102322[114] using wrap on e3_comp_socks_proxy

12.4.1.3.4 Injecting input via dynamic linking

From https://github.com/zardus/preeny#de-socketing:

> *"Certain tools (such as American Fuzzy Lop, for example) are unable to handle network binaries. Preeny includes two "de-socketing" modules.* `desock.so` *neuters* `socket()`, `bind()`, `listen()`, *and* `accept()`, *making it return sockets that are, through hackish ways, synchronized to* `stdin` *and* `stdout`. `desock_dup.so` *is a simpler version for programs that dup accepted sockets over file descriptors 0, 1, and 2."*

113 https://ftp.gnu.org/old-gnu/Manuals/ld-2.9.1/html_node/ld_3.html
114 https://devstack.vwgroup.com/bitbucket/plugins/servlet/jira-integration/issues/E3-102322

No pilot done yet.

If you chose to go this way and it works out, please get in touch with us so we can use your work as example. Thanks!

12.4.1.3.5 Input via file

Required	Provided
path to file	fuzz_utils.cc[115]::buf_to_file(), see 1. example
istream	istringstream(), see 2. example
FILE*	fmemopen(), see https://pubs.opengroup.org/onlinepubs/9699919799/functions/fmemopen.html

If the code under test reads input from a file (or preferably a file handle) we can create a temporary file with the input from the fuzzing engine and hand over the file name to the code under test.

Examples File input - file

```
#include <stddef.h>
#include <stdint.h>

#include <cifuzz/cifuzz.h>

#include "FileProcessing/Config/ConfigParser.hpp"
#include "FileProc/FileProc.hpp"

using namespace vwg::e3p::diag::lum;

#define FUZZING_TEMP_BASEPATH   "/dev/shm"

//
// Some code omitted for clarity.
// See  https://github.com/google/security-research-pocs/blob/master/autofuzz/fuzz_utils.cc
// for the original source
//

FUZZ_TEST_SETUP() {
  // Perform any one-time setup required by the FUZZ_TEST function.

  // FileProc does not allow arbitrary paths. Whitelist the base path.
  // FileProc::addTrustedPath() is a static function and must only be executed once
```

115 https://github.com/google/security-research-pocs/blob/master/autofuzz/fuzz_utils.cc

```
  vwg::e3p::diag::lum::mfw::FileProc::addTrustedPath(FUZZING_TEMP_BASEPATH);
}

FUZZ_TEST(const uint8_t *data, size_t size) {
  // write the content of the given array into a temporary file
  // and return the name to that file
  char* pFilename = buf_to_file(data, size);

  if ( nullptr == pFilename ) {
    warn( "FAILED to create temp file");
    return;
  }

  Config config;
  ConfigParser configParser(config);

  std::string path(pFilename);
  configParser.loadConfig(path);

  // reset state and free all locally allocated resources
  delete_file(pFilename);
}
```

Repo: https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_lum/browse/fuzzing/fuzz_ConfigParser_loadConfig.cpp?at=custom%2Fcifuzz

In this example the fuzzing input is fed via istringstream to the function under test:

Examples File input - stream

```
#include <iostream>
#include <sstream>
#include <cifuzz/cifuzz.h>

#include "FileProcessing/Persistency/PersistencyParser.hpp"
#include "minfw/Exception.hpp"

using namespace vwg::e3p::diag::lum;

FUZZ_TEST_SETUP()
{
  // Perform any one-time setup required by the FUZZ_TEST function.
}

FUZZ_TEST(const uint8_t *data, size_t size)
{
  std::string content(&data[0], &data[size]); // string from array
  std::istringstream isstream{content};       // istringstream from string
  PersistencyParser persistencyParser;

  try {
```

```
      persistencyParser.parse(isstream);
    } catch ( mfw::FileCorruptException e ) {
      // do nothing, this is expected
    }
  }
```

Repo: https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_lum/browse/fuzzing/fuzz_PersistencyParser.cpp?at=custom%2Fcifuzz

12.4.1.4 Handling structured input

Given that your application uses an off-the-shelf publicly available library to parse input, you do not want to spend cpu cycles fuzzing that library. Chances are good, that that was already done by e.g. OSS-Fuzz (see [7][116]).

Instead you want to inject evil values into a valid structure to pass the validation stage of the library and reach your application logic.

TODO: (Add example here)

There are several possibilities to tackle that problem:

Option	description	Effort	Chance to explore application logic	Further info
custom mutator	build a custom mutator to only produce valid data containers	high	excellent	https://github.com/google/fuzzing/blob/master/docs/structure-aware-fuzzing.md#structure-aware-fuzzing-with-libfuzzer
protobuf mutator	Describe your structure or a subset of it with proto, then use the protobuf mututor to fuzz it	medium	depends on your grammar	https://github.com/google/fuzzing/blob/master/docs/structure-aware-fuzzing.md#protocol-buffers-as-intermediate-format https://github.com/google/libprotobuf-mutator

116 https://github.com/google/oss-fuzz

Option	description	Effort	Chance to explore application logic	Further info
manual seeds	Add a pool of "good" seeds, i.e. valid input data the fuzzer can use as a starting point.	low	medium	https://chromium.googlesource.com/chromium/src/testing/libfuzzer/+/refs/heads/main/efficient_fuzzing.md#seed-corpus https://llvm.org/docs/LibFuzzer.html#corpus `./cifuzz.sh run -s <path to seed directory>`
text dictionary	add a dictionary of valid snippets the fuzzer can choose from	low	depends on structure	https://chromium.googlesource.com/chromium/src/testing/libfuzzer/+/refs/heads/main/efficient_fuzzing.md#fuzzer-dictionary
Raw fuzzing engine	Let the fuzzer inject arbitrary input to your parser	low	• low for highly structured data • high for simple structures	

12.4.1.5 Handling state

From the perspective of the fuzzing engine a few things are pretty important:

1. The code under test should act deterministic for a given fuzzing input. That means there should not be another source of "randomness".
2. The code under test should be single-threaded.

12.4.1.5.1 Fuzzing stateful APIs

Some APIs may store state after a function is called so the order of function calls does matter. Example: sha256.init(), sha256.update(data), sha256.finalize()

Consider chaining several function calls together, using a single fuzzing input.

Do not maintain state over several fuzzing calls, as that ruins reproducibility; each fuzzing iteration must be independent from previous ones.

Further information: https://github.com/google/fuzzing/blob/master/docs/structure-aware-fuzzing.md#fuzzing-stateful-apis

Basic structure of stateful API fuzzing

Example: Call chaining

```
/*
  CMyApiContext is an imaginary API with at least these testable functions
    void init();
    void addName(std::string name);
    void removeName(std::string name);
    std::string finalize();
 */
FUZZ_TEST(const uint8_t *data, size_t size) {

  FuzzedDataProvider fuzzed_data(data, size);
  CMyApiContext ctx;

  // how many commands to chain?
  uint32_t numSteps = fuzzed_data.ConsumeIntegralInRange(1, 20);

  // execute chained steps
  for (uint32_t step = 0; step < numSteps; step++) {

    // select function to call
    uint32_t selected = fuzzed_data.ConsumeIntegralInRange(0, 3);

    switch (selected) {
      case 0:
        ctx.init();
        break;

      case 1: {
        std::string name{fuzzed_data.ConsumeRandomLengthString(20)};
        ctx.addName(name);
        break;
      }

      case 2: {
        std::string name{fuzzed_data.ConsumeRandomLengthString(20)};
        ctx.removeName(name);
        break;
      }

      case 3:
        ctx.finalize();
        break;

      default:
        break;
    }
  }
```

```
}
```

12.4.2 Maintaining a fuzz target

12.4.2.1 Identifying blockers

At some stage the coverage will not improve any further: it saturates at a steady level.

- create coverage report
  - identify important branches which were not taken
  - try to deduct why these branches were not taken
    - structure of input data insufficient?

12.4.2.2 Handling blockers

TBD

- last resort: flag out code with FUZZING_BUILD_MODE_UNSAFE_FOR_PRODUCTION

12.4.2.3 Documentation

It is suggested to create a one-pager for each(!) fuzz target and store it alongside the source, i.e. in the ./fuzzing-directory

```
./fuzzing
+- fuzz_target_name.cpp    # impl. of the fuzz target
+- fuzz_target_name.html   # snapshot of coverage report for future
reference
+- fuzz_target_name.md     # one pager documentation
```

Please use the following template to create the one-pager:

```
# Fuzz target spec: <NAME OF TARGET>
Last update: <DATE>, <EDITOR>

## Approach
<BRIEF OVERVIEW ON HOW THE COMPONENT UNDER TEST IS SETUP,>

## Preconditions
<LIST OF PRECONDITIONS>

## Coverage
```

```
### time and coverage reached after 10min / 1core fuzzing

```
<ADD TABLE HERE>
```

Further potential if used on CI/CD
<EXPECTATION GIVEN MORE PROCESSING TIME>

Further work
<IS THE CURRENT SETUP SUFFICIENT? IS ALL RELEVANT CODE REACHED, OR IS
THERE ROOM FOR IMPROVEMENT?>
```

Examples:

- fuzz_ConfigParser_loadConfig.md[117]
- fuzz_ManifestReader.md[118]
- fuzz_mfw_utils.md[119]
- fuzz_PersistencyParser.md[120]
- fuzz_TomlParser_parseFile.md[121]

12.4.3 Debugging a fuzz target

12.4.3.1 Analyzing build problems (CMake, Compiler, Linker)

If your fuzz target does not build:

- analyze console output
- increase verbosity if necessary
  - cifuzz option: --verbose
  - `CMake: set(CMAKE_VERBOSE_MAKEFILE ON)`

117 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_lum/browse/fuzzing/fuzz_ConfigParser_loadConfig.md?at=refs%2Fheads%2Fcustom%2Fcifuzz
118 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_lum/browse/fuzzing/fuzz_ManifestReader.md?at=refs%2Fheads%2Fcustom%2Fcifuzz
119 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_lum/browse/fuzzing/fuzz_mfw_utils.md?at=refs%2Fheads%2Fcustom%2Fcifuzz
120 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_lum/browse/fuzzing/fuzz_PersistencyParser.md?at=refs%2Fheads%2Fcustom%2Fcifuzz
121 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_lum/browse/fuzzing/fuzz_TomlParser_parseFile.md?at=refs%2Fheads%2Fcustom%2Fcifuzz

12.4.3.2 How to use coverage info to find blockers

At some stage your fuzzing run will not reach any new edges and your coverage saturates at a certain level.

cifuzz progress indication

```
# 00:13:36 - still progress: the last edge was found 8 seconds ago
paths: 3947 - last new Path: 8s - exec/s: 753 (13m36s)

# 01:23:20 - likely saturated: the last edge was found 7 minutes ago
paths: 4814 - last new Path: 7m13s - exec/s: 697 (1h23m20s)
```

Create a coverage report to get an overview on fuzzing progress.

The coverage information is created by instrumenting your target for coverage and rebuilding it. That new target is then used to replay the existing corpus; the corpus is not touched.

Corpus

The corpus is a minimized set of input vectors which leads to the shown coverage.
If your corpus consists of N files, your fuzz target is called exactly N times during coverage generation.
Corpus size is unrelated to number of edges found.

Simple report (single file)

```
# Simple report
# =============
# - single html file
# - easy to create, but also easy to get lost in it
./cifuzz.sh coverage my_fuzz_test_1 --output=fuzzing/my_fuzz_test_1.html


# Detailed report
# ===============
#   1. Generate coverage information in lcov format
#   2. Optional: use 'lcov' to post-process coverage file ('man lcov')
#   3. Run 'genhtml' to create a report
./cifuzz.sh coverage my_fuzz_test_1 --output=fuzzing/my_fuzz_test_1.lcov --format=lcov
--build-jobs=$(nproc)
genhtml fuzzing/my_fuzz_test_1.lcov --output-directory fuzzing/my_fuzz_test_1 --branch-
coverage
```

LCOV - code coverage report

			Hit	Total	Coverage
Current view:	top level - src/FileProcessing/Config				
Test:	fuzz_ManifestReader.lcov	**Lines:**	189	420	45.0 %
Date:	2022-12-15 20:24:52	**Functions:**	40	77	51.9 %
		Branches:	43	238	18.1 %

Filename	Line Coverage			Functions		Branches	
Config.cpp		52.5 %	107 / 204	11.1 %	1 / 9	0.0 %	0 / 36
Config.hpp		40.9 %	27 / 66	48.0 %	12 / 25	22.9 %	11 / 48
ConfigItem.cpp		100.0 %	4 / 4	100.0 %	2 / 2	-	0 / 0
ConfigItem.hpp		73.1 %	19 / 26	77.8 %	7 / 9	16.7 %	1 / 6
ConfigItem.ipp		36.0 %	9 / 25	33.3 %	5 / 15	20.8 %	25 / 120
ConfigItemCheck.cpp		5.9 %	4 / 68	40.0 %	2 / 5	0.0 %	0 / 12
ConfigItemCheck.hpp		100.0 %	6 / 6	100.0 %	4 / 4	-	0 / 0
ConfigItemCheck.ipp		61.9 %	13 / 21	87.5 %	7 / 8	37.5 %	6 / 16

Generated by: LCOV version 1.14

Inspect the coverage of your function under test:

- detect relevant branches which were not taken; 'relevant' as in: not error handling
- try to deduce why these branches were not taken.
  Does the fuzzer have **any** leverage to discover these branches, i.e. is there a direct or indirect relation between fuzzing input and condition?

  YES

  - Does the fuzzer have a realistic chance to discover these branches?
    If the fuzzing input is sanitized and has to match a given magic hash, the chance that the fuzzer finds correct input are slim.
    In that case you could resort to flagging out the blocker using the defined `FUZZING_BUILD_MODE_UNSAFE_FOR_PRODUCTION`
  - Consider adding more structure to the input, review options under "Handling structured input"

  NO

  - Check that the preconditions of the functionality under test are met
  - Compare with the unit test setup

12.4.3.3 How to debug a fuzz target (with gdb or VSCode)

This is similar to debugging a unit test, therefor it is not explained in detail here.

The fuzz targets is executed in the docker container, therefor you need to enter the docker container and start debugging with gdb or gdbserver.

- enter the docker container using `./cifuzz.sh /shell`
- set the `"NO_CIFUZZ=1"` environment variable to enable standalone mode and launch gdb:
  `NO_CIFUZZ=1 gdb <path_to_executable>`

- Select the asan-undefined instrumented executable

If you already use a gdbserver setup adjust it accordingly.

To debug with VSCode on the host machine, you can use our script *cifuzz.sh* to spawn a gdbserver inside a docker container which will send all the debugging information to *localhost:55555*:

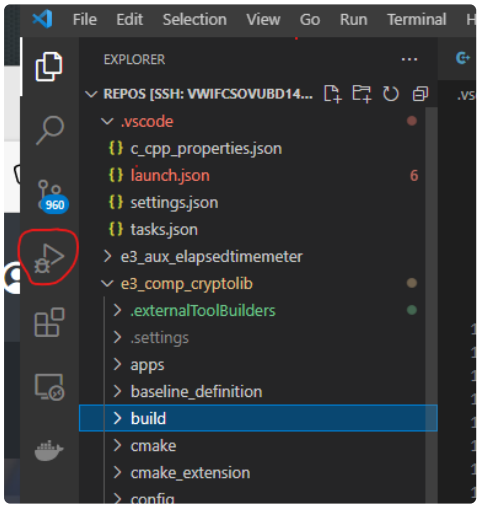
Debug with a combination of VSCode and gdbserver

```
$ ./cifuzz.sh /debug [path to fuzztarget bin] [path to fuzz input]
# For example
$ ./cifuzz.sh /debug .cifuzz-build/libfuzzer/address+undefined/fuzzing/fuzz_processGET
 fuzzing/fuzz_processGET_inputs/kind_emu
```

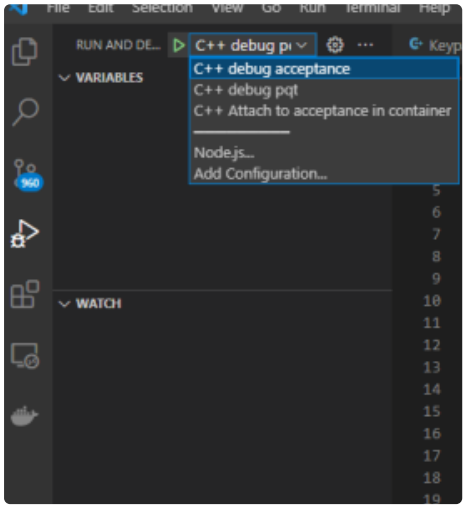
Inside your repository, create a .vscode/launch.json file with followings content to receive the debugging information from the gdbserver:

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "(gdb) Remote debug cifuzz",
            "type": "cppdbg",
            "request": "launch",
            "program": "/path to the binary executable e.g /.../
e3_comp_otam_common/.cifuzz-build/libfuzzer/address+undefined/fuzzing/fuzz_processGET",
            "args": [],
            "stopAtEntry": true,
            "cwd": "${fileDirname}",
            "environment": [],
            "externalConsole": false,
            "filterStderr": false,
            "filterStdout": false,
            "preLaunchTask": "",
            "sourceFileMap":{
                "/your home directory": "/home/e3-user"
            },
            "miDebuggerServerAddress": "localhost:55555",
            "MIMode": "gdb",
            "setupCommands": [
                {
                    "description": "Enable pretty-printing for gdb",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                }
            ]
        }
    ]
}
```

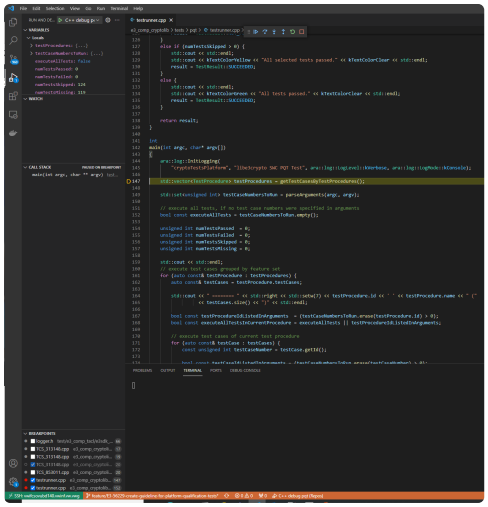
Then, in VSCode, open the Debug View:



After the debug view is activated we choose our debug configuration from the drop down. (The last active configuration can be started via F5 as well):



After that the normal debugging flow can be started:



12.4.3.4 How to debug a finding

Use `./cifuzz.sh findings` to get an overview of findings.

cifuzz progress indication

```
$ ./cifuzz.sh findings
clever_peafowl       deadly              in abort (abort.c:79:7)
hopeful_possum       undefined behavior in exploreMe (fuzzing/cmake_example/src/
explore_me.cpp:13:11)
```

Use `./cifuzz.sh findings finding_name` to get details of a finding.

cifuzz progress indication

```
$ ./cifuzz.sh findings hopeful_possum
[hopeful_possum] undefined behavior in exploreMe (fuzzing/cmake_example/src/
explore_me.cpp:13:11)
Date: 2022-12-09 14:22:43.225889724 +0100 CET

  /home/developer/repos/20221202_e3_comp_lum_test-run/fuzzing/cmake_example/src/
explore_me.cpp:13:11: runtime error: shift exponent 32 is too large for 32-bit type
'int'
      #0 0x560627ebd3f1 in exploreMe(int, int, std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>) /home/developer/repos/
20221202_e3_comp_lum_test-run/fuzzing/cmake_example/src/explore_me.cpp:13:11
      #1 0x560627ebbf1b in LLVMFuzzerTestOneInputNoReturn(unsigned char const*,
unsigned long) /home/developer/repos/20221202_e3_comp_lum_test-run/fuzzing/
cmake_example/my_fuzz_test.cpp:18:3
      #2 0x560627ebbf1b in LLVMFuzzerTestOneInput /home/developer/repos/
20221202_e3_comp_lum_test-run/fuzzing/cmake_example/my_fuzz_test.cpp:8:1
      #3 0x560627de1b12 in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*,
unsigned long) (/home/developer/repos/20221202_e3_comp_lum_test-run/.cifuzz-build/
libfuzzer/address+undefined/bin/my_fuzz_test+0x40b12) (BuildId:
06588a474294cfa8957b36db6f308fb365fc8fb9)
      #4 0x560627de1225 in fuzzer::Fuzzer::RunOne(unsigned char const*, unsigned long,
bool, fuzzer::InputInfo*, bool, bool*) (/home/developer/repos/
20221202_e3_comp_lum_test-run/.cifuzz-build/libfuzzer/address+undefined/bin/
my_fuzz_test+0x40225) (BuildId: 06588a474294cfa8957b36db6f308fb365fc8fb9)
      #5 0x560627de2ab7 in fuzzer::Fuzzer::MutateAndTestOne() (/home/developer/repos/
20221202_e3_comp_lum_test-run/.cifuzz-build/libfuzzer/address+undefined/bin/
my_fuzz_test+0x41ab7) (BuildId: 06588a474294cfa8957b36db6f308fb365fc8fb9)
      #6 0x560627de36a5 in fuzzer::Fuzzer::Loop(std::vector<fuzzer::SizedFile,
std::allocator<fuzzer::SizedFile>>&) (/home/developer/repos/20221202_e3_comp_lum_test-
run/.cifuzz-build/libfuzzer/address+undefined/bin/my_fuzz_test+0x426a5) (BuildId:
06588a474294cfa8957b36db6f308fb365fc8fb9)
      #7 0x560627dd1e52 in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned char
const*, unsigned long)) (/home/developer/repos/20221202_e3_comp_lum_test-run/.cifuzz-
build/libfuzzer/address+undefined/bin/my_fuzz_test+0x30e52) (BuildId:
06588a474294cfa8957b36db6f308fb365fc8fb9)
      #8 0x560627dfaab2 in main (/home/developer/repos/20221202_e3_comp_lum_test-
run/.cifuzz-build/libfuzzer/address+undefined/bin/my_fuzz_test+0x59ab2) (BuildId:
06588a474294cfa8957b36db6f308fb365fc8fb9)
      #9 0x7f7c489720b2 in __libc_start_main ../csu/libc-start.c:308:16
      #10 0x560627dc69ed in _start (/home/developer/repos/20221202_e3_comp_lum_test-
run/.cifuzz-build/libfuzzer/address+undefined/bin/my_fuzz_test+0x259ed) (BuildId:
06588a474294cfa8957b36db6f308fb365fc8fb9)

  SUMMARY: UndefinedBehaviorSanitizer: undefined-behavior /home/developer/repos/
20221202_e3_comp_lum_test-run/fuzzing/cmake_example/src/explore_me.cpp:13:11 in MS: 2
 CrossOver-InsertByte-; base unit: f130e52d1c89bf6c78932a72b99ee02ce90cf86d
  0xf1,0xf1,0xf1,0xf1,0x37,0xf1,0xd5,0xf5,
  \361\361\361\3617\361\325\365
```

```
  artifact_prefix='/tmp/libfuzzer-out-4008698476/'; Test unit written to /tmp/
libfuzzer-out-4008698476/crash-3a8a23ffabc8ed2d140384a382903b4a550831f3
  Base64: 8fHx8Tfx1fU=
    NEW_FUNC[1/1]: 0x560627ebcc20 in FuzzedDataProvider::Advance(unsigned long) /home/
e3-user/.local/share/cifuzz/share/cifuzz/cmake/../include/cifuzz/fuzzer/
FuzzedDataProvider.h:355
```

Enter the shell, set options via environment variables and launch gdb to replay a finding:

Reproducing findings

```
# enter shell
$./cifuzz.sh /shell

# ensure that the ubsan finding stops gdb but does not kill the process
e3-user@242657e1552b:$ export UBSAN_OPTIONS='halt_on_error=1:abort_on_error=1'

# skip CI fuzz wrapper
e3-user@242657e1552b:$ export NO_CIFUZZ=1

# To simply execute the binary with a certain input, do the followings:
NO_CIFUZZ=1 .cifuzz-build/libfuzzer/address+undefined/bin/my_fuzz_test .cifuzz-
findings/predictive_barnacle/crashing-input

# start gbd and run the fuzz target with the input which caused the issue
# Optionally pass the argument "-ex 'break LLVMFuzzerTestOneInput'" to set a breakpoint
at the start of the fuzz target
e3-user@242657e1552b:$ gdb -ex run --args .cifuzz-build/libfuzzer/address+undefined/
bin/my_fuzz_test .cifuzz-findings/hopeful_possum/crashing-input

# All in one
e3-user@242657e1552b:$ UBSAN_OPTIONS='halt_on_error=1:abort_on_error=1' NO_CIFUZZ=1 gdb
-ex 'break LLVMFuzzerTestOneInput' -ex run --args .cifuzz-build/libfuzzer/
address+undefined/bin/my_fuzz_test $(find -type f -iname hopeful_possum)
```

12.5 Deploying fuzz target to the CIFuzz Server

12.5.1 Server Information

Server address: https://cifuzz.dev.hub.vwgroup.com/

Status: **NOT READY** The server is currently still under construction and is not to be used. Alternatively, we can use this mock up server to test https://cifuzz-megatron-dev.westeurope.cloudapp.azure.com[122] or set up our own server locally. If you want to run your own

122 https://cifuzz-megatron-dev.westeurope.cloudapp.azure.com/login

server locally, please contact Code-Intelligence to get the server bundle and follow their instruction here https://docs.code-intelligence.com/ci-fuzz-installation to install the server.

12.5.2 Creating and uploading the bundle to the server

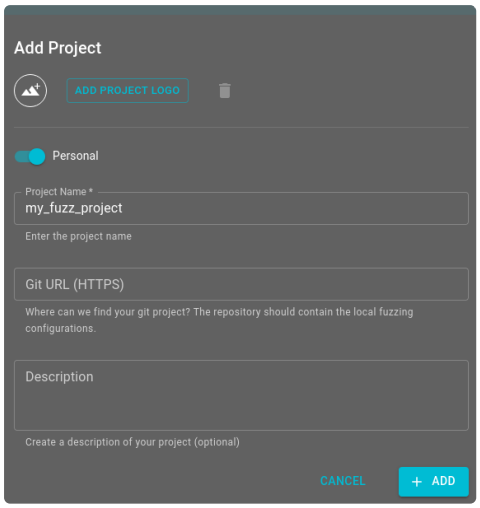
First of all, we need to create a bundle for our fuzz target by executing following command inside the root folder of our project:

Creating fuzz target bundle named my_fuzz_test_1_bundle.tar.gz with cifuzz

```
./cifuzz.sh bundle my_fuzz_test_1 -o my_fuzz_test_1_bundle.tar.gz --build-jobs=$(nproc)
-v
# -v is optional to see verbose output for more information in case our bundle creation
fails
# --build-jobs=$(nproc) utilizes multithreading to speed up the compiling process
```

The bundle command above will create a tar.gz file containing 2 binaries (one to execute the fuzz test and the other one to provide code coverage information) and all the libraries our 2 binary files require.

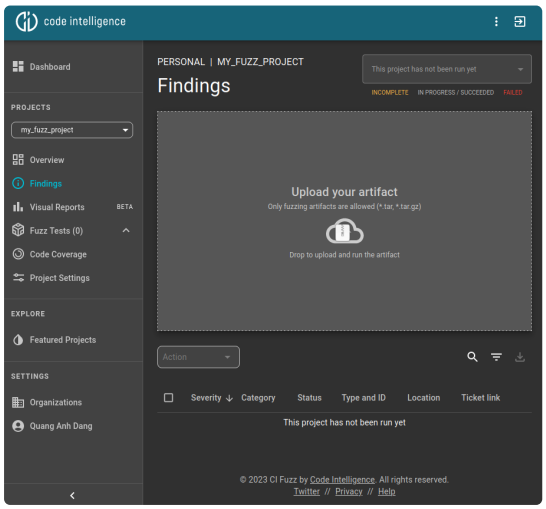
After creating our bundle, we can proceed on to the cifuzz server and create a new project. On the main page, go to "Select Project" and choose "Add project". Then, provide project name and project GIT Repo URL and description if needed:



After creating a fuzz project, there are 3 ways to upload our bundle to the cifuzz server:

12.5.2.1 Drag and drop

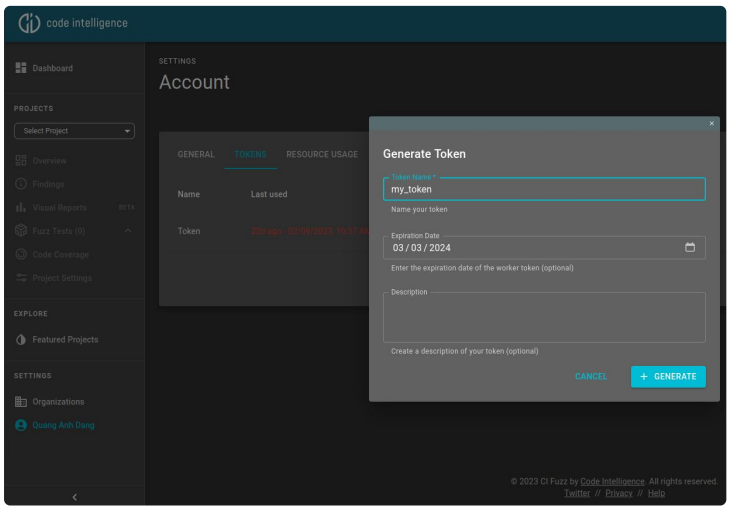
On the cifuzz server page, choose our project on the left side panel and drag and drop the my_fuzz_test_1_bundle.tar.gz file previously generated on to the project area.



Wait for the upload to finish and the fuzz test will start by itself.

12.5.2.2 Cifuzz remote-run

Alternatively, the bundle can be uploaded to the server via following command. This command, however, requires you to generate an access token before hand. To do this, go to the cifuzz server webpage. On the left panal, click on your name, "TOKENS" and "GENERATE TOKEN". Note down your token, because you won't be able to access this again.



Upload the bundle to cifuzz server via cifuzz remote-run

```
$ ./cifuzz.sh remote-run --bundle my_fuzz_test_1_bundle.tar.gz --server https://cifuzz.
dev.hub.vwgroup.com --project my_fuzz_project-7acafdc2
You need to authenticate to a CI Fuzz Server instance to use this command.
Log in now? [Y/n]: Y
```

```
You need an API access token which can be generated here:
https://cifuzz-megatron-dev.westeurope.cloudapp.azure.com/dashboard/settings/account/
tokens?create=&origin=cli
Open browser to generate a new token? [Y/n]: n
Paste your access token:
✓ Successfully authenticated with https://cifuzz-megatron-dev.westeurope.cloudapp.azur
e.com
Your API access token is stored in /home/e3-user/.config/cifuzz/access_tokens.json
Uploading...
Upload complete
✓ Successfully started fuzzing run. To view findings and coverage, open:
    https://cifuzz-megatron-dev.westeurope.cloudapp.azure.com/dashboard/projects/
my_fuzz_project-753a4e62/campaign_runs/93e3d2a5-f752-4397-ac18-07b83d7f31d6/overview?
origin=cli
```

Info

The URL passed to the --server parameter must not contain a trailing backslash '/'.

Info

The project name can be found in the URL when accessing the project on the cifuzz webpage e.g https://cifuzz-megatron-dev.westeurope.cloudapp.azure.com/dashboard/projects/my_fuzz_project-753a4e62/campaign_runs/last_run/findings

12.5.2.3 cictl tool

The cictl provides an usefull API to access and interact with the cifuzz Server. This would be useful for automation of the fuzzing process via Jenkins or Github in the future. Naturally, we can also use this to upload our fuzz target to the server.

Login to the docker container

```
# First, login to our docker container. There, we have the cictl tool prepared for you
./cifuzz.sh /shell
```

Then, create a script to execute the followings:

Steps to upload our fuzz target to the server via cictl tool

```
#!/bin/bash
```

```
set -ex

# Minimum severity for findings that causes the pipeline to fail.
# Findings with lower severity are still reported but do not fail
# the pipeline.
# Possible values: 'LOW', 'MEDIUM', 'HIGH', 'CRITICAL'
MIN_FINDINGS_SEVERITY=MEDIUM

# Then, define some parameters. Note that the server address should not contain
https:// nor a backslash in the end
FUZZING_SERVER=cifuzz-megatron-dev.westeurope.cloudapp.azure.com:443
CICTL_CMD="cictl --server ${FUZZING_SERVER}"
BUNDLE_LOCATION=./my_fuzz_test_1_bundle.tar.gz
PROJECT=projects/my_fuzz_project-753a4e62
WEB_APP_ADDRESS=https://cifuzz-megatron-dev.westeurope.cloudapp.azure.com
TIMEOUT=300
CI_FUZZ_API_TOKEN=[[Insert your Access TOKEN here]]

# Then, login to the server
echo "${CI_FUZZ_API_TOKEN}" | ${CICTL_CMD} login --quiet

# Then, upload the bundle to the server
ARTIFACT_NAME=$(${CICTL_CMD} import artifact ${BUNDLE_LOCATION} --project-name "$
{PROJECT}")

# Finally, start campaign run via
CAMPAIGN_RUN_NAME=$(${CICTL_CMD} start "${ARTIFACT_NAME}")

# Observe Campaign Run with timeout, watching for findings
${CICTL_CMD} monitor_campaign_run \
  --dashboard_address="${WEB_APP_ADDRESS}" \
  --duration="${TIMEOUT}" \
  --min_findings_severity="${MIN_FINDINGS_SEVERITY}" \
  ${CAMPAIGN_RUN_NAME}
```

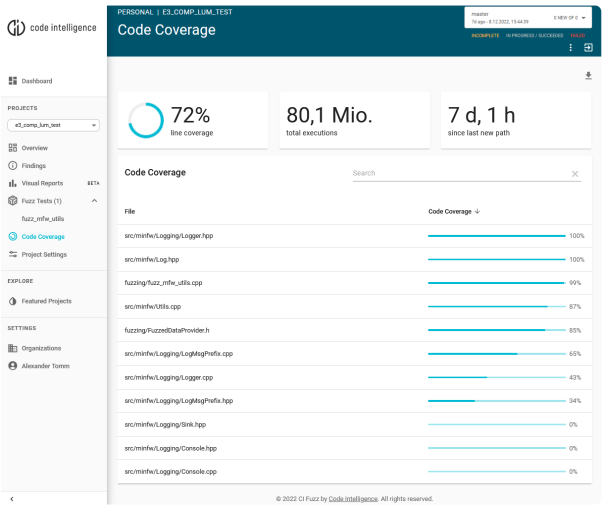
12.6 Outlook

12.6.1 Server side fuzzing

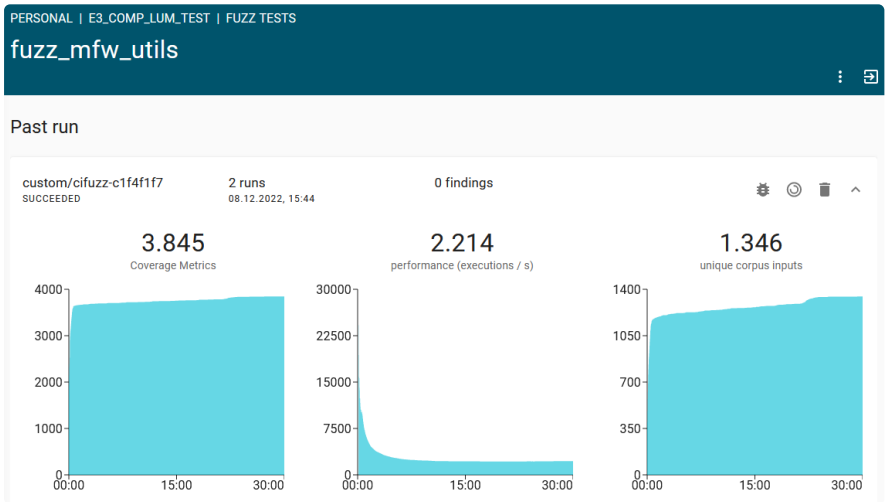
So far focus was on local fuzz target creation, maintenance and execution.

Ultimately fuzz targets will be built by the CI/CD and uploaded to a fuzzing server where they are executed.

Details on coverage:



Details on a fuzzing run



Can only be used to track progress, but cannot be used as criteria for completeness.

12.6.2 SDK Integration

Currently CI FUZZ CLI has to be installed into the build container by the user.

Eventually the tool will either be integrated in every build container, or separate pre-configured build containers will be provided.

12.7 References & Further Reading

[1] Patrice Godefroid, Microsoft research "Fuzzing: Hack, Art, and Science"
Overview on fuzzing techniques
https://patricegodefro id.github.io/public_psfiles/Fuzzing-101-CACM2020.pdf[123]

[2] google / fuzzing
Tutorials, examples, discussions, research proposals, and other resources related to fuzzing
https://github.com/google/fuzzing

[3] libprotobuf-mutator
Library for structured fuzzing with protocol buffers
https://github.com/google/libprotobuf-mutator

[4] The Fuzzing Book
Tools and Techniques for Generating Software Tests
https://www.fuzzingbook.org/

[5] Fuzzing: Challenges and Reflections
Paper by Marcel Böhme et al. on current state of fuzzing (2020)
https://mboehme.github.io/paper/IEEESoftware20.pdf

[6] Chromium: Efficient Fuzzing Guide
Guideline on how to improve fuzzing efficiency with libfuzzer
https://chromium.googlesource.com/chromium/src/testing/libfuzzer/+/refs/heads/main/efficient_fuzzing.md

[7] Google / OSS-Fuzz
Google's ClusterFuzz[124] instance to improve the quality of open source components
https://github.com/google/oss-fuzz

[8] Fuzzing software
Common challenges and potential solutions (Part 1) / Advanced tricks (Part 2)
https://securitylab.github.com/research/fuzzing-challenges-solutions-1/
https://securitylab.github.com/research/fuzzing-software-2/

[9] Debugging with VSCode and gdbserver

- LUM: Debugging in Docker[125]
- Cryptonite: Visual Studio Code[126]

[10] Workshop: Fuzzing 101 & advanced session SLIDES

CARIAD Workshop 2023-05-04.pdf[127]

123 https://patricegodefroid.github.io/public_psfiles/Fuzzing-101-CACM2020.pdf
124 https://github.com/google/clusterfuzz
125 https://devstack.vwgroup.com/confluence/display/E3SPPIF/Debugging+in+Docker
126 https://devstack.vwgroup.com/confluence/display/E3SPPS/Visual+Studio+Code
127 https://devstack.vwgroup.com/confluence/download/attachments/860960818/CARIAD%20Workshop%202023-05-04.pdf?api=v2&modificationDate=1683730526089&version=1

12.8 Downloads

File	Modified
cifuzz.sh[128] CIFuzz helper script from team Security	Apr 12, 2023 by Dang, Quang Anh (EXTERN: ITK)[129]

128 https://devstack.vwgroup.com/confluence/download/attachments/860960818/cifuzz.sh?api=v2
129 https://devstack.vwgroup.com/confluence/display/~gchogqb

13 (Deprecated) Process of fuzzing & release criteria/ definition of done (in progress)

DEPRECATED: new version -> 1.1 ADA+1.2+2.0 - VW.os - Workflow of fuzzing & release criteria/ definition of done[130] && 1.1 ADA +2.0 - Guideline to extract a list of security relevant fuzzing items[131]

13.1 Abbreviations

TODO: Dang, Quang Anh (EXTERN: ITK) link Central Process Glossary[132] instead of writing our own Abbreviations

- SRA: Security Risk Accessment
- SRS: Security Requirement Specification

13.2 Agenda & Purpose

Fuzzing is to become part of the requirement, meaning all components deemed security relevant by the SRS is to be included into Test Master Plan in 2.0. In general, fuzz tests should be run in the background continuously, even for released product (if possible). However, due to product release, we need to "draw a line" at some point and consider the question: "Is the product secured enough regarding fuzzing to be released?". Moreover, eventually when the cifuzz server is extensively used, we would have to prioritize different components to fuzz due to limited resources. Corespondingly, this confluence page documents the followings:

- Which components need to be fuzzed?
- Definition of done aka. release criteria for products in regarding to fuzzing

13.3 Timeline

When?	To be achieved?
soon	Cifuzz server is operational
30 Apr 2023	First draft of exit criteria for the team to orientate themselves
30 Jun 2023	Improved version of fuzzing exit criteria

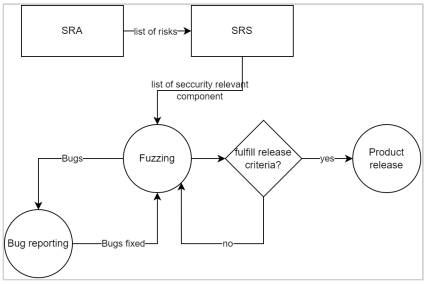
130 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1378239380
131 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1378239468
132 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/605198312/Central+Process+Glossary

When?	To be achieved?
30 Jun 2023	Fuzzing is to be inclued into Test Master Plan 2.0

13.4 Good metrics

Metric	Pros	Cons	Source
Code coverage	Simple to implement	Doesn't consider dead/irrelevant code Code coverage of what? Function? Branch? Line? Which files? Whole repository?	cifuzz tool
Code coverage from security relevant components	Avoid dead/irrelevant code	Need work from Architecture/Test Engineer	cifuzz tool
#New Paths / time	Simple to implement Indicate saturation and fuzzer's limit	Doesn't give info about code coverage	cifuzz tool
CPU/Memory/Storage usage	Must be considered eventually since every server has limited resources	Not a good metric. This needs to be combined with others Usage information must come from SDK team Code Intelligence needs to provide a functionality to access this information	SDK Team
Function Coverage	Most simple to implement Good with list of functions from SRS	Only provide information about which function is called, not the percentage of code being executed inside the function	cifuzz tool
Branch Coverage	More thorough than Function Coverage	Difficult as hard-requirement. More as recommendation	cifuzz tool
Line Coverage	The most thorough metric	Difficult as hard-requirement. More as recommendation	cifuzz tool

13.5 Process of fuzzing and exit criteria

Proposal	How?	Problem	TODO
SRA based	SRA gives information about which component has security risks on a high level. SRS takes the information from SRA and derives Security Requirements which need to be implemented to mitigate the risks. SRS (Requirement Engineer) should gives a list of security relevant components. These security relevant components must then be: • Unit tested • Fuzzed • (Pentested) 	SRA is abstract (module and data flow based) and not granulate and code specific (e.g C++). This is the responsiblity of the requirement engineer to	PoC with team Remote Repair is currently ongoing
Continuous fuzzing	All components should be fuzzed continuosly with a scheduler. Priority is to be determined: • Components with known bugs/ vuln have higher priority • Components without known bugs/vuln have lower priority	Code Intelligence and SDK team should provide sheduler functionality with automatic bug reporting in JIRA	Ask Code Intelligenc e about this matter
Bug/ Vulnerability triggered fuzzing	Components with known bugs and vulnerability should be prioritized in term of fuzzing		
Based on "Security Relevant" functions tag	Currently, there are already tagging in place (Code-Beamer) which specify which components are security relevant. Creating fuzz test for these components	No information about criticality. This is more of a temporary solution	

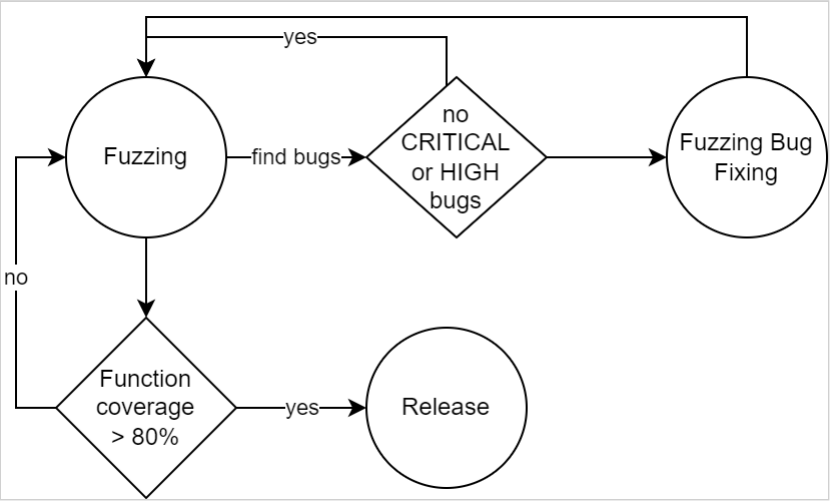
13.5.1 Release criteria

In case of findings:

- Is the bug resolved?
- Criticality:
  - CRITICAL and HIGH must be resolved before release
  - Risk Evaluation before release. If we have SRA => we have the link to Risk Analysis right away => Can be evaluated right away whether or not it can be released.
  - Regarding MEDIUM findings => depends on our Security Level/Goal

For the product to be ready for release, following criterias must be fulfilled:

- If there are still unfixed bugs, it must be differencitate between following cases:
  - If all the bugs are MEDIUM or LOW, then a hot fix for these bugs should be implemented so the fuzzing process can be proceeded. Fuzzing will halt as long as the fuzzer can identify bug regardless of the criticality of the bug. These hot fix can be a simple "skipping the function" with the bug so the fuzz test can proceed.
  - If there are CRITICAL or HIGH bugs, these bugs should be fixed before release. Also consider to build a hot fix so the fuzzing process can be proceeded while waiting for the fix.
- If there is no bug found, check the function coverage of the security relevant functions (from SRS). If the function coverage is larger than 80% (temporal value - the exact value is to be determined), the product is ready for release.



13.6 Open Questions

TODO: Define concrete percent of function coverage

TODO: Cifuzz Server should provide a scheduler, to jobs "abarbeiten"

TODO: Incorporate, if a bug is found which belong to another team, what to do?

TODO: Clarify how to extract this list of sec relevant functions from SRS => Platform Requirement => Component Requirement => List of Functions and class with Sec Relevance

Dang, Quang Anh (EXTERN: ITK) set up a separate meeting with Hartung, Martin (EXTERN: ITK) to discuss this in detail maybe at the retro meeting in Nürnberg on 18 Apr 2023 , invite Weichslgartner, Andreas, Dr. (TX-NS)

TODO: We want 100% function coverage. Document this

TODO: 80% code coverage of each requirement maybe?

- Start with 80% then tone it down
- Flexible, team can say due to complex logic => certain code can not be reached without complex modification of fuzz input => Could still be accepted

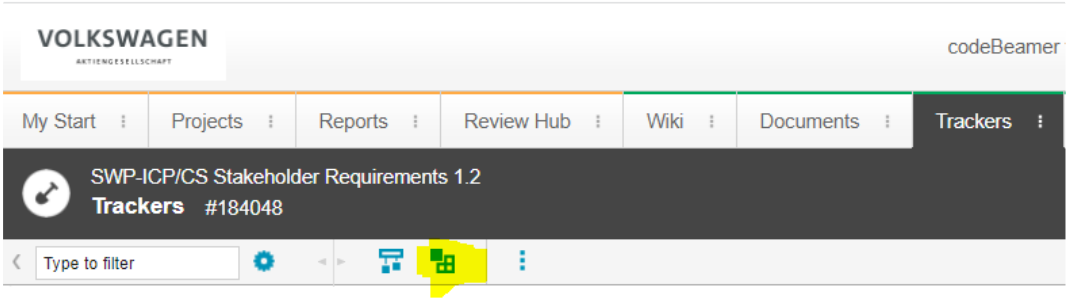
13.6.1 How to extract a list of security relevant component from SRS

Code Beamer > Trackers > Tracing Logging Recording > SRS > SW_PLAT_REQ (multiple) > SWAD_REQ (Component Requirement) (multiple) > Manually look in Bitbucket Repo for the specific requirements.

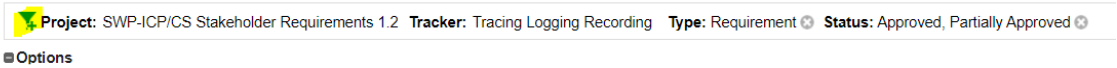
Potential possible procedure:

13.6.1.1 For 1.2

1. In CodeBeamer, go to the Tab Tracker / Traceability Browser



2. In the inital filter, filter for following things. To add more attributes, click this



   This initial filter will give us the Stakeholder Requirements for the specific component we request.

   a. **Projects**: SWP-ICP/CS Stakeholder Requirements 1.2
   b. **Tracker**: Choose the project/component you want to filter for. Here we choose Tracing Logging Recording (TLR)
   c. **Type**: Requirement
   d. **Status**: Approved, Partially Approved
3. To filter for Platform Requirement (PLAT), we add a Level 1 filter with the following attributes:
   a. Projects: E3_SW_Platform 1.2
   b. Tracker: SW Platform Requirements
4. To filter for SW Architectural Design Requirements (SWAD), we add a Level 2 filter with the following attributes:

   a. Projects: E3_SW_Platform 1.2
   b. Tracker: SW Architectural Design Requirements
5. In the end, we have a list of SWAD requirements for the specific requested component. This list can be exported to Excel via Export to Office
6. To create a list of functions/interfaces/classes to fuzz, look in the repository/code of the component e.g e3_comp_tlrflightrecorder[133] for occurences of the SWAD requirement such as *req-tlr-flightrecorder-security-logfile-encryption*. The result[134]will show you all the classes and functions which needs to be fuzzed.
   In this case, the list of functions/classes to fuzz would be:
   a. src/FlightRecorder/FileWriter/FileWriter.hpp - Class FileWriter and all its functions
      i. Constructor FileWriter
      ii. writeDataToFile
      iii. writeDataToArchive
      iv. addFileToArchive
      v. closeArchive
      vi. finalize
      vii. isOpen
      viii. removeFile
      ix. fileExists
7. Repeat step 6 for all the SWAD requirements and you will have a list of functions/interfaces/classes to fuzz.

133 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_tlrflightrecorder/browse
134 https://devstack.vwgroup.com/bitbucket/plugins/servlet/search?q=project%3AE3COMP%20repo%3Ae3_comp_tlrflightrecorder%20req-tlr-flightrecorder-security-logfile-encryption

14 CI-Fuzz List of Onboarding Teams

14.1 CI Fuzz on-boarding sessions and components status

- Sync sessions: introduction of the component
- Security team prepares fuzzing feature branch for component
- Part 1: introduction feature branch, setup of fuzzing container on component dev's workspace
- Part 2: workshop to tackle more complex functionality

TODO

OUTDATED: Fuzzing code hasn't been updated for a while (meaning, no changes from develop branch has been applied)

NOT DONE

IN PROGRESS

DONE

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**
1	E3-92646[135] - [DAST] Integrate two further teams into CIFuzz DONE VWOS-15789[136] - Support Team Skynet in fuzzing - PI24.4 DONE VWOS-16421[137] - Fuzzing activities Skynet in fuzzing DONE	Skynet[138] State Management 1.2 **06 Jan 2025** • Confluence Page: • https://devstack.vwgroup.com/confluence/display/E3SPSTM/Team+Skynet	e3_comp_socks_proxy[139]- Reverse Socks Proxy • SocksifiedConnection	**06 Jan 2025** • PO: Peterka, Steve (T5-PC) Mail: steve.peterka@cariad.techno[140]logy PO Deputy: • SM: Peterka, Steve (T5-PC) Mail: steve.peterka@cariad.techno[141]logy SM Deputy: • Dev (Test Manager): Simut, Pavel (EXTERN: Bertrandt) Mail: extern.pavel.simut@cariad.techno[142]logy Dev (Test Manager) Deputy: • Dev (Software architect): Dev (Software architect) Deputy:	Socket mock poc ready 17 Aug 2023 Sent reminder to reserve min. 5 SP for PI 23.4 19 Jun 2024 Reminder sent to reserve ca. 10 SP for PI 24.3 TODO Update to new build system OUTDATED IN PROGRESS **06 Jan 2025** • PoC (in creation), Hands on session	20220829 Skynet Onboarding Call.pdf[152]

at=refs%2Fheads%2Fcustom%2Fcifuzz

140 mailto:steve.peterka@cariad.techno

141 mailto:steve.peterka@cariad.techno

142 mailto:extern.pavel.simut@cariad.techno

152 https://devstack.vwgroup.com/confluence/download/attachments/925873858/20220829%20Skynet%20Onboarding%20Call.pdf?api=v2&modificationDate=1671115257788&version=1

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**
		• Phone Book: • https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=736994446		• Dev (Fuzzing responsible): Simut, Pavel (EXTERN: Bertrandt) Mail: extern.pavel.simut@cariad.techno[143]logy **OLD Entry (Before 📅 06 Jan 2025):** PO: yannick.volkenborn@volkswagen-infotainment.com[144] SM: michael.stens@volkswagen-infotainment.com[145] Reichelt, Sebastian (VWIFG-TE/S22) <sebastian.reichelt@volkswagen-infotainment.com[146]> Ulrich, Robin (VWIF G-TS/32) / DEV <robin.ulrich@volkswagen-infotainment.com[147]> Kirks, Steffen (VWIF G-TS/43) <steffen.kirks@volkswagen-	• → Awaiting response from Simut, Pavel (EXTERN: Bertrandt) regarding building fuzz test. • Task: • ☑ VWOS-16421[150] - Fuzzing activities Skynet in fuzzing DONE	

143 mailto:extern.pavel.simut@cariad.techno
144 mailto:yannick.volkenborn@volkswagen-infotainment.com
145 mailto:michael.stens@volkswagen-infotainment.com
146 http://volkswagen-infotainment.com
147 http://volkswagen-infotainment.com
150 https://devstack.vwgroup.com/jira/browse/VWOS-16421?src=confmacro

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**
				infotainment.com[148] > Reddy, Kaushik Jayaraja (VWIF G-TS/31) <kaushik.jayaraja.reddy@volkswagen-infotainment.com[149] >	• Kanban board Task: VWOS-16421[151] - Fuzzing activities Skynet in fuzzing DONE • Team intern: Ticket not found (2025-01-15)	

148 http://volkswagen-infotainment.com
149 http://volkswagen-infotainment.com
151 https://devstack.vwgroup.com/jira/browse/VWOS-16421?src=confmacro

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**
2	VWOS-11935[153] - Pilot risk driven fuzzing carried out with OTA Master DONE E3-92646[154] - DONE	TeamUpdateCore[158] (OTA Master[159]) Over the Air Update 1.2 **06 Jan 2025** • Confluence Page: • https://devstack.vwgroup.com/confluence/display/E3OTA	e3_comp_otam_common[161] • common/Utilities.hpp • helpers/CSProxyCommunicationHelper::processGET • helpers/CSProxyCommunicationHelper:processPOST • vwg::otamaster::helpers::OTACommunicationCommHelper::getServiceListUrl e3_comp_otam_tm[162] • ImmobilizerSpecialFunctionLoginSubSystem.hpp • AddressConverterV6.hpp e3_comp_otam_vsm[163] • n/A	**06 Jan 2025** • PO: Heininger, Tobias (T5-PH) Mail: tobias.heininger@cariad.techno[165]logy, PO Deputy: Zalilova, Evgeniia (EXTERN: LUX) Mail: evgeniya.pushkina@dxc.com[166] • SM: Krishnan Shanmugam, Sharmesh (EXTERN: LUX) Mail: s.krishnanshanmugam@dxc.com[167] SM Deputy: Krishnan Shanmugam, Sharmesh (EXTERN: LUX) Mail: s.krishnanshanmugam@d	Viktor Maslennikov is out of capacity for PI 24.1 19 Jun 2024 Reminder sent to reserve ca. 10 SP for PI 24.3 TODO Update all e3_comp_otam_tm[176] and e3_comp_otm_vsm[177] to the current build process TODO Write further fuzz tests (See list of functions to fuzz for UpdateCore[178]) OUTDATED TODO **06 Jan 2025**	20220819 OTA Master Sync regarding CI-Fuzz.pdf[183] 2023.10.26 Sync OTA Master Fuzzing 1.2 (see page 244) 2023.11.14 Hands-on OTA Master Fuzzing 1.2 (see page 246)

162 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_otam_tm/browse?at=custom%2Fcifuzz
163 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_otam_vsm/browse?at=refs%2Fheads%2Fcustom%2Fcifuzz
165 mailto:tobias.heininger@cariad.techno
166 mailto:evgeniya.pushkina@dxc.com
167 mailto:s.krishnanshanmugam@dxc.com
176 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_otam_tm/browse?at=custom%2Fcifuzz
177 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_otam_vsm/browse?at=refs%2Fheads%2Fcustom%2Fcifuzz
178 https://devstack.vwgroup.com/confluence/pages/viewpage.action?spaceKey=E3SPVMAN&title=%5B%21DEV-DB%21%5D+Fuzzing+List
183 https://devstack.vwgroup.com/confluence/download/attachments/925873858/20220819%20OTA%20Master%20Sync%20regarding%20CI-Fuzz.pdf?api=v2&modificationDate=1671115211860&version=1

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**

	E3-131808[155] - [TeamUC][CL8] Fuzzing DONE
	VWOS-10968[156] - Work together with OTA Master to fuzz intensively WON'T FIX
		• Phone Book:
		  • Phone Book - HCP5 SWP - Update Core (Team Update Core)[160]
			See list of functions to fuzz for UpdateCore[164]
			
			OTA-Master data flow mapping and requirements
			• Functions relevant for fuzzing (fuzzing targets) derived from dataflow and requirements (SRS)
			• 2024.12.05 Sync with OTA-Master Code to Data flow mapping and Requirements (see page 336)
				xc.com[168]
				• Dev (Test Manager): Vyshnevska, Olga (EXTERN: LUX) Mail: olga.vyshnevska@dxc.com[169]
				Dev (Test Manager) Deputy: Morozov, Dmitrii (EXTERN: LUX) Mail: dmitrii.morozov@dxc.com[170]
				• Dev (Software architect): Hendy, Mohammad (EXTERN: LUX) Mail: mohammad.hendy@dxc.com[171]
				Dev (Software architect) Deputy: Ivanov, Sergei (EXTERN: LUX) Mail: se
					• First Onboarding Session, PoC, 2nd Hands on session.
					• → This needs to be checked again with team
					  • Task:
					  • VWOS-16422[179] - [Dependency] Fuzzing activities UpdateCore DONE	

155 https://devstack.vwgroup.com/jira/browse/E3-131808?src=confmacro
156 https://devstack.vwgroup.com/jira/browse/VWOS-10968?src=confmacro
160 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/736994408/Phone+Book+-+HCP5+SWP+-+Update+Core+TeamUpdateCore
164 https://devstack.vwgroup.com/confluence/pages/viewpage.action?spaceKey=E3SPVMAN&title=%5B%21DEV-DB%21%5D+Fuzzing+List
168 mailto:s.krishnanshanmugam@dxc.com
169 mailto:olga.vyshnevska@dxc.com
170 mailto:dmitrii.morozov@dxc.com
171 mailto:mohammad.hendy@dxc.com
179 https://devstack.vwgroup.com/jira/browse/VWOS-16422?src=confmacro

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**
	VWOS-16725[157] - Support UpdateCore in fuzzing activities DONE			rgei.ivanov4@dxc.com[172] • Dev (Fuzzing responsible): Maslennikov, Viktor (EXTERN: LUX) Mail: viktor.maslennikov2@dxc.com[173] **OLD Entry (Before 06 Jan 2025):** • PO: tobias.heininger@cariad.techno[174]logy • SM: s.krishnanshanmugam@dxc.com[175] • Eiser, Daniel • Alexander Vaksman • Abramov, Mikhail (DXC Luxoft) / Architect • Maslennikov, Viktor (DXC Luxoft) => Fuzzing responsible	• VWOS-16725[180] - Support UpdateCore in fuzzing activities DONE	

157 https://devstack.vwgroup.com/jira/browse/VWOS-16725?src=confmacro
172 mailto:sergei.ivanov4@dxc.com
173 mailto:viktor.maslennikov2@dxc.com
174 mailto:tobias.heininger@cariad.techno
175 mailto:s.krishnanshanmugam@dxc.com
180 https://devstack.vwgroup.com/jira/browse/VWOS-16725?src=confmacro

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**
					• Kanban board Task: ☑ VWOS-16725[181] - Support UpdateCore in fuzzing activities DONE •	

181 https://devstack.vwgroup.com/jira/browse/VWOS-16725?src=confmacro

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**
					• Team internal Task: • E3-131808[182] - [Team UC][CL8] Fuzzing DONE	

182 https://devstack.vwgroup.com/jira/browse/E3-131808?src=confmacro

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**
3	☑ VWOS-10796[184] - Support LUM to implement fuzzing extensively WON'T FIX ☑ E3-106680[185] - DONE	LUM[186] Local Update Manager 1.2	e3_comp_lum[187] • FileProcessing/Config/ConfigParser::loadConfig • Manifest/ManifestReader::parse* • minfw/Utils:* • FileProcessing/Persistency/PersistencyParser::parse • minfw/FileProc/TomlParser::parseFile • test_download_service.cpp mock receive function with gtest	Bukovsky, Matthias / Dev Hoepfner, Marcus (T1-23) (PO) <marcus.hoepfner@cariad.technology>	19 Jun 2024 Reminder sent to reserve ca. 10 SP for PI 24.3 TODO Update to the current build process OUTDATED	20221202 LUM Onboarding 1.pdf[188] LUM Fuzzing Candidates[189] 2023.07.14 Fuzzing Activity for LUM (see page 224)

184 https://devstack.vwgroup.com/jira/browse/VWOS-10796?src=confmacro
185 https://devstack.vwgroup.com/jira/browse/E3-106680?src=confmacro
186 https://devstack.vwgroup.com/confluence/x/UqTtKw
187 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_lum/browse?at=refs%2Fheads%2Fcustom%2Fcifuzz
188 https://devstack.vwgroup.com/confluence/download/attachments/925873858/20221202%20LUM%20Onboarding%201.pdf?api=v2&modificationDate=1671115383430&version=1
189 https://devstack.vwgroup.com/confluence/spaces/E3SPPIF/pages/1462682252/LUM+Fuzzing+Candidates

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**
4	☑ E3-102202[190] - Integrate ICP Team Adaptive DONE	Adaptive[191] Adaptive Stack (EB) 1.2 **06 Jan 2025** • Confluence Page: • https://devstack.vwgroup.com/confluence/display/E3SPT2	e3_comp_commanifestparser[193] • eb::cmp::CommManifestParser::getSomeIPData • eb::cmp::CommManifestParser::parseManifest • eb::cmp::CommManifestParser::getSomeipMethodID	**06 Jan 2025** • PO: Peterka, Steve (T5-PC) Mail: , Deputy: ??? Sobocinski, Maciej (EXTERN: EB) Mail: maciej.sobocinski@elektrobit.com[194] PO Deputy: • SM: Sobocinski, Maciej (EXTERN: EB) Mail: maciej.sobocinski@elektrobit.com[195] SM Deputy: • Dev (Test Manager): Dev (Test Manager) Deputy: • Dev (Software architect): Dev (Software architect) Deputy:	Team will contact us in case of help needed 17 Aug 2023 Sent reminder to reserve min. 5 SP for PI 23.4 19 Jun 2024 Reminder sent to reserve ca. 10 SP for PI 24.3 **IN PROGRESS** Sync session: • Fuzzing for this PI (24.4) planned. Fuzzing targets this PI indentified and work on fuzzing is ongoing. Team will contact us in case support is needed. **IN PROGRESS** **06 Jan 2025** • Some support needed. About 70% targets for testing created	20221207 Onboarding Team Adaptive.pdf[204] 26042023 [Continuation] CI-Fuzz Hands-on Session Team Adaptive Notes.pdf[205] Muckenschnabl, David Jose (EXTERN: ITK) Sync session: 2024.11.13 Sync with Team Adaptive regarding fuzzing status (see page 328)

194 mailto:maciej.sobocinski@elektrobit.com
195 mailto:maciej.sobocinski@elektrobit.com
204 https://devstack.vwgroup.com/confluence/download/attachments/925873858/20221207%20Onboarding%20Team%20Adaptive.pdf?api=v2&modificationDate=1671115454497&version=1
205 https://devstack.vwgroup.com/confluence/download/attachments/1314972588/26042023%20%5BContinuation%5D%20CI-Fuzz%20Hands-on%20Session%20Team%20Adaptive%20Notes.pdf?api=v2&modificationDate=1682503203174&version=1

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**

		• Phone Book:
		  • Phone Book - HCP5 SWP - Adaptive[192]
				• Dev (Fuzzing responsible): Isokaeaentae, Johanna (EXTERN: EB) Mail: johanna.isokaeaentae@elektrobit.com[196], ??? Jiri (Digiteq Automotive) Mail: jiri.dostal@digiteqautomotive.com[197]
				
				OLD Entry (Before 📅 06 Jan 2025):
				People:
				• Peterka, Steve **Function Owner**
				  • steve.peterka@cariad.techno[198]**logy**
				• PO: steve.peterka@cariad.techno[199]logy
				• SM: maciej.soboc
					• Probably some support is needed for fuzzing in the pipeline
					  • Task:
					  • ☑ VWOS-16414[201] - [Dependency] Fuzzing activities Team Adaptive DONE	

192 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/736994044/Phone+Book+-+HCP5+SWP+-+Adaptive
196 mailto:johanna.isokaeaentae@elektrobit.com
197 mailto:jiri.dostal@digiteqautomotive.com
198 mailto:steve.peterka@cariad.techno
199 mailto:steve.peterka@cariad.techno
201 https://devstack.vwgroup.com/jira/browse/VWOS-16414?src=confmacro

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**
				inski@elektrobit.com[200]	• Kanban board Task: VWOS-16712[202] - Support Team Adaptive - PI25.1 DONE •	

200 mailto:maciej.sobocinski@elektrobit.com
202 https://devstack.vwgroup.com/jira/browse/VWOS-16712?src=confmacro

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**
					• Team internal: • E3-158131[203] - [Team Adaptive] Fuzzing activities DONE	

203 https://devstack.vwgroup.com/jira/browse/E3-158131?src=confmacro

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**

5	☑ VWOS-10795[206] - Support TLR to implement fuzzing extensively DONE
	☑ VWOS-7423[207] - DONE
		BOS[208] **(previously TLR)**
		Tracing logging Recording 1.2
		Team TLR Archive[209]
			e3_comp_tlrlib[210]
			• Messages/TLRMessage.hpp
			  • vwg::e3p::tlr::TLRMessage::isTLRMessage
			  • vwg::e3p::tlr::TLRMessage::fromBuffer
			  • vwg::e3p::tlr::TLRMessage::getHeaderData
			  • vwg::e3p::tlr::TLRMessage::getMessageData
				People:
				• DoraisWamy, Suganya **PO**
				  • suganya.doraiswamy@cariad.techno[211] **logy (contact person)**
				• Schlünder, Felix **SM**
				  • felix.schluender@cariad.techno[212] **logy**
				• Salich, Sebastian **FO**
				  • sebastian.salich@cariad.techno[213] **logy**
				• fabian.schmon@cariad.techno[214] **logy**
					Sent initial mail to team 22 Mar 2023
					Sent reminder 30 Mar 2023
					Sent appointment for hands-on session with Bernd in CC 08 May 2023 13:00
					TLR is not responding. Communicate to Alexander Aurich on 15 May 2023 and waiting for his response
					12 Jun 2023 Bernd has sent a reminder to Alexander Aurich
					13 Jun 2023 Sent out reminder to team to plan 1-2 story point
					Plan onboarding session for PI 23.3-3rd Sprint First week and send to Doraiswamy, Suganya (T2-EE)
					Invitation sent out for 08 Aug 2023 Done
						2023.08.08 Sync and Hands-On session TLR (see page 232)
						2023.08.21 3rd Hands-On session TLR (see page 237)

206 https://devstack.vwgroup.com/jira/browse/VWOS-10795?src=confmacro
207 https://devstack.vwgroup.com/jira/browse/VWOS-7423?src=confmacro
208 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=690771084
209 https://devstack.vwgroup.com/confluence/spaces/E3LOG/pages/34550075/Team+TLR+Archive
210 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_tlrlib/browse?at=refs%2Fheads%2Fcustom%2Fcifuzz
211 mailto:suganya.doraiswamy@cariad.techno
212 mailto:felix.schluender@cariad.techno
213 mailto:sebastian.salich@cariad.techno
214 mailto:fabian.schmon@cariad.techno

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**
				• **'Desai, Abhijeet (T1-23)' <abhijeet.desai@cariad.technology>** • Jakub.Kuzel@digiteqautomotive.com[215] • ext.jelena.vujakovic@digiteqautomotive.com[216] • saad.ouassil.allak@cariad.technology[217]	Hands-on session invitation for 21 Aug 2023 to onboard the new build system 17 Aug 2023 Sent reminder to reserve min. 5 SP for PI 23.4 19 Jun 2024 Reminder sent to reserve ca. 10 SP for PI 24.3 **TODO** Update to the current build process **OUTDATED**	

215 mailto:Jakub.Kuzel@digiteqautomotive.com
216 mailto:ext.jelena.vujakovic@digiteqautomotive.com
217 mailto:saad.ouassil.allak@cariad.technology

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**
6	NO CAPA DEPRECATED ☑ VWOS-9150[218] - Integrate ICP Team Diag Master DONE ☑ VWOS-12210[219] - Interg rate Diag Master into fuzzing DONE	DiagMaster[220] Service orientierte Diagnose 1.2	e3_comp_diagupdate[221] • DUSSkeleton::createUpdateSession[222] NO CAPA • VehicleManifestParser.cpp[223] DONE e3_diagmaster_libofe[224] • DiagComChannel.cpp[225] • DiagLibraryResult.cpp[226] • FlashSessionExecutor.cpp[227]	**People:** • Grill, Micha **PO** • micha.grill@cariad.techno[233] **logy** • SM: tetiana.pylypchuk@dxc.com[234] benjamin.david@dxc.com[235] • General mail • LuxE3TeamDiagMaster@dxc.com[236]	10 Jul 2023 Bug[241] found in VehicleManifest. Starting with next PoC 09 Aug 2023 Sent Hand-on Meeting invitation. Team Acepted for meeting on 11 Aug 2023 . Team declined since most of the team is in vacation Sent out another invitation for 22 Aug 2023 17 Aug 2023 Sent reminder to reserve min. 5 SP for PI 23.4	Muckenschnabl, David Jose (EXTERN: ITK) Notes 21 Apr 2023 :Diagmaster_Protocol.pdf[242] 2023.08.29 1st Hands-On session DiagMaster (see page 240)

222 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_diagupdate/browse/src/diagupdate/skeleton?at=refs%2Fheads%2Fdevelop
223 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_diagupdate/browse/src/diagupdate/manifest/VehicleManifestParser.cpp?at=refs%2Fheads%2Fdevelop
224 https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_diagmaster_libofe/browse?at=refs%2Fheads%2Fcustom%2Fcifuzz
225 https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_diagmaster_libofe/browse/tests/fuzzing/fuzz_DiagComChannel.cpp?at=refs%2Fheads%2Fcustom%2Fcifuzz
226 https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_diagmaster_libofe/browse/tests/fuzzing/fuzz_DiagLibraryResult.cpp?at=refs%2Fheads%2Fcustom%2Fcifuzz
227 https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_diagmaster_libofe/browse/tests/fuzzing/fuzz_FlashSessionExecutor.cpp?at=refs%2Fheads%2Fcustom%2Fcifuzz
233 mailto:micha.grill@cariad.techno
234 mailto:tetiana.pylypchuk@dxc.com
235 mailto:benjamin.david@dxc.com
236 mailto:LuxE3TeamDiagMaster@dxc.com
241 https://devstack.vwgroup.com/jira/browse/E3-122187
242 https://devstack.vwgroup.com/confluence/download/attachments/1314972588/Diagmaster_Protocol.pdf?api=v2&modificationDate=1682078891346&version=1

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**

			• FunctionHandleDataBlock.cpp[228]
			• FunctionHandleSegment.cpp[229]
			• FunctionRoutingActivation.cpp[230]
			• FunctionUnlockECU.cpp[231]
			• KonzernErrorTypes.cpp[232]
				• **Developer**
				  • anton.soldatkin@dxc.com[237]
				  • gerardo.cuozzo@dxc.com[238]
				  • k.kallappasoraganvi@dxc.com[239]
				  • o.shashkevych@dxc.com[240]
					29 Aug 2023
					1st hands-on session carried out
					TODO Might have to update APfuzzing.cmake version
					OUTDATED	

228 https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_diagmaster_libofe/browse/tests/fuzzing/fuzz_FunctionHandleDataBlock.cpp?at=custom%2Fcifuzz
229 https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_diagmaster_libofe/browse/tests/fuzzing/fuzz_FunctionHandleSegment.cpp?at=custom%2Fcifuzz
230 https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_diagmaster_libofe/browse/tests/fuzzing/fuzz_FunctionRoutingActivation.cpp?at=custom%2Fcifuzz
231 https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_diagmaster_libofe/browse/tests/fuzzing/fuzz_FunctionUnlockECU.cpp?at=custom%2Fcifuzz
232 https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_diagmaster_libofe/browse/tests/fuzzing/fuzz_KonzernErrorTypes.cpp?at=custom%2Fcifuzz
237 mailto:anton.soldatkin@dxc.com
238 mailto:gerardo.cuozzo@dxc.com
239 mailto:k.kallappasoraganvi@dxc.com
240 mailto:o.shashkevych@dxc.com

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**

7	☑ VWOS-9224[243] - Integrate ICP Team SOCD -Babelfish to fuzzing DONE	SOCD - Babelfish[244] **Comet**	TODO Create POC	• PO: Zange, Kevin (T1-23) (**Product Owner**)
				  • kevin.zange@cariad.technology[245]
				• Koch, Paul (T1-23) (**Former PO**)
				  • paul.koch@cariad.technology[246]
				• SM: agnes.obermann@tagueri.com[247]
					• Sync session set for 11 Jul 2023 . Waiting for the team response.
					17 Aug 2023 Sent reminder to reserve min. 5 SP for PI 23.4
					19 Jun 2024 Reminder sent to reserve ca. 10 SP for PI 24.3
					TODO No POC has been created
					OUTDATED
						Dang, Quang Anh (EXTERN: ITK)
						
						2023.07.11 Onboarding session SOCD-Babelfish / Comet (see page 215)

8		**DiagNinjas**	• diag_procedure_interpreter_service (prio) • diag_routine_service • diag_data_service • diag_access_manager	• **Responsible for the whole DiagMaster** Meyerhoefer, Marcus, Dr. (T5-PH)[248] • Responsible for DiagNinjas muhammed.koese@caria	• Synced with Marcus Meyerhoefer TODO First sync pending to determine repo and code to fuzz	2024.10.07 Discuss E3-122194 - DiagMaster Risk (see page 324)

243 https://devstack.vwgroup.com/jira/browse/VWOS-9224?src=confmacro
244 https://devstack.vwgroup.com/confluence/display/E3SB/Team+Babelfisch
245 mailto:kevin.zange@cariad.techno
246 mailto:paul.koch@cariad.techno
247 mailto:agnes.obermann@tagueri.com
248 https://devstack.vwgroup.com/confluence/display/~foppsdd

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**
				d.techno[249]logy • Responsible for Diagnosis: michael.eberspaecher@cariad.techno[250]logy		

249 mailto:muhammed.koese@cariad.techno
250 mailto:michael.eberspaecher@cariad.techno

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**

9	VWOS-16718[251] - [Sprint 2] Support Team Code Spiders - PI25.1 DONE	**CodeSpiders (DiagMaster CodeSpiders)**
		
		06 Jan 2025
		• Confluence Page:
		  • Team DiagMaster[252]
		  • DiagMaster Component's Contacts[253]
		• Phone Book:
		  • DEPRECATED
			https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_diagevent/browse	**06 Jan 2025**
				• PO: Bruno, Vittorio (T5-PH) Mail: vittorio.bruno@cariad.techno[255]logy PO Deputy: ??? Hruda, Petr (PECZ-OS) Mail: petr.hruda@porsche-engineering.cz[256]
				• SM: Glowacki, Aleksander (PECZ-OS) Mail: aleksander.glowacki@porsche-engineering.cz[257] SM Deputy:
				• Dev (Test Manager): Herman, Adam (PECZ-OS) Mail: adam.herman@porsche-engineering.cz[258] Dev (Test Manager) Deputy:
					TODO
					06 Jan 2025
					• First Onboarding Session, PoC, Hands on session.
					• → This needs to be checked again with team
					  • Task:
					  • VWOS-16525[259] - [Dependency] Fuzzing activities Code Spiders DONE	

251 https://devstack.vwgroup.com/jira/browse/VWOS-16718?src=confmacro
252 https://devstack.vwgroup.com/confluence/spaces/E3DIAG/pages/34549963/Team+DiagMaster
253 https://devstack.vwgroup.com/confluence/spaces/E3DIAG/pages/1892095250/DiagMaster+Component+s+Contacts
255 mailto:vittorio.bruno@cariad.techno
256 mailto:petr.hruda@porsche-engineering.cz
257 mailto:aleksander.glowacki@porsche-engineering.cz
258 mailto:adam.herman@porsche-engineering.cz
259 https://devstack.vwgroup.com/jira/browse/VWOS-16525?src=confmacro

Index	VW.os Core					
	Ticket	Team / component(s)	Component with POC	Contact	Status	Meeting minutes
		Phone Book - HCP5 SWP - Code Spiders (Team DM_Code Spiders)[254]		• Dev (Software architect): Dev (Software architect) Deputy: • Dev (Fuzzing responsible):	• Kanban board Task: • VWOS-16718[260] - [Sprint 2] Support Team Code Spiders - PI25.1 DONE	

254 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1941836295/DEPRECATED+Phone+Book+-+HCP5+SWP+-+CodeSpiders+TeamDM_CodeSpiders
260 https://devstack.vwgroup.com/jira/browse/VWOS-16718?src=confmacro

Index	VW.os Core					
	Ticket	**Team / component(s)**	**Component with POC**	**Contact**	**Status**	**Meeting minutes**
					• Team internal Task: • ☑ E3-158314[261] - Implement Fuzz-Tests in DEVS TODO • Agreed for and prepare meeting on 17 Feb 2025	

261 https://devstack.vwgroup.com/jira/browse/E3-158314?src=confmacro

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes

1	E3APP007-1244
	E3APP007-1469
	E3APP007-1168
		Remote repair[262] **(high)**
		Remote repair 1.2
		06 Jan 2025
		• Confluence Page:
		  • Team Remote Repair (see page 158)
		  • Team CodeBusters[263]
		• Phone Book:
		  • ???Team Co
			e3_app_remote_repair[265]
			• utils/Timestamp.hpp
			• utils/FileHandler.hpp::readConfigFile
			• vwg::e3p::remote_repair::RRCommunicator::assemblyStatus
			• vwg::e3p::remote_repair::RRCommunicator::changeSessionType
			• vwg::e3p::remote_repair::RRCommunicator::closeSession
			• vwg::e3p::remote_repair::RRCommunicator::diagData
			• vwg::e3p::remote_repair::RRCommunicator::diagEntry
			• vwg::e3p::remote_repair::RRCommunicator::diagEvents
			• vwg::e3p::remote_repair::RRCommunicator::diagProcedureInterpreter
			• vwg::e3p::remote_repair::RRCommunicator::diagRoutines
			• vwg::e3p::remote_repair::RRCommunicator::getSessionStatus
				06 Jan 2025
				• PO: Hoche, Matthias (T5-PF) Mail: matthias.hoche@cariad.techno[266]logy PO Deputy:
				• SM: SM Deputy:
				• Dev (Test Manager): Komenaka, Bjoern (EXTERN: ITPower) Mail: extern.bjoern.komenaka@cariad.techno[267]logy, Bachsan, Fachrun (EXTERN: ITPower), Mail: extern.fachrun.bachsan@cariad.techno[268]logy, Pan, Su (T5-PF) Mail: su.pan@cariad.techno[269]logy Dev (Test Manager) Deputy:
				• Dev (Software architect): Dev (Software architect) Deputy:
					Wait for SRS to extract list of fuzz functions
					17 Aug 2023 Sent reminder to reserve min. 5 SP for PI 23.4
					Next session is planned for 12 Mar 2024
					DONE All fuzz targets (recommended by SRA) have been created and executed
					IN PROGRESS
					06 Jan 2025
					• Support is needed. They plan for creating fuzzing tests for CL8
					  • Task:
						20221206 Remote Repair.pdf[277]
						Tracker: RR Security Requirements 1.2 (vwgroup.com)[278]
						2024.01.12 Sync fuzzing topic team Remote Repair (1. Part) (see page 257)
						Dang, Quang Anh (EXTERN: ITK)

267 mailto:extern.bjoern.komenaka@cariad.techno
268 mailto:extern.fachrun.bachsan@cariad.techno
269 mailto:su.pan@cariad.techno
277 https://devstack.vwgroup.com/confluence/download/attachments/925873858/20221206%20Remote%20Repair.pdf?api=v2&modificationDate=1671115403063&version=1
278 https://cb.vwgroup.com/cb/tracker/66888802

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes

		deBusters[264]	• vwg::e3p::remote_repair::RRCommunicator::openSession
			• vwg::e3p::remote_repair::RRCommunicator::resetDevice
			• vwg::e3p::remote_repair::utils::Configuration::parseConfig
			• vwg::e3p::remote_repair::proxy::CoreServicesGreyCommunicator::onServiceListResponse
			• vwg::e3p::remote_repair::proxy::HMICommunicator::onEvent
			• vwg::e3p::remote_repair::proxy::HMICommunicator::onStateChange
			• vwg::e3p::remote_repair::proxy::OtaVSMCommunicator::onPreconditionStartResponseAsync
			• vwg::e3p::remote_repair::proxy::OtaVSMCommunicator::onPreconditionStopResponseAsync
				• Dev (Fuzzing responsible):
				
				Old Entry (Before 📅 06 Jan 2025):
				• Gerber, Jürgen **PO**
				  • **juergen.gerber@**cetitec.com[270]
				• Vrbanec, Vjeran **Dev**
				  • vjeran.vrbanec@cetitec.com[271]
				• Bernard Gvardijan **SM**
				  • bernard.gvardijan@cetitec.com[272]
				• Ante Buterin **Developer responsible for fuzzing**
				  • ante.buterin@cetitec.com[273]
					• ☑ VWOS-16416[274] - [Dependency] Fuzzing activities Remote Repair DONE	

264 https://devstack.vwgroup.com/confluence/spaces/E3APP017/pages/207985905/Team+CodeBusters
270 http://cetitec.com
271 mailto:vjeran.vrbanec@cetitec.com
272 mailto:bernard.gvardijan@cetitec.com
273 mailto:ante.buterin@cetitec.com
274 https://devstack.vwgroup.com/jira/browse/VWOS-16416?src=confmacro

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes
					• Kanban board Task: VWOS-16720[275] - [Sprint 2] Support Team RemoteRepair - PI25.1 DONE •	

275 https://devstack.vwgroup.com/jira/browse/VWOS-16720?src=confmacro

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes
					• Team internal Task: E3-160795[276] - [RemoteRepair] Fuzzing support activities buffer WONT FIX •	

276 https://devstack.vwgroup.com/jira/browse/E3-160795?src=confmacro

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes

2	☑ VWOS-11116[279] - Support RPC+ in fuzz testing DONE	RPC+[282] **(lowest)**
		RemoteProductionClient 1.2
			e3_app_remoteproductionclient[283]
			• src/utilities
			  • base64.cpp
			  • hash.cpp
			  • common_converters.cpp
			• src/baseline_update/engine/state_machine/baseline_state_download.cpp calculateDownloadProgress
				• Hithesh Sreedhara **PO**
				  • hithesh.sreedhara@dxc.com[284]
				• Team members / related
				  • s.krishnanshanmugam@dxc.com[285]
				  • carina.mathae@cariad.technology[286]
					Prio set to lowest after sync with Daniel Neeb
					Sent a second contact mail for a status report on fuzzing. 📅 08 Apr 2024
					Invitation sent 📅 05 Apr 2023 waiting for response
					Appointment for 📅 05 May 2023 at 14:05. Onboarding session was carried out.
					📅 17 Aug 2023 Sent reminder to reserve min. 5 SP for PI 23.4
					Re-sync took place 📅 30 Jan 2024
					The team is trying to fuzz independently
					TODO Might have to update APfuzzing.cmake version
					OUTDATED
						Dang, Quang Anh (EXTERN: ITK)
						05052023 Ci-Fuzz Onboarding Hands-On session Team RPC+.pdf[287]
						2024.02.09 Sync fuzzing topic RPC+ regarding effort for fuzzing (see page 269)
						2024.01.30 Sync fuzzing topic RPC+ (see page 264)

279 https://devstack.vwgroup.com/jira/browse/VWOS-11116?src=confmacro
282 https://devstack.vwgroup.com/confluence/x/3iv8L
283 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remoteproductionclient/browse?at=custom%2Fcifuzz
284 mailto:hithesh.sreedhara@dxc.com
285 mailto:s.krishnanshanmugam@dxc.com
286 mailto:carina.mathae@cariad.techno
287 https://devstack.vwgroup.com/confluence/download/attachments/1314972588/05052023%20Ci-Fuzz%20Onboarding%20Hands-On%20session%20Team%20RPC%2B.pdf?api=v2&modificationDate=1683293412462&version=1

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes
	E3-110601[280] - DONE E3APP012-4921[281] - [RPC+] Establish CI-Fuzz Testing for all external components DONE					

280 https://devstack.vwgroup.com/jira/browse/E3-110601?src=confmacro
281 https://devstack.vwgroup.com/jira/browse/E3APP012-4921?src=confmacro

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes

3	2.0 STOP
	E3-118119[288] - Integrate BSrv Team WUM
	DONE
		WUM [289]
		Wake Up Monitor 2.0
			e3_app_wakeup_monitor[290]
			• 2.0 STOP Marshaler[291]
			• 2.0 STOP MeasuredValues[292]
			• 2.0 STOP ClientHandler.cpp[293]
				• Vladislav, Jasek **PO**
				  • Vladislav.Jasek@digiteqautomotive.com[294]
				• Vinecka, Lucie **SM**
				  • Lucie.Vinecka@digiteqautomotive.com[295]
					• On 14 Jun 2023 Sent a mail to Lucie.Vinecka@digiteqautomotive.com[296] as a reminder to book 1 or 2 story points for next PI
					2.0 STOP No Fuzz targets have been created
						Muckenschnabl, David Jose (EXTERN: ITK)
						07062023 CiFuzz Sync Session Team WUM.pdf[297]

288 https://devstack.vwgroup.com/jira/browse/E3-118119?src=confmacro
289 https://devstack.vwgroup.com/confluence/x/Cyv8L
290 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_wakeup_monitor/browse
291 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_wakeup_monitor/browse/src/Modules/Marshaler
292 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_wakeup_monitor/browse/src/Handlers/DMExtensions/MeasuredValues
293 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_wakeup_monitor/browse/src/Handlers/Clients/ClientHandler.cpp
294 mailto:Vladislav.Jasek@digiteqautomotive.com
295 mailto:Lucie.Vinecka@digiteqautomotive.com
296 mailto:Lucie.Vinecka@digiteqautomotive.com
297 https://devstack.vwgroup.com/confluence/download/attachments/1314972588/07062023%20CiFuzz%20Sync%20Session%20Team%20WUM.pdf?api=v2&modificationDate=1686730682160&version=1

Index	BSrv					
	Ticket	**Team / component(s)**	**Component with POC**	**Point of contact / role**	**Status**	**Notes**

4	2.0 STOP
	☑ E3-116131[298] - Integrate BSrv Team CodeBuster DONE
		CodeBuster[299]
		Function on Demand 2.0
			e3_app_fod_sp[300]
			• src/serviceprovider/services/function_enabling/response_cache.hpp
			  • updateCache
			• function_enabling_service_validator.cpp[301] DONE
			  • validateSetFunctionState
			• function_enabling_service_mapper.cpp[302] DONE
			  • vwg::e3p::fod::sp::serv::FunctionEnablingServiceMapper::mapToExternal
			• function_enabling_service_impl.cpp[303] TODO
			NOT DONE
				• Hoche, Matthias **PO**
				  • matthias.hoche@cariad.techno[304] **logy**
				• Puskajler, Martin **SM**
				  • martinpuskajler@gmail.com[305]
					TODO Update to the current build process
					OUTDATED
						Muckenschnabl, David Jose (EXTERN: ITK)
						2023_04_23_CodeBuster_Protocol.docx[306]
						2023.07.20 Hands-on session CodeBuster
						(see page 228)

298 https://devstack.vwgroup.com/jira/browse/E3-116131?src=confmacro
299 https://devstack.vwgroup.com/confluence/x/1473L
300 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_fod_sp/browse?at=custom%2Ffuzzing
301 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_fod_sp/browse/src/serviceprovider/services/function_enabling/function_enabling_service_validator.cpp?at=refs%2Fheads%2Fdevelop
302 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_fod_sp/browse/src/serviceprovider/services/function_enabling/function_enabling_service_mapper.cpp?at=refs%2Fheads%2Fdevelop
303 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_fod_sp/browse/src/serviceprovider/services/function_enabling/function_enabling_service_impl.cpp?at=refs%2Fheads%2Fdevelop
304 mailto:matthias.hoche@cariad.techno
305 mailto:martinpuskajler@gmail.com
306 https://devstack.vwgroup.com/confluence/download/attachments/1314972588/2023_04_23_CodeBuster_Protocol.docx?api=v2&modificationDate=1682344830915&version=1

Index	BSrv					
	Ticket	**Team / component(s)**	**Component with POC**	**Point of contact / role**	**Status**	**Notes**

5	E3-118120[307] - Integrate BSrv Team Adaptive Wizards WON'T FIX
	NOT DONE
		1.1 ADA
		Adaptive Wizards [308] (LUCA)
		Environmental Data 2.0 Provider 2.0
			NOT DONE	• Yavuz, Onur **PO**
				  • onur.yavuz@cariad.techno[309]**logy**
				• Puskajler, Martin **PO&SM**
				  • martinpuskajler@gmail.com[310]
					NOT DONE
					• as EDP is no security relevant, the team chose not to do the on-boarding, we will give support in the future if needed
					• Not security relevant based on SRA
						Muckenschnabl, David Jose (EXTERN: ITK)

307 https://devstack.vwgroup.com/jira/browse/E3-118120?src=confmacro
308 https://devstack.vwgroup.com/confluence/x/e5H3L
309 mailto:onur.yavuz@cariad.techno
310 mailto:martinpuskajler@gmail.com

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes

6	2.0 STOP	UDSM[311]
		Upload Download Storage Manager (UDSM 2.0)
			e3_app_css[312] Info[313]
			/src/datacore/BucketHandler.cpp
			• BucketHandler::createBucket DONE
			/clientlibrary/CssClient.hpp
			• createObjectWithPolicy
			src/utils/InputValidator.hpp
			• css::utils::InputValidator::validateBucketName(input); DONE
			• css::utils::InputValidator::validateObjectName(input); DONE
			• css::utils::InputValidator::validateId(input); DONE
			• css::utils::InputValidator::checkEmptyString(input); DONE
			• css::utils::InputValidator::validateBucketPath(input); DONE
				• Krishnamurthy, Manmath Kumar **PO**
				  • manmath.kumar@embitel.com[314]
				• Venkatraman, Pasanna **SM**
				  • prasanna.venkatraman@embitel.com[315]
					Sync meeting set up on 11 Apr 2023
					Status: SRE was done 05 Jun 2023 . UDSM is security relevant. However, SRA is not required.
					13 Jun 2023 Sent out reminder to team to plan 1-2 story point
					Hands-on session request sent for 09 Aug 2023
					Follow up hands.on session sent for 22 Aug 2023
					2nd Hands-on carried out for 23 Aug 2023
					DONE 2.0 STOP
					OUTDATED
						Dang, Quang Anh (EXTERN: ITK)
						Sync Session Minutes & Participants
						Minutes
						• Bi presents who we are and explains agenda of the meeting
						• Bi explains how Fuzzing works and the current workflow
						• Bi proceeds to do a demo on Fuzzing

311 https://devstack.vwgroup.com/confluence/display/E3APP020/Team+UDSM
312 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_css/browse?at=refs%2Fheads%2Fdevelop
313 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/560154548/1.+Introduction+and+Goals+DELETED+SPACE+E3APP020
314 mailto:manmath.kumar@embitel.com
315 mailto:prasanna.venkatraman@embitel.com

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes
			• css::utils::InputValidator::validateObjectPath(input); DONE **/src/datacore/ObjectHandler.cpp** • ObjectHandler::writeObject 2.0 STOP • ObjectHandler::createObject 2.0 STOP • ObjectHandler::createObjectWithPolicy IN PROGRESS			• Manmath Kumar asks: How to know what we have to fuzz. • Bi answers: The SRS will state what to fuzz • The repository they use is e3_app_css[316] develop branch • The information about the project is here[317]

316 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_css/browse?at=refs%2Fheads%2Fdevelop
317 https://devstack.vwgroup.com/confluence/display/E3APP020/1.+Introduction+and+Goals

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes

						• Targets for PoCs
						  • Create bucket
						  • Create oject
						  • WriteObject
						• Mocks are available at test/mock[318]
						• For support ask Manmath or Jayasankar
						Participants
						• Hoffmann, Benjamin

318 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_css/browse/tests/mock?at=refs%2Fheads%2Fdevelop

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes
						• Thippabattini, Jayasankar • Krishnamurthy, Manmath Kumar • Prasanna Venkatraman • Quang Dang • David Jose Muckenschnabl 2023.08.09 Sync and Hands-On session UDSM (see page 234) 2023.08.23 2nd Hands-On session UDSM (see page 238)

Index	BSrv					
	Ticket	**Team / component(s)**	**Component with POC**	**Point of contact / role**	**Status**	**Notes**

7	VWOS-12051[319] - Integrate LostPilots into fuzzing DONE
	E3APP01-2441[320] - [LostPilots] Establish CI-Fuzz Testing for all external components DONE
		LostPilots[321] **(DDA/DRM?)**
		Remote Key Value Storage Service 1.2
			e3_app_lostpilots_rkvs[322]
			• src/rkvs/e3p/corbos/KeyValueStoreSkeleton.hpp
			  • add
			  • getValue
			  • remove
			  • getKeys
			  • raiseOnChangedEvent
			  • registerAddCallback
			  • unregisterAddCallback
			  • registerRemoveCallback
			  • unregisterRemoveCallback
			  • unregisterAllCallbacks
			IN PROGRESS
				• Constantin Popescu **PO**
				  • constantin.popescu@cariad.technology[323]
				• Team members / related
				  • kamyar.morgenstern@tagueri.com[324]
				  • haythem.touil@cariad.techno[325]logy
				  • extern.maik.lauschkin@cariad.technology[326] (responsible for fuzzing)
				  • Lüthen Hendrik (responsible for fuzzing)
					12 Jun 2023 1st Sync Meeting carried out. Follow session on 16 Jun 2023
					13 Jun 2023 Sent out reminder to team to plan 1-2 story point
					16 Jun 2023 Lost Pilot doesn't have a fuzztarget for differential fuzzing for us yet => Waiting for their feedback.
					19 Jul 2023 Sent a reminder to joerg.hohensohn@voi[327]kswagen.de[328] to ask the status of the POC examples functions
					23 Aug 2023 Joerg doesnt have any update. Wait a bit before sending the next reminder
					01 Feb 2024 Resynced again with Lost Pilot after restructuring TODO Might have to update APfuzzing.cmake version
						Dang, Quang Anh (EXTERN: ITK)
						2023_06_12 Lost Pilot CiFuzz Sync Meeting.pdf[329]
						20230_06_16_Architecture_Functional_Explained_Session.pdf[330]
						2024.02.01 Sync fuzzing topic Lost Pilot (see page 267)
						2024.02.13 Hands-on creating fuzz targets for Lost Pilot (see page 271)

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes
					OUTDATED	2024.02.20 Hands-on creating fuzz targets for Lost Pilot 2 (see page 275) 2024.02.23 Hands-on creating fuzz targets for Lost Pilot 3 (see page 277) 2024.03.01 Support session Lost Pilot (see page 278)
8	NO CAPA NOT DONE	Battery Proxy[331] **(high)** Battery Proxy 1.2	NO CAPA	NO CAPA	NO CAPA NOT DONE	NO CAPA

331 https://devstack.vwgroup.com/confluence/x/bID3L

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes
9	VWOS-12037[332] - Integrate Time Travelers into fuzzing DONE E3APP0004-4935[333] - [Time Master] Establish CI-Fuzz Testing TMSP DONE	Time Travelers[334] **(high)** Time Master 1.2, 2.0 • **TM Base** Classic AUTOSAR • **TM Obd** Classic AUTOSAR • **TM SP** Adaptive AUTOSAR **06 Jan 2025** **TM Base (Autosar Classic):** • Confluence Page: • Team Tim	• **TM Obd & TM Base** • e3_TM_Base[341] • e3_TM_Base/tests/fuzzing/fuzz_Time_Diff_Calculation.cpp[342] • ... see repo • IN PROGRESS e3_TM_Base/tests/fuzzing/fuzz_TM_Bas	**06 Jan 2025** **TM Base (Autosar Classic):** • PO: Bashtannik, Oleksandr (Digiteq Automotive) Mail: oleksandr.bashtannik@digiteqautomotive.com[345] • PO Deputy: • SM: Korinkova, Ludmila (Digiteq Automotive) Mail: ludmila.korinkova@digiteqautomotive.com[346] • SM Deputy: • Dev (Test Manager): • Dev (Test Manager) Deputy: • Dev (Software architect):	**TMSP** Team is working on compatibility of build script with their pipeline **TMBase & TMObd** Team has finished 2nd POC and are working on a list of targets. IN PROGRESS TODO Might have to update APfuzzing.cmake version IN PROGRESS Sync session: TM SP (Adaptive Autosar) and TM Obd (Classic Autosar) • Targets identfied and testcases about 70% done. Working on reaching 100% of the fuzzing testcases. • Waiting for using CI/CD pipeline.	Muckenschnabl, David Jose (EXTERN: ITK) 2024.01.15 Sync fuzzing topic TM SP (see page 259) 2024.01.17 Sync fuzzing topic TMBase & TMObd (see page 262) 2024.02.19 Sync fuzzing topic TM SP (see page 272) 2024.02.20 Fuzzing session with TMBase (see page 274) 2024.03.06 Sync fuzzing topic TM SP (see page 283)

346 mailto:ludmila.korinkova@digiteqautomotive.com

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes
		eTravelers[335] • Phone Book • Phone Book - HCP5 SWP - Time Travelers[336] **TM OBD (Autosar Classic):** • Confluence Page: • Team TimeTra	e_Adapter_ClientServer.cpp[343] • ... see repo • **TM SP** • e3_app_timemaster_au416/tests/fuzzing[344] • ... see repo	• Dev (Fuzzing responsible): ??? Oscadal, Marek (SKODA) Mail: Marek.Oscadal@digiteqautomotive.com[347], ??? Bohac, Marek (SKODA) Mail: marek.bohac@digiteqautomotive.com[348], Rudaljevic, ??? Tamara (EXTERN: Synchrotek) Mail: ext.Tamara.Rudaljevic@digiteqautomotive.com[349] **TM OBD (Autosar Classic):** • PO: Bashtannik, Oleksandr (Digiteq Automotive) Mail: oleksandr.bashtannik@digiteqautomotive.	• Will contact us if support is needed **TODO** **06 Jan 2025** **TM Base (Autosar Classic):** • Some support needed. About 70% targets for testing created • Task:	2024.03.08 Sync fuzzing topic TMBase & TMObd (see page 284) 2024.03.27 [TMBase & TMObd] Setting up the dev environment for fuzzing (see page 288) Sync session: 2024.11.21 Sync with Team TM SP (Adaptive Autosar) and TM Obd (Classic Autosar) regarding fuzzing status (see page 329)

+HCP5+SWP+-+Time+Travelers

343 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_tm_base/browse/tests/fuzzing/fuzz_TM_Base_Adapter_ClientServer.cpp?at=refs%2Fheads%2Ffeature%2FE3APP004-5134-tmbase-provide-support-for-fuzz-test-poc-implementation-client-server-module

344 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_timemaster_au416/browse?at=refs%2Fheads%2Ffeature%2FE3APP004-5186-tmsp-implement-fuzz-test-for-selected-interfaces

347 mailto:Marek.Oscadal@digiteqautomotive.com

348 mailto:marek.bohac@digiteqautomotive.com

349 mailto:ext.Tamara.Rudaljevic@digiteqautomotive.com

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes
		velers[337] • Phone Book • Phone Book - HCP5 SWP - Time Travelers[338] **TM SP (Autosar Adaptive):** • Confluence Page: • Team TimeTravel		comBas[350]hatannik • PO Deputy: • SM: orinkova, Ludmila (Digiteq Automotive) Mail: ludmila.korinkova@digiteqautomotive.com[351] • SM Deputy: • Dev (Test Manager): • Dev (Test Manager) Deputy: • Dev (Software architect):	☑ VWOS-16413[361] - [Dependency] Fuzzing activities TM Base/ TM OBD/ TM SP • DONE	

337 https://devstack.vwgroup.com/confluence/spaces/E3APP004/pages/120816496/Team+TimeTravelers
338 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/754419083/Phone+Book+-+HCP5+SWP+-+Time+Travelers
350 mailto:oleksandr.bashtannik@digiteqautomotive.comBas
351 mailto:ludmila.korinkova@digiteqautomotive.com
361 https://devstack.vwgroup.com/jira/browse/VWOS-16413?src=confmacro

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes
		ers[339] • Phone Book • Phone Book - HCP5 SWP - Time Travelers[340]		• Dev (Fuzzing responsible): ??? Oscadal, Marek (Digiteq Automotive) Mail: Marek.Oscadal@digiteqautomotive.com[352], ??? Dostal, Jiri (Digiteq Automotive) Mail: jiri.dostal@digiteqautomotive.com[353] **TM SP (Autosar Adaptive):** • PO: Bashtannik, Oleksandr (Digiteq Automotive) Mail: oleksandr.bashtannik@digiteqautomotive.com[354] • PO Deputy: • SM: Korinkova, Ludmila (Digiteq Automotive) Mail: ludmila.korinkova@digiteqautomotive.com[355] • SM Deputy:	• Kanban board Task: • ☑ VWOS-16723[362] - Support Team Time Travelers TM Base/ TM OBD/ TM SP in fuzzing - PI25.1 DONE	

339 https://devstack.vwgroup.com/confluence/spaces/E3APP004/pages/120816496/Team+TimeTravelers
340 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/754419083/Phone+Book+-+HCP5+SWP+-+Time+Travelers
352 mailto:Marek.Oscadal@digiteqautomotive.com
353 mailto:jiri.dostal@digiteqautomotive.com
354 mailto:oleksandr.bashtannik@digiteqautomotive.com
355 mailto:ludmila.korinkova@digiteqautomotive.com
362 https://devstack.vwgroup.com/jira/browse/VWOS-16723?src=confmacro

Index	BSrv					
	Ticket	**Team / component(s)**	**Component with POC**	**Point of contact / role**	**Status**	**Notes**

				• Dev (Test Manager):
				• Dev (Test Manager) Deputy:
				• Dev (Software architect):
				• Dev (Fuzzing responsible): Isokaeaentae, Johanna (EXTERN: EB) Mail: johanna.isokaeaentae@elektrobit.com[356], ??? Jiri (Digiteq Automotive) Mail: jiri.dostal@digiteqautomotive.com[357]
				
				OLD Entry (Before 06 Jan 2025):
				• **TM Obd & TM Base**
				  • Muenz, Vladimir
				  • **Vladimir.Muenz@**
					• Team internal Task:
					  E3APP004-5829[363] - [TM-Classic] CI-Fuzz Testing for TM Base/TM OBD - R24.4 - Part 2 DONE
					•
					
					TM OBD (Autosar Classic):
					• Some support needed. About 70% targets for testing created
					  • Task:	

356 mailto:johanna.isokaeaentae@elektrobit.com
357 mailto:jiri.dostal@digiteqautomotive.com
363 https://devstack.vwgroup.com/jira/browse/E3APP004-5829?src=confmacro

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes

				digiteqautomotive.com[358]
				• **TM SP**
				  • Bashtannik, Oleksandr
				  • oleksandr.bashtannik@digiteqautomotive.com[359]
				• Team members / related
				  • Patrik Moravek
				  • Patrik.
					• VWOS-16413[364] - [Dependency] Fuzzing activities TM Base/ TM OBD/ TM SP DONE	

358 http://digiteqautomotive.com
359 mailto:oleksandr.bashtannik@digiteqautomotive.com
364 https://devstack.vwgroup.com/jira/browse/VWOS-16413?src=confmacro

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes
				Moravek@digiteqautomotive.com[360]	• Kanban board Task: • ☑ VWOS-16723[365] - Support Team TimeTravelers TM Base/TM OBD/TM SP in fuzzing - PI25.1 DONE	

360 mailto:Patrik.Moravek@digiteqautomotive.com
365 https://devstack.vwgroup.com/jira/browse/VWOS-16723?src=confmacro

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes
					• Team internal Task: E3APP004-6034[366] - [TM-Classic] CI-Fuzz Testing for TM Base/TM OBD - R25.1 DONE •	

366 https://devstack.vwgroup.com/jira/browse/E3APP004-6034?src=confmacro

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes

					TM SP (Autosar Adaptive):
					• Some support needed. About 70% targets for testing created
					  • Task:
					  • ☑ VWOS-16413[367] - [Dependency] Fuzzing activities TM Base/ TM OBD/ TM SP DONE	

367 https://devstack.vwgroup.com/jira/browse/VWOS-16413?src=confmacro

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes
					• Kanban board Task: • ☑ VWOS-16723[368] - Support Team Time Travelers TM Base/ TM OBD/ TM SP in fuzzing - PI25.1 DONE • Team internal Task:	

368 https://devstack.vwgroup.com/jira/browse/VWOS-16723?src=confmacro

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes

10	☑ VWOS-11119[369] - Follow-up support DRM in fuzz testing DONE	DRM [373]
		DrivingReflectionModule 1.2
			e3_app_[374]e3_app_drm[375]drm
			• JsonUtil_Fuzz_Test.cpp[376]
			• my_fuzz_test_1[377]
			• my_fuzz_test_2[378]
			• my_fuzz_test_3[379]
				• Radhika Putty **PO**
				  • **radhika.putty@**embitel.de[380]
				• Belur Nataraj, Navya **TL**
				  • navya.bn@embitel.com[381]
				• Team members / related
				  • ankita.parija@embitel.com[382]
				  • premalatha.vijayabalan@embitel.com[383]
					IN PROGRESS
					Team is trying to fuzz independently and will contact us if they need support
					Team has written 1 fuzz target on their own. And wrote their own short How to run fuzz test[387]. Our page is liked there
					08 Apr 2024 Team has not planned SP for PI24.2, we will clarify this in SCB on 11 Apr 2024
					TODO Might have to update APfuzzing.cmake version
					OUTDATED
						Muckenschnabl, David Jose (EXTERN: ITK)
						
						20221031 Onboarding DRM für Fuzzing.pdf[388]
						2023.11.24 Sync fuzzing topic team DRM (see page 251)

369 https://devstack.vwgroup.com/jira/browse/VWOS-11119?src=confmacro
373 https://devstack.vwgroup.com/confluence/x/-Cf8L
374 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_drm/browse?at=refs%2Fheads%2Fcustom%2Fcifuzz
375 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_drm/browse/tests?at=refs%2Fheads%2Ffeature%2FE3APP013-5905-set-up-fuzz-environment-and-try-out-a-test
376 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_drm/browse/tests/fuzzing/JsonUtil_Fuzz_Test.cpp?at=refs%2Fheads%2Ffeature%2FE3APP013-5905-set-up-fuzz-environment-and-try-out-a-test
377 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_drm/browse/tests/fuzzing/my_fuzz_test_1.cpp?at=refs%2Fheads%2Ffeature%2FE3APP013-5905-set-up-fuzz-environment-and-try-out-a-test
378 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_drm/browse/tests/fuzzing/my_fuzz_test_2.cpp?at=refs%2Fheads%2Ffeature%2FE3APP013-5905-set-up-fuzz-environment-and-try-out-a-test
379 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_drm/browse/tests/fuzzing/my_fuzz_test_3.cpp?at=refs%2Fheads%2Ffeature%2FE3APP013-5905-set-up-fuzz-environment-and-try-out-a-test
380 http://embitel.de
381 mailto:navya.bn@embitel.com
382 mailto:ankita.parija@embitel.com
383 mailto:premalatha.vijayabalan@embitel.com
387 https://devstack.vwgroup.com/confluence/display/E3APP013/How+to+run+fuzz+test
388 https://devstack.vwgroup.com/confluence/download/attachments/925873858/20221031%20Onboarding%20DRM%20f%C3%BCr%20Fuzzing.pdf?api=v2&modificationDate=1671115359709&version=1

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes
	E3-92646[370] - DONE E3APP013-5905[371] - Setup Fuzz environment and try out a test DONE			• shrikrishna.hebbar@embitel.com[384] • chandana.kushakumar@embitel.com[385] • amit.kulkarni@embitel.com[386]		2024.01.11 Sync fuzzing topic team DRM (2. Part) (see page 255) 2024.03.15 [DRM] Session with Rahul to clarify fuzzing related questions (see page 286)

370 https://devstack.vwgroup.com/jira/browse/E3-92646?src=confmacro
371 https://devstack.vwgroup.com/jira/browse/E3APP013-5905?src=confmacro
384 mailto:shrikrishna.hebbar@embitel.com
385 mailto:chandana.kushakumar@embitel.com
386 mailto:amit.kulkarni@embitel.com

Index	BSrv					
	Ticket	Team / component(s)	Component with POC	Point of contact / role	Status	Notes
	E3APP013-5374[372] - [DRM] Establish CI-Fuzz Testing for all external components (follow up) DONE					

372 https://devstack.vwgroup.com/jira/browse/E3APP013-5374?src=confmacro

MID / MACRO / MICRO						
Ticket	**Team / component(s)**	**Component with POC**	**Sessions**	**Point of contact / role**	**Status**	**Notes**
VWOS-11991[389] - Integrate MID Stack to fuzzing DONE VWOS-9140[390] - DONE	**MID**	**TODO Send info to Nikci Betim**	04 Jul 2023 Joint sync with MID, MACRO, MICRO	• @Nikci, Betim (T1-24)[391] (MID)	Onboarding session carried out 12 Jul 2023	2023.07.04 Onboarding session MID / MACRO / MICRO Stack (see page 211) 2023.07.12 Onboarding session MID Stack (see page 218)
2.0 STOP VWOS-9222	**MACRO**		04 Jul 2023 Joint sync with MID, MACRO, MICRO	• @Morillo-Velarde, Victoria (T1-27)[392] (MACRO). • @Toso, Gianpiero (T1-27)[393] (MACRO)	2.0 STOP Set up a PoC in RUST code. A plus would be to have a finding so teams see the use of fuzzing	2023.07.04 Onboarding session MID / MACRO / MICRO Stack (see page 211) 2023.07.27 Sync Session with MACRO Stack (see page 229)

389 https://devstack.vwgroup.com/jira/browse/VWOS-11991?src=confmacro
390 https://devstack.vwgroup.com/jira/browse/VWOS-9140?src=confmacro
391 mailto:betim.nikci@cariad.technology
392 mailto:victoria.morillo-velarde@cariad.technology
393 mailto:gianpiero.toso@cariad.technology

MID / MACRO / MICRO						
Ticket	**Team / component(s)**	**Component with POC**	**Sessions**	**Point of contact / role**	**Status**	**Notes**
☑ VWOS-9737[394] - Integrate MICRO Stack to fuzzing WON'T FIX ☑ VWOS-9223[395] - DONE	**MICRO**	TODO find an example for trying the build system	04 Jul 2023 Joint sync with MID, MACRO, MICRO	• @Perera, Yasas (T1-24)[396] (MICRO) • @Himmighofen, Andre (T1-24)[397] (MICRO)	IN PROGRESS Find example to tryout build system, following this set up meeting to start set up of build process	2023.07.04 Onboarding session MID / MACRO / MICRO Stack (see page 211) 2023.07.19 Sync session regarding fuzzing with MICRO Stack (see page 226)

Inspiration:

Structured the teams like this: Security Plan Training[398]

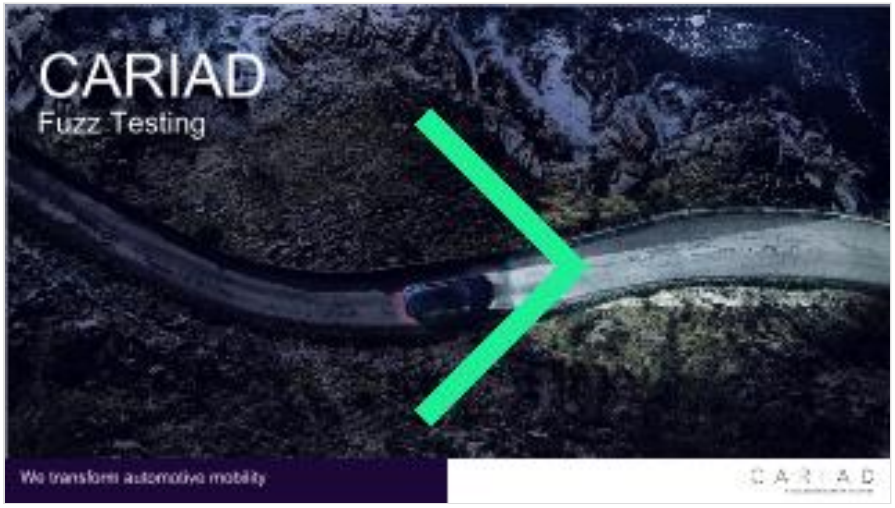
Added the Q&A like this: FAQ[399]

Keep track of people in detail. Not only team names.

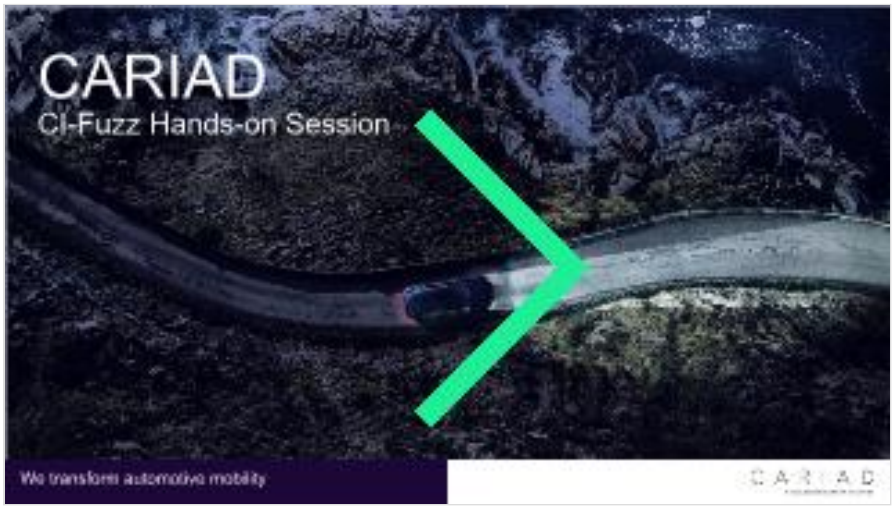
14.2 Onboarding slides

First Sync session

394 https://devstack.vwgroup.com/jira/browse/VWOS-9737?src=confmacro
395 https://devstack.vwgroup.com/jira/browse/VWOS-9223?src=confmacro
396 mailto:yasas.perera@cariad.technology
397 mailto:andre.himmighofen@cariad.technology
398 https://devstack.vwgroup.com/confluence/spaces/E3SEC/pages/683522474/Security+Plan+Training
399 https://devstack.vwgroup.com/confluence/spaces/E3SEC/pages/705382270/FAQ



Hands-on session



14.3 2023.07.04 Onboarding session MID / MACRO / MICRO Stack

14.3.1 Participants

Dang, Quang Anh (EXTERN: ITK)

Muckenschnabl, David Jose (EXTERN: ITK)

Schaller, Bernd (T5-SF)

Weichslgartner, Andreas, Dr. (TX-NS)

MID

Unknown User (drqsass)
MICRO

Himmighofen, Andre (T5-PB)

Perera, Yasas (TX-NP)

MACRO

Toso, Gianpiero (TX-NP)

Morillo-Velarde, Victoria (T1-Y)

14.3.2 Protocol

	MID	MACRO	MICRO
Code & Teams	Uses conan heavily. Conan = integration tool C/C++ Code only Compiler: qcc / gcc (for linux) Use Docker containers for building. Not based on the docker_commonpart Use Qemu MID stack uses another docker container which doesnt based on the common part one Example: https://git.hub.vwgroup.com/swp-vwos/vwos-aux-elapsed-time-meter TODO: Separate introduction / meeting Contact for conan and architecture related topics: Unknown User (sq5ybff) Information from Unknown User (drqsass) " It looks like that we can in general nicely embed it to our project, although a lot of details need to be clarified. Due to the nature of your tooling and our project setup (C/C++, cmake, heavily using conan), it seems like we can easily introduce CI fuzz – with according onboarding / training. @Alex + Aleksei:	RUST code: Github => Bendroid (Contact: Morillo-Velarde, Victoria (T1-Y)) Unknown User (vx1zsdn) is currently working on POC for this Android Code: BaseOS (Contact: Toso, Gianpiero (TX-NP)) Kotlin Code: Bendroid Java Code: Bendroid Many components dont use conan => TODO: Separate introduction / meeting	Usage of Bazel instead of Cmake. MICRO bazel server is behind the required bazel version from cifuzz. TODO: Separate introduction / meeting

	MID	MACRO	MICRO
	This is the protocol of the meeting: https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1701451103 (see page 211) @CI Fuzz team: This is a demo application and project: https://git.hub.vwgroup.com/swp-vwos/vwos-aux-elapsed-time-meter Maybe you can already verify how to quickly and easily introduce CI fuzz in our project. "		
Demand for fuzzing	Yes (specific training/ onboarding would be needed... Aleksei + some devs should be present)	not sure	not sure
Current BUG handling process			

TODO: Invite every stack into a separate meeting to discuss specifics.

14.3.3 FAQ

- Nikci Betim
  - Is the workflow provided by security team? Yes it is a reusable workflow set up by us.
  - Can we run QNX applications? We build the project from scratch and test parts of the code.
  - Do we have access to the tool through github? Yes it will work with github.
  - We would like an extra intro into fuzzing. It looks like it would not be that complicated to integrate fuzzing into the stack.
  - Can we integrate the build process in our setup/containers ? Yes we can, this is poddible
- Yasas Perera

    - How did we ended up using CIFuzz? We evaluetaed many tools in the market and CIFuzz fits for out usecase. TA and TS looked into it. CiFuzz uses LibFuzzer, we use it to have a server and a cicd integration.
    - How do we provide an entry point ? It is defined by the function FUZZ_TEST, an optional seed can be used to make the process of fuzzing faster.
    - Where is the test executed, If we start fuzzing in our CICD where will it run? It will run in the CI-server provided by PMT team it will be accessible for teams. Continous fuzzing is the goal.
    - You mentioned sanitizers, what kind do you mean? To observe a crash you need sanitizers t catch errors and bugs. It can be built with extra sanitizers.
    - NOTE: Yasa is interested in the list of tools we use. Dang, Quang Anh (EXTERN: ITK) forward the list of tools
- Tosso Giampiero:
    - We do not build from source. To cover Macro we have to Track all the people. We have android and Rust projects. And are focusing on using Google tools.
    - We are fuzzing Rust already
    - It is conan heavy, we use conan packing. We use cappa. Will this be compatible?
- Victoria Morillo-Velarde
    - Do we assume when we fuzz part of the code that the hole stack is error free? Yes we believe if we have fuzzing on Component level it will make the stack stronger.
    - Some parts of code we do not have source code of. In that case we we would have to go to the team that produces this.
    - Schaller, Bernd (T5-SF) invites Toso, Gianpiero (TX-NP) Morillo-Velarde, Victoria (T1-Y) to a meeting to discuss if and how Macro would use fuzzing. Also discuss What parts/ components have source code and which are provided by some other team.
- Himmighofen Andre
    - Have we tested it in an embedded system? No, we have not.
    - We have so many basics missing in our test coverage, where I do not see value. Response: It can find a lot of types of errors.
    - We have bazel. We use different compilers, we have to see how it will work, when building a fuzz-target. We are working to use the new Bazel update. Response: CIFuzz supports Bazel. We have not tried it yet. For commercial compilers we will net to use clang.
    - We would like to have hardware in the loop. But it is hard to manage. We are interested.

14.4 2023.07.11 Onboarding session SOCD-Babelfish/Comet

14.4.1 Participants

Dang, Quang Anh (EXTERN: ITK)

Muckenschnabl, David Jose (EXTERN: ITK)

Ostrowski, Igor (EXTERN: int2code)

Iuga-Popescu, Lucian (EXTERN: Cetitec)

Unknown User (u8ebth4)

Unknown User (gjillnl)

14.4.2 Protocol

- Introduction to Fuzzing and CIFUZZ process done by Dang, Quang Anh (EXTERN: ITK)
- SOCD-Babelfish code functionality and architecture

14.4.3 FAQ

- Q: We have an architectural change we made a second SRE, why do we now have the fuzzing topic ? (Zange Kevin)
  - A: The fuzzing process is under construction and is new in the process.
- Q: What are the requirements for fuzzing? Are they architecture specific ? (Zange Kevin)
  - A: Process of fuzzing & release criteria/definition of done (in progress) (see page 152)
- Q: How do you measure the function coverage? (Rafal Brewka)
  - A: The tool generates a coverage file and we can extract the information from there. The coverage from the tool will be compared with the list of functions/components that should be fuzzed.
- Q: The 80% coverage, is it meant for only the relevant functions ?(Paul Koch)
  - Yes.
- Q: Do we have a guide ? (Koch Paul)
  - Yes https://devstack.vwgroup.com/confluence/display/E3SEC/CI-Fuzz+Proof+of+Concept. (see page 136) https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=860960818 (see page 114).
- Q: Do we need to fuzz ourselves? Or analyse the bugs ourselves?
  - Right now we will create examples and in the future a tester would write the targets and analyze them.
- Q: Do we provide a severity for the bugs?
  - Yes the server has a database of bugs mapped to their severity.
- Q: What are the next steps ?
  - We write PoCs and contact you again to showcase these.

Component Service bridge e3_comp_servicebridge, no good for PoC, it is the most complex

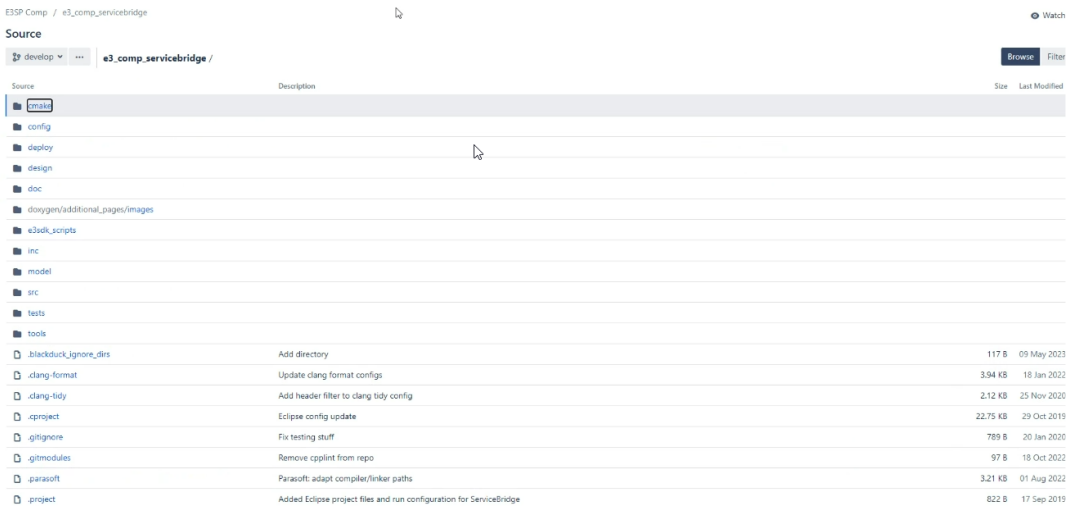
-

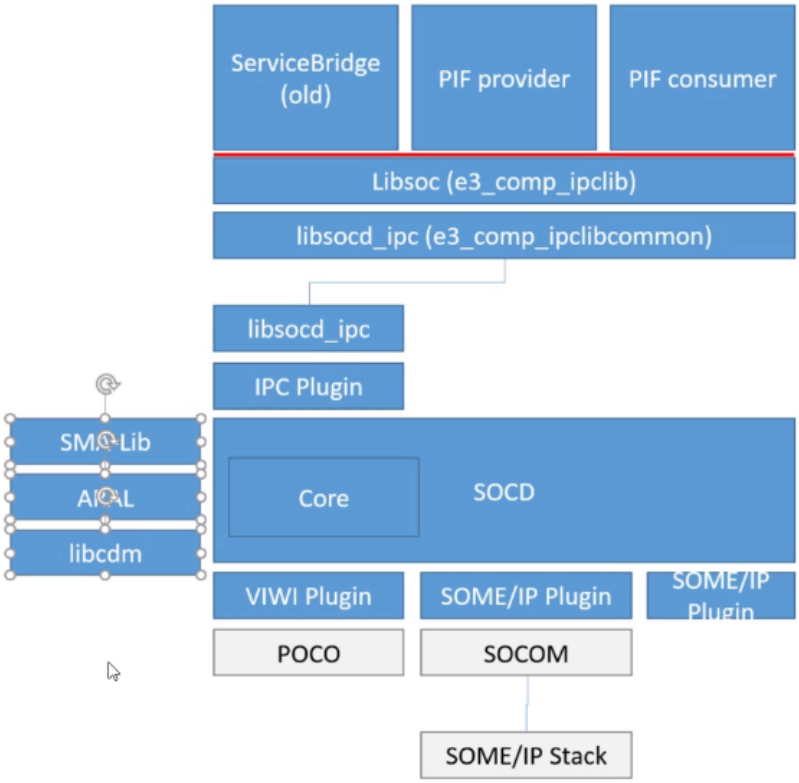
libsoc is the main interface to the applications that want to use the stack

e3_comp_icplib: libsoc_ipc It contins ipc transpor mechanisms is used to comunicate. it is used by libsoc and Soc deamon. We should start here, LibSoc

- IPC communication towards
- Should be the start point for fuzzing because this

SocDeamon is the deamon component it uses a poco library to add vivi headers and forward them to the applocation, it has security checks, checks for keys. Connects to diagnostics and many more interfaces. Has a core section (routing). Plugins for different protocols, cdm plugin is usde for intrusion detetion





Libsoc, VIWI Plugin, SOME/IP Plugin

[14:06] Koch, Paul (T1-23)

Example for a costumer[400]

e3_subm_comdaemonpluginviwi /

e3_comp_ipclib / src / libsoc / socapi / include / soc / **api** /

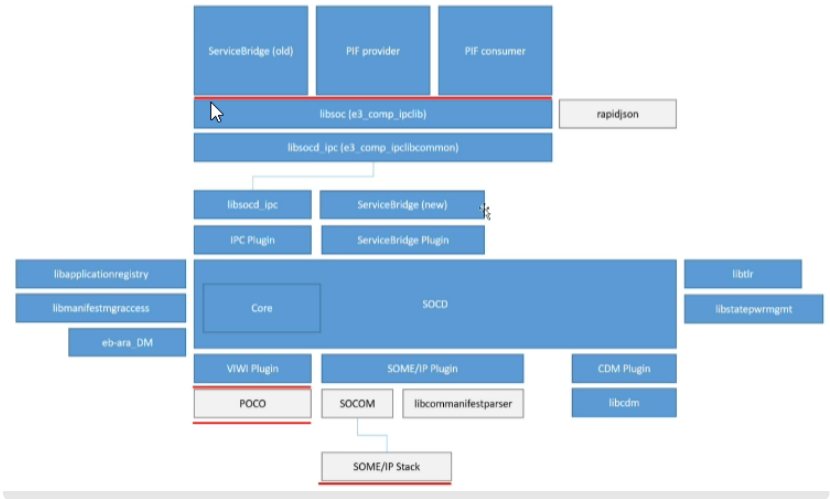
https://devstack.vwgroup.com/bitbucket/projects/E3SPSUBM/repos/e3_subm_comdaemonpluginviwi/browse?at=develop

https://devstack.vwgroup.com/bitbucket/projects/E3SPSUBM/repos/e3_subm_socd_someip/browse?at=develop

https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_ipclib/browse/src/examples/socapi/subscription/consumer/src

VIWI (Paul Koch)

Libsoc (Rafal Brewka)



14.5 2023.07.12 Onboarding session MID Stack

14.5.1 Participants

Dang, Quang Anh (EXTERN: ITK)

400 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_ipclib/browse/src/examples/socapi/subscription/consumer/src

Muckenschnabl, David Jose (EXTERN: ITK)

Unknown User (drqsass)

Weichslgartner, Andreas, Dr. (TX-NS)

Schaller, Bernd (T5-SF)

14.5.2 Protocol

MID Stack: Driving Assistance

Agenda	Discussion
Demand for fuzzing	Yes (specific training/onboarding would be needed... Aleksei + some devs should be present)
SRE & SRA & SRS Demand for fuzzing	Status of SRE for MID Stack? TODO Which teams / components would be interested in / require fuzzing? TODO Aleksei + some devs should be present
Introduction to MID Stack	Programming language: C/C++ Current build process: Conan / Docker Compiler: gcc / qcc Docker Container: Build Tools are packaged via https://git.hub.vwgroup.com/swp-mid/vwos-mid-build-tools/blob/develop/vwos-mid-build-tools-linux/conanfile.py
Fuzzing Candidates	List of components 28 Teams in MID Stack Effective 10-15 Teams

Agenda	Discussion
Access	How to git clone https://git.hub.vwgroup.com/swp-vwos/vwos-aux-elapsed-time-meter and branch? **Error cloning project** ``` bi@bi-VirtualBox:~/workspace/tmp$ git clone https://git.hub.vwgroup.com/swp-vwos/vwos-aux-elapsed-time-meter.git Cloning into 'vwos-aux-elapsed-time-meter'... fatal: unable to update url base from redirection: asked for: https://git.hub.vwgroup.com/swp-vwos/vwos-aux-elapsed-time-meter.git/info/refs?service=git-upload-pack ```

Agenda	Discussion

```
   redirect: https://idp.cloud.vwgroup.com/auth/
realms/kums-mfa/protocol/saml?
SAMLRequest=fZJNT8IwGMe%2FytL71m04Jw1bgnCQBJXA9OD
FdFthjV07%2B7S8fHsLkwgeOPf%2F%2FP4v6QhoKzoytqaRS%
2FZtGRhv3woJ5PSQIaslURQ4EElbBsRUZDV%2BnpM4CEmnlVG
VEsgbAzBtuJITJcG2TK%2BY3vKKvS3nGWqM6YBgvOEmaGwZbH
cbrWwXVKrFq4aXpRLMNAGAwkd0jBevqwJ5U5eFS3qk%2FjF47
e6EsvUVhbr4WDMqWsBftgW%2FXVN8ToePTZA3m2boc10mgyQa
lGGVDquQJsO7%2B4TVaToI6zR6iJiTAVg2k2CoNBmKw3jgh6k
fxUUUkiQicfiBvMUv%2BZHLmsvN7ZHKXgTkqSgWfl%2FtnWk4
1XIClI%2BOAcnJWF%2BMfxtLz4uj%2FLwNZdJ3pfeH%2FyuP8
IVDb9eRF4ecTRdK8OrgjYVQu4lb0LAMRQjn%2Fcn1x8h%2FAA
%3D%3D&RelayState=ss%3Amem%3A1fc01ca5315d8291b48a
3a360ba7ac6b9fa55b09d77722d05bb474bf9cc93142&SigA
lg=http%3A%2F%2Fwww.w3.org%2F2001%2F04%2Fxmldsig-m
ore%23rsa-
sha256&Signature=WzoaUGwztVIhsbg9O8zv3T1mmIIZGnQ%
2BzX3oxoNmmOG4IR0HtsLcl4MBNNfTrv8RNzsoXaQuCrtRzbN
gCC9hxNWw2Ah94mB2uA28Hw44mbt2%2B9aIkWMUKIe4zOAsGv
KET0Jln5nbQ8cQX%2FmCQHk1Idg%2Fvm%2FTSS6mqK%2BZL5S
OXcWyud2v%2BCe%2FHSxtBmV7sgcA4E7K5Dr2jLJBblNSKkFd
DYm1rHg9HrY39hlSXkD7vC%2FCt1y94ZJlC5s%2Fp%2Bv%2Fy
jOxLhDFFskAkgupS13L25xaE0SCFtwgPGbJUQ8N8H2eAssCzA
HRLiRwAmV2VSU3iB%2BMTNfqfZBkfOIeQ9fBMc0I%2BLyDDLQ
pPnarZ7goTKVK8QVw9LFo5QAIpSmuwTUGwQhYGZQbczC5I3o9
Jr9p7DxpjV0%2BiNKcNYkJn8onxNEcW61Oy08%2FL5qwfBrAk
XtKk1tvRHgrz8rQxnkr8orMu%2B%2FJi4elUvv9EyvEnYIDMu
HO3wD9iMcmcpWuvvP1mdOC3ZcI
```

=> Lösung: https://git.hub.vwgroup.com/swp-mid/vwos-mid-wiki/wiki/DevEnv-Setup

Access to build?

=> Lösung: https://git.hub.vwgroup.com/swp-mid/vwos-mid-wiki/wiki/DevEnv-Setup

Build error?

Build error

```
bi@bi-VirtualBox:~/workspace/vwos-aux-elapsed-
time-meter-develop$ conan create
deployment.conanfile.py -pr:h qnx-host

WARN: Migration: Updating settings.yml

WARN: ****************************************
```

Agenda	Discussion
	``

WARN: settings.yml is locally modified, can't be updated

WARN: The new settings.yml has been stored in: /home/bi/.conan/settings.yml.new

WARN: **

ERROR: Profile not found: qnx-host

bi@bi-VirtualBox:~/workspace/vwos-aux-elapsed-time-meter-develop$ conan create conanfile.py -prh qnx-host --require-override elapsed_time_meter-deployment/1.0.0

ERROR: Profile not found: qnx-host```
=> Lösung: https://git.hub.vwgroup.com/swp-mid/vwos-mid-wiki/wiki/DevEnv-Setup |
Current BUG handling process	No bug handling process: ▪ No JIRA ticket ▪ No github issue ▪ Not yet an established Release Control Board. Bug fixing is done ad-hoc and manual.
Contact person	PO MID Integration Stack: Unknown User (sq5ybff)
TODO	• Shared Service Team Security provides documentation and tutorial • MID Stack will create profile for clang compiler and integrate the tool MID Stack currently hasn't planned Fuzzing Acitivites for this PI. Next PI the latest.
Useful links	https://git.hub.vwgroup.com/swp-vwos/vwos-aux-elapsed-time-meter/actions/runs/704374 https://git.hub.vwgroup.com/swp-mid/vwos-mid-wiki/wiki/DevEnv-Setup https://git.hub.vwgroup.com/swp-mid/vwos-mid-conan-settings/blob/release/conan_settings/profiles_includes/linux-host-common https://jfrog.hub.vwgroup.com/ui/repos/tree/General/cicd-maven-virtual https://git.hub.vwgroup.com/swp-mid/vwos-mid-build-tools/blob/develop/vwos-mid-build-tools-linux/conanfile.py https://git.hub.vwgroup.com/swp-mid/vwos-mid-wiki/wiki/How-to-use-MID-stack-release#integrate

04 Jul 2023 Status

Uses conan heavily.

Conan = integration tool

C/C++ Code only

Compiler: qcc / gcc (for linux)

Use Docker containers for building. Not based on the docker_commonpart

Use Qemu

MID stack uses another docker container which doesnt based on the common part one

Example: https://git.hub.vwgroup.com/swp-vwos/vwos-aux-elapsed-time-meter

TODO: Separate introduction / meeting

Contact for conan and architecture related topics:
Unknown User (sq5ybff)

Information from Unknown User (drqsass)

"

It looks like that we can in general nicely embed it to our project, although a lot of details need to be clarified.

Due to the nature of your tooling and our project setup (C/C++, cmake, heavily using conan), it seems like we can easily introduce CI fuzz – with according onboarding / training.

@Alex + Aleksei:

This is the protocol of the meeting: https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1701451103 (see page 211)

@CI Fuzz team:

This is a demo application and project:

https://git.hub.vwgroup.com/swp-vwos/vwos-aux-elapsed-time-meter

Maybe you can already verify how to quickly and easily introduce CI fuzz in our project.

"

14.5.3 FAQ

14.6 2023.07.14 Fuzzing Activity for LUM

14.6.1 Participants

Dang, Quang Anh (EXTERN: ITK)

Weichslgartner, Andreas, Dr. (TX-NS)

Schaller, Bernd (T5-SF)

Muckenschnabl, David Jose (EXTERN: ITK)

Hoepfner, Marcus (T5-PH)

Bulai, Ovidiu-Daniel (T5-PH)

14.6.2 Protocol

- Current status fuzzing for LUM PI 23.3:
  - 2 collueges less this PI
  - This PI fuzzing is not planned for LUM due to different priority
- Future plan:
  - Sync with Tihomir the current progress of fuzzing after Tihomir comes back from vacation on 01 Aug 2023
    - What works?
    - What are still need to be done?
    - The current problems why run fails
    - Clarify how to do statefull fuzzing POC
  - Planning capacity based on backlog for PI 23.4

14.6.3 FAQ

14.7 2023.07.14 Security Activities on MACRO Level

14.7.1 Participants

Dang, Quang Anh (EXTERN: ITK)

Weichslgartner, Andreas, Dr. (TX-NS)

Schaller, Bernd (T5-SF)

Unknown User (ez1gcj4)

MACRO STACK:

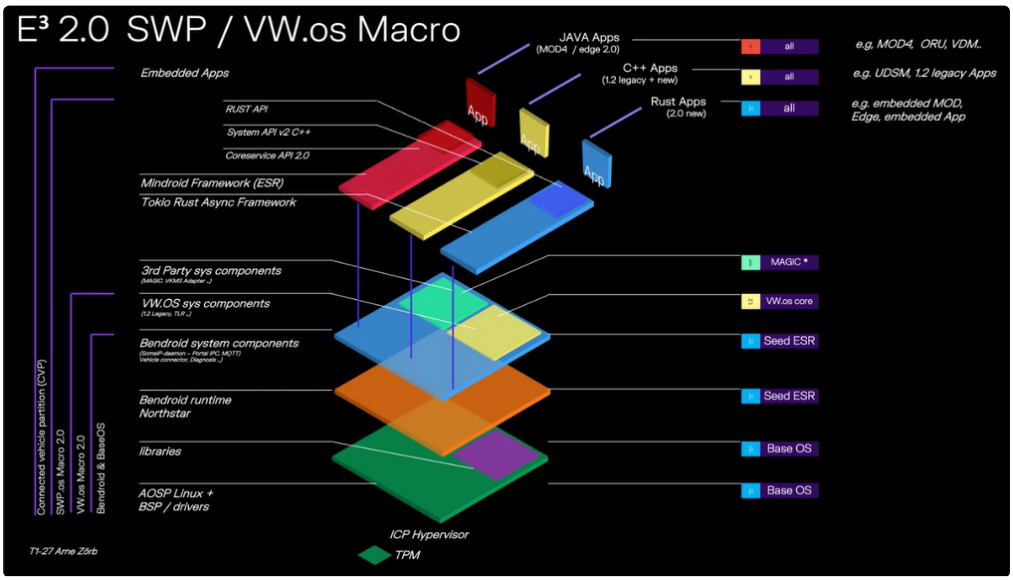
Toso, Gianpiero (TX-NP)

Morillo-Velarde, Victoria (T1-Y)

Zoerb, Arne (T1-KU)

14.7.2 Protocol

MID Stack: Driving Assistance

Identification of Risk	 ▪ API parts receiving input from outside What can be rebuilt from Source right now? ▪ Android BaseOS - 100% RUST Code Not suitable, unable to build from source (Victoria Morillo): ▪ Android BaseOS - Doesn't make sense to fuzz right now because complicated ▪ GDC (C++ Code) - Dev-ops Azur Platform Future work: Build everything from Source

Next steps	• Identify the tools to identify all the bugs and low hanging fruits
	• Find a point to do fuzzing. Probably nightly run parallel to smoke tests
	  •
	  • Regression Tests parallel to smoke/unit test
	• Muckenschnabl, David Jose (EXTERN: ITK) move the meeting on 18 Jul 2023 with MACRO cause Toso, Gianpiero (TX-NP) won't be available. Need Morillo-Velarde, Victoria (T1-Y) for RUST topic and Daniel Haap

14.7.3 FAQ

14.8 2023.07.19 Sync session regarding fuzzing with MICRO Stack

Participants:

Perera, Yasas (TX-NP) (Contact person)

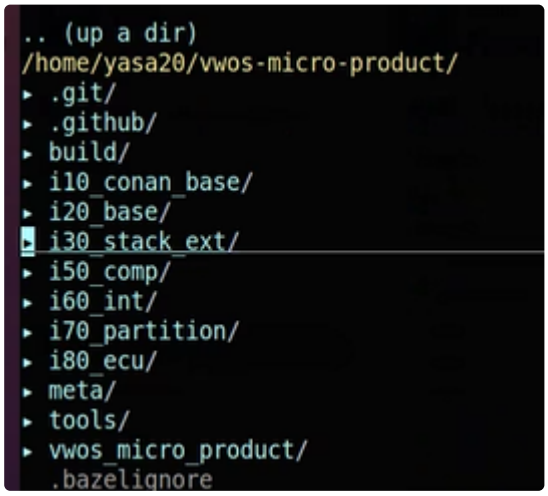
Dang, Quang Anh (EXTERN: ITK)

Weichslgartner, Andreas, Dr. (TX-NS)

Muckenschnabl, David Jose (EXTERN: ITK)

Agenda	Discussion
Demand for fuzzing	

Agenda	Discussion

Introduction to MICRO Stack	Software for Micro Controller written in C.
	What is vwos-micro-product?
	• Anything related to CAN Network. How to get signal?
	• (ECU Firmware) MICRO Product software will go to those Micro Controller
	• Micro produces packages to be used to assemble ECU
	Repository: https://git.hub.vwgroup.com/CARIAD/vwos-micro-product
	 
	
	Code comming from different teams:
	▪ i30_stack_ext: Plugins
	  ▪ There are unit tests for each of these plugins
	▪ i50_comp: Different software component
	  ▪ There are unit tests for each of these software components
	Micro Controller own code:
	▪ i10_conan_base
	▪ i20_base
	=> packaging library, creating conan package, build script
	
	MICRO Stack provides the tooling for other teams to fuzz.
	
	Build Tools: Bazel

Fuzzing Candidates	
Access	

Agenda	Discussion
Current BUG handling process	
Contact person	
TODO	☐ Adjust the fuzz process for C and MICRO Stack set-up. • Toolchain for fuzzing in i20_base ☐ We will create Fuzzing Build Process for MICRO Stack ☐ Get Push and create branch access to the repo https://git.hub.vwgroup.com/CARIAD/vwos-micro-product ☑ Muckenschnabl, David Jose (EXTERN: ITK) will set up a meeting to see how the build system works and start setting up the build system for fuzzing
Useful links	Central Information point for MICRO Stack: Good to know[401] Getting Access to Github repo: Good to know[402]

14.9 2023.07.20 Hands-on session CodeBuster

14.9.1 Participants

Dang, Quang Anh (EXTERN: ITK)

Muckenschnabl, David Jose (EXTERN: ITK)

CodeBuster Teams:

Komenaka, Bjoern (EXTERN: ITPower)

Unknown User (avg6ib6)

Hennig, Christian (T5-PF)

Quarre, Peter (T5-PF)

Unknown User (alqqoe1)

14.9.2 Protocol

- Showcase created PoCs

401 https://devstack.vwgroup.com/confluence/spaces/BSWM/pages/661114699/Good+to+know
402 https://devstack.vwgroup.com/confluence/spaces/BSWM/pages/661114699/Good+to+know#Goodtoknow-GettingaccesstoVW.osrepositories

    - Demonstration fuzzing of validateSetFucntionState
- Implement PoC with a developer
    - Hennig, Christian (T5-PF) volunteered for the hands-on run
    - Creating a fuzz target "response cache". It worked.
        - Half of the response_cache.cpp
        - see https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_fod_sp/commits/4c2a4418b5822529ef9dad37caf9d4042e62cbd1

14.9.3 FAQ

Hennig, Christian (T5-PF) asked what we want to test for?
=> we ignore exceptions and aim for runtime error such as heap/buffer overflow, undefined behavior, out of memory etc.

Hennig, Christian (T5-PF) executing the cifuzz run command, is the logs written somewhere?
Yes. inside .cifuzz-build/logs/...
The log is not updated real time though

Hennig, Christian (T5-PF) Recommended maybe reduce the $(nproc) by 1 to make the vm more responsive

Hennig, Christian (T5-PF) Do you think the issue that no bug is found is because of the logger?
=> The logger can be disabled by setting the log level

14.9.4 Notes

14.10 2023.07.24 Sync Session with MACRO Stack

14.10.1 Participants

Dang, Quang Anh (EXTERN: ITK)

Muckenschnabl, David Jose (EXTERN: ITK)

MACRO

Haap, Daniel (T1-KD)

Morillo-Velarde, Victoria (T1-Y)

Toso, Gianpiero (TX-NP)

14.10.2 Protocol

- Present CiFuzz and current build process
- Discuss Build systems on Macro side
- Discuss language scope on Macro
- Discuss Structure of the Stack

Agenda	MACRO
Demand for fuzzing	No SRE or SRA done or even planned
Introduction to MACRO Stack	MACRO Stack: No C++ in Bendroid right now Only fuzzable components are RUST and Kotlin code:
Fuzzing Candidates	• Micks Bug: https://git.hub.vwgroup.com/VWOSMacro/bendroid/pull/188
Access	Repo: https://git.hub.vwgroup.com/VWOSMacro/bendroid
Current BUG handling process	The BUG creation is being setup with JIRA right now:
Contact person	Set up build proces and POC Buidling Bendroid Daniel Haap(CICD expert), Process contact Victoria
TODOs	☐ Check out Micks work and create POC based on existing example. Best would be POC finding real bug

Useful links	• Kotlin Repo: https://git.hub.vwgroup.com/VWOSMacro/portal
	• Landing Page for MACRO https://git.hub.vwgroup.com/VWOSMacro
	  • Release
	  • Documents
	• RUST & Building Blocks
	  • https://pages.git.hub.vwgroup.com/Cariad/arch-vwos-raas-published-doc/latest-develop/#_macro_stack
	• https://pages.git.hub.vwgroup.com/Cariad/arch-vwos-raas-published-doc/latest-develop/stackview/stackoverview.html?stack=macro

14.10.3 FAQ

- Who is responsible for creating SRE and SRA?
  A security owner from our side in conjunction with MACRO stack

14.10.4 Notes

14.11 2023.07.25 Summary of Stacks language usage and component type

14.11.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Macro Stack:

- **Code Usage:** RUST, Java, Cpp
- **What they do:**
  - software platform for *µProcessor-based* systems intended for applications with a high demand for compute power and RAM or online connectivity functions, without safety requirements and without audio/video/graphics support.

Mid Stack:

- **Code Usage:** Cpp
- **What they do:**
  - **Persistency**
    - provides software components and APIs to store and read data from the file system.
    - provides functionality to check the integrity of data
  - **Communication**
    - provides service oriented communication
    - provides signal based communication

Micro Stack:

- **Code Usage:** C
- **What they do:** Applications hosted on µControllers with AUTOSAR classic as basis

14.12 2023.08.08 Sync and Hands-On session TLR

14.12.1 Participants

Dang, Quang Anh (EXTERN: ITK) (Bi)

Muckenschnabl, David Jose (EXTERN: ITK)

TLR TeamTeams:

Vujakovic, Jelena (EXTERN: Synchrotek)

Allak, Saad Ouassil (T5-VF)

Kuzel, Jakub (Digiteq Automotive)

14.12.2 Agenda 2023.08.08

- Introduction to Fuzzing (quick)
- Show case current build process starting with CI-Fuzz Proof of Concept (see page 66)
- Execute fuzz target on local machine of the volunteer

14.12.3 Protocol 2023.08.08

- Determine the volunteer to execute fuzz target in their local machine
  - Run
    docker pull e3-e3sp-dockerdev.docker.devstack.vwgroup.com/testing/e3sdk-build-bp1-commonpart:feature-vwos-7577-fuzzing-13-13c14f3-61fcffae[403]
    to save time
- Showcase created PoCs
  - Demonstration fuzzing of ...
- Implement PoC with a developer

14.12.4 Minutes 2023.08.08

- Bi introduces CiFuzz and the process we have drafted.
- Questions where asked an documented, some have to be cleared with Bernd
- Bi does over Demo including build process with build.sh
- Execution problems are solved on dev. environment
- Went over the new build process from zero on to test it on TLR dev. environment
- A follow-up has been created for 10 Aug 2023 to finish fixing dev. environment problems

403 http://e3-e3sp-dockerdev.docker.devstack.vwgroup.com/testing/e3sdk-build-bp1-commonpart:feature-vwos-7577-fuzzing-13-13c14f3-61fcffae

14.12.5 Minutes 2023.08.10 Follow-up

- Bi did a re-cap on fuzzing and what we, at the Shared Security Team, are doing with fuzzing
- Yelena and Jakub are now able to debug the TLR bug

  E3-122158[404] - e3_comp_tlrlib out of memory in vwg::e3p::tlr::createTLRUTF8StringMessage DONE
- TODO: A hands-on for the new build process is to be done another time due to lack of time

14.12.6 FAQ 2023.08.08

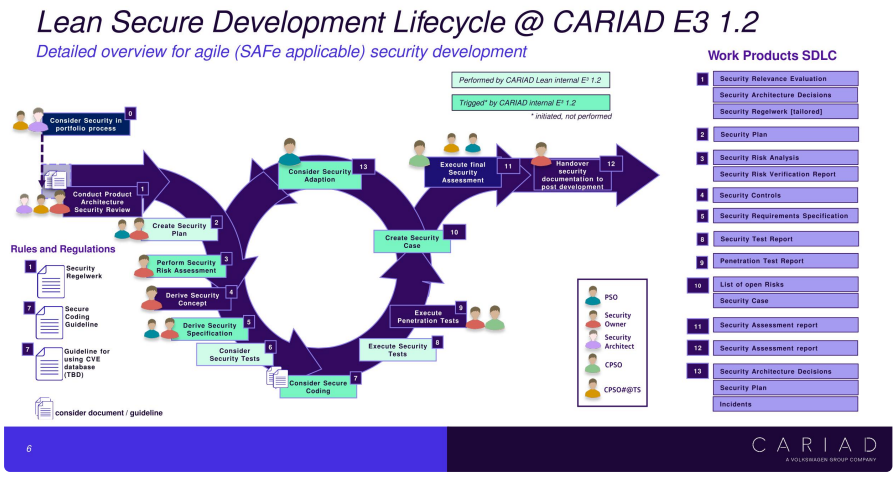
- Who create the List of fuzzing targets
  - Joint effort: Development team with support from shared security team
- Who does Risk analysis?
  - Joint effort from dev. team and Shared Sec. team
- When in the timeline should a Sec. Risk Analysis be done
  - Asap, to have a better overview of Security Risks while development.
- Are Security Reqs. and Stakeholder reqs. relevant in a Sec. Risk Analysis ?

  TODO: Ask Schaller, Bernd (T5-SF)
- What is the overview of fuzzing?
  - Many teams have been on-boarded list of on-boarded teams (see page 158)
  - We want to prepare all teams for the requirements come
- How to do an SRA, where can I find an SRA, is it a Document?
  - Are we Security relevant ? Yes/No (SRE)
  - Do we have to analyze Risks ? (SRA)
  - Is it the needed to create requirements for the Risks ? (SRS)
- Is there a deadline for a SRE ?
  - TLR has done SRE/SRA/SRS on Code-Beamer for 1.2 already (CodeBeamer)
  - For 2.0 Software Platform SWP Team (TA hat Steven Michna, Function Owner ist Malte Geisler) will be responsible as they have a FuLi Entry (meaning SWP will be responsible)

14.12.7 FAQ 2023.08.10

- Will the change in build systems affect the detected bugs ? (If a bug was detected with the old system and we try to reproduce it with the new system, will we find it ?)
  - Yes, the new build system should not change discoveries the fuzzer makes.

404 https://devstack.vwgroup.com/jira/browse/E3-122158?src=confmacro



14.12.8 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101)

14.13 2023.08.09 Sync and Hands-On session UDSM

14.13.1 Participants

Dang, Quang Anh (EXTERN: ITK) (Bi)

Muckenschnabl, David Jose (EXTERN: ITK)

USDM Team:

Singh, Deepanshu (EXTERN: Embitel)

Chamarajanagar Mohan Kumar, Kishore Kumar (EXTERN: Embitel)

Krishnamurthy, Manmath Kumar (EXTERN: Embitel)

Mishra, Nilesh Kumar (EXTERN: Embitel)

@Kumar Rohit

Unknown User (al9dinw)

@Jegatheesan Priyadharsan

Thippabattini, Jayasankar (EXTERN: Embitel)

14.13.2 Agenda

- Execute fuzz targets on one of the developer's machine
- Fuzz a new function together with a developer

14.13.3 Protocol

- Provide documentations (see interesting links)
- Determine the volunteer to execute fuzz target in their local machine
  - Run

    docker pull e3-e3sp-dockerdev.docker.devstack.vwgroup.com/testing/e3sdk-build-bp1-commonpart:feature-vwos-7577-fuzzing-13-13c14f3-61fcffae[405]
    to save time
- Showcase created PoCs and directory structure via https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_css/compare/diff?sourceBranch=refs%2Fheads%2Fcustom%2Fcifuzz&targetRepoId=31164&targetBranch=refs%2Fheads%2Fdevelop
- Implement PoC with a developer

14.13.4 Minutes

- Bi presents Build process and set-up of CiFuzz
- Singh, Deepanshu (EXTERN: Embitel) volunteered to execute target
  - build command worked
  - run command worked
  - coverage build worked
  - coverage overview was shared

14.13.5 FAQ

- Have you modified the existing profiles?
  - No we just added a new profile as a Hot-fix. In the future the fuzzing profile will be deleted and all of the existing profiles will contain the required tools for fuzzing.
- Where can we find commands required for fuzzing?

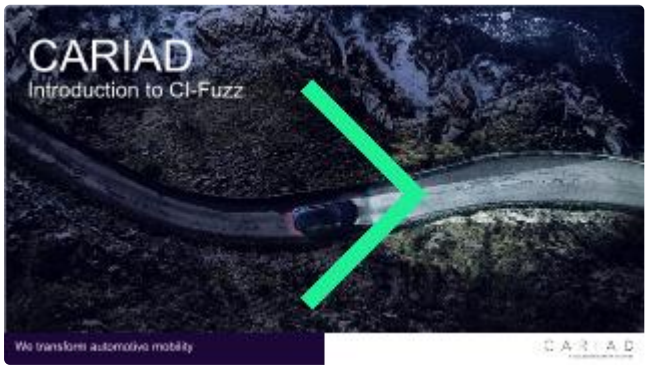
---

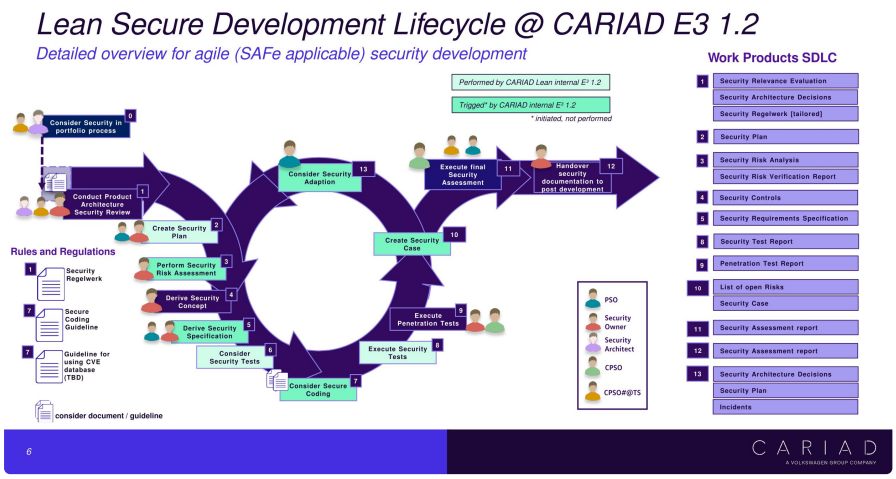
405 http://e3-e3sp-dockerdev.docker.devstack.vwgroup.com/testing/e3sdk-build-bp1-commonpart:feature-vwos-7577-fuzzing-13-13c14f3-61fcffae

    - All PoCs Created by us should have the command required commented.
    - Also in CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) all information about fuzzing is documented
- How can we make a target better? i.e pass a block in an if case
    - Here are some ideas https://devstack.vwgroup.com/confluence/display/E3SEC/CI-Fuzz+Proof+of+Concept#CIFuzzProofofConcept-Maintainingafuzztarget (see page 75)

14.13.6 TODOs

- [ ] Make a follow-up to improve target Dang, Quang Anh (EXTERN: ITK) Invite Singh, Deepanshu (EXTERN: Embitel) as a req.





14.13.7 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101)

14.14 2023.08.21 3rd Hands-On session TLR

14.14.1 Participants

Dang, Quang Anh (EXTERN: ITK) (Bi)

TLR TeamTeams:

Allak, Saad Ouassil (T5-VF)

Protocol:

- Trying to execute build with

```
./build.sh --docker --target fuzzing --profile fuzzing-x86_64 --verbose --fuzzing
-j2
```

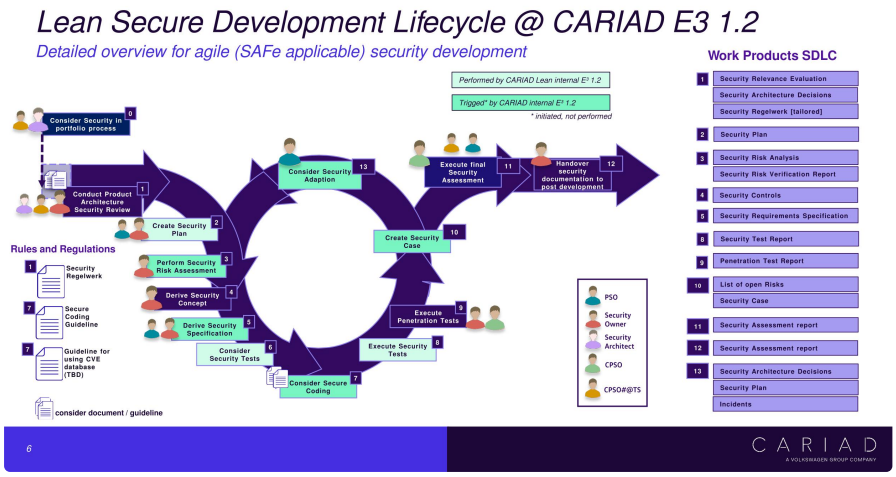
We got following problem:

```
ERROR: /home/spltxqd/Downloads/e3_comp_tlrlib/conanfile.py: option 'fuzzing'
 doesn't exist
Possible options are ['sanitizer', 'sanitizer_args', 'coverage']
conan install not successful. You used the following conan profile: fuzzing-x86_64
```

Issue: For some reason, the docker image on Allak, Saad Ouassil (T5-VF) is not up to date. The package vwos-conan-base inside the docker container should be version 1.2.0. To test this, execute:

```
$ e3sdk_scripts/shell.sh --docker --target fuzzing
$ python3 -m pip list> vwos-conan-base 1.2.0
```

Solution: Clean pull the image and then build again



14.14.2 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101)

14.15 2023.08.23 2nd Hands-On session UDSM

14.15.1 Participants

Dang, Quang Anh (EXTERN: ITK) (Bi)

Muckenschnabl, David Jose (EXTERN: ITK)

USDM Team:

Singh, Deepanshu (EXTERN: Embitel)

Unknown User (al9dinw)

Mishra, Nilesh Kumar (EXTERN: Embitel)

14.15.2 Agenda

- Improve Fuzz Target on one of the developer's machine

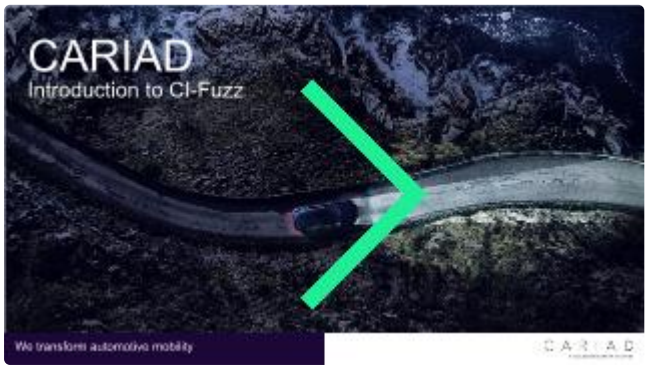
14.15.3 Protocol

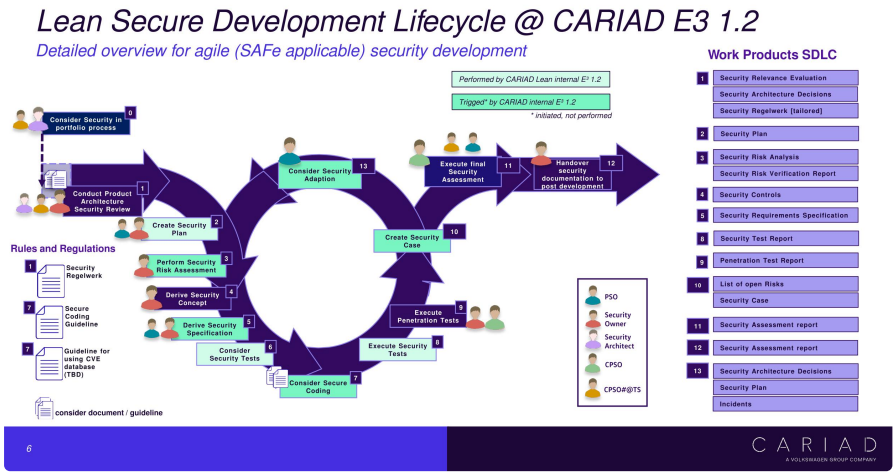
- Singh, Deepanshu (EXTERN: Embitel) volunteered to execute target
  - Issue: Code Coverage for fuzz_BucketHandler needs improvement. Stucked at the if-else clause sofar
- Showed case how to start a GDB session to debug a finding

14.15.4 FAQ

14.15.5 TODOs

- [x] Make a follow-up to improve target Dang, Quang Anh (EXTERN: ITK) Invite Singh, Deepanshu (EXTERN: Embitel) as a req.
- [x] TODO: Dang, Quang Anh (EXTERN: ITK) will improve the fuzz_BucketHandler.cpp fuzz target and push the changes





14.15.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101)

14.16 2023.08.29 1st Hands-On session DiagMaster

14.16.1 Participants

Dang, Quang Anh (EXTERN: ITK) (Bi)

- [x] Muckenschnabl, David Jose (EXTERN: ITK)

DiagMaster

- [x] Soldatkin, Anton (EXTERN: LUX)
- [x] Cuozzo, Gerardo (EXTERN: LUX)
- [ ] Shashkevych, Oleksandr (EXTERN: LUX)
- [ ] Unknown User (vnd5ht9)
- [x] Unknown User (vmht2me)
- [ ] Unknown User (wkcgb7l)
- [ ] Lebedeva, Valentina (EXTERN: LUX)

14.16.2 Agenda

- Pick a volunteer from team DiagMaster to execute the fuzz test locally
- Create another simple fuzz target or work together on fuzz_DUSSkeleton.cpp
- Answer Q&A
- Showcase the bug Cifuzz found
  - E3-122187[406] - Null pointer dereference in e3_comp_diagupdate due to old version of rapidjson DONE

14.16.3 Protocol

- Soldatkin, Anton (EXTERN: LUX) said that Fuzzing will not be done by DiagMaster for 1.2
  TODO: Dang, Quang Anh (EXTERN: ITK) Discuss this with Daniel Neeb or Bernd on SCB
- Unknown User (vmht2me) volunteered to execute the fuzz test on his local machine
- The volunteer can pull and tag the docker container:
  `docker pull e3-e3sp-dockerdev.docker.devstack.vwgroup.com/master/e3_docker_ci_fuzz:1.0.2`

406 https://devstack.vwgroup.com/jira/browse/E3-122187?src=confmacro

```
docker tag e3-e3sp-dockerdev.docker.devstack.vwgroup.com/master/
e3_docker_ci_fuzz:1.0.2 cifuzz:latest
```

- Go to e3_comp_diagupdate:custom/fuzzing repo and execute:
  ./cifuzz.sh run fuzz_VehicleManifestParser --build-jobs=$(nproc)

- While waiting for build, explain what we have done with the repo sofar via:
  https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_diagupdate/compare/diff?sourceBranch=refs%2Fheads%2Fcustom%2Ffuzzing&targetRepoId=20582&targetBranch=refs%2Fheads%2Fdevelop#CMakeLists.txt
- Unknown User (vmht2me)executed the fuzz run successfully on his local machine
  - Executed debug with vscode
- Explain new build => Need another session to explain the new build
- Q&A

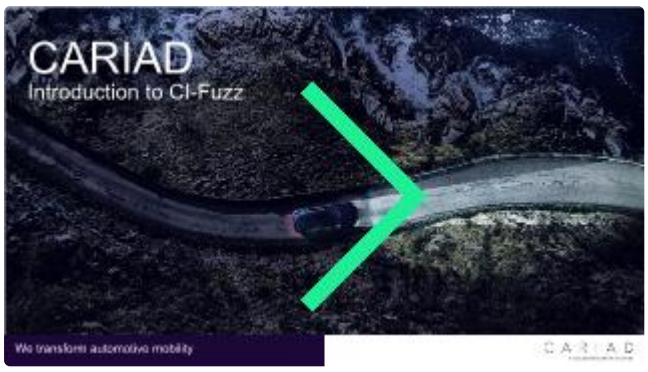
14.16.4 Q&A

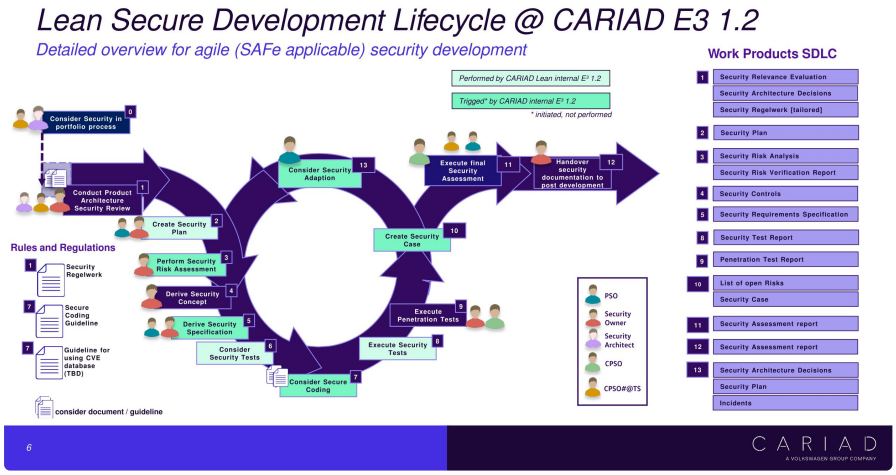
- VSCode Debug didn't work. Unknown User (vmht2me) 's debug view didn't switch to the correct view
  - Troubleshooting: Tried installing C/C++ extension
- TODO: Contact ajosties@dxc.com[407] and SM regarding whether or not DiagMaster is relevant for fuzzing

  benjamin.david@dxc.com[408]

407 mailto:ajosties@dxc.com
408 mailto:ajosties@dxc.com

14.16.5 TODOs





14.16.6 Interesting Links

- (Deprecated) CI Fuzz Integration/Training (see page 114) (Current build process for DiagMaster) - Deprecated
- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (New buildprocess)

14.17 2023.10.20 Sync MICRO STACK Fuzzing 1.1, 1.2

14.17.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [x] Perera, Yasas (TX-NP)
- [x] @weichs
- [x] Schaller, Bernd (T5-SF)

14.17.2 Agenda

- Fuzzing for MICRO Stack 1.1 and 1.2

14.17.3 Protocol

- MICRO Stack doesn't have anything in 1.1 or 1.2 which requires fuzzing
- E3 1.1 ADA , SVCP is still not in development. Nothing has started yet
  - TImeline not yet fixed
  - 1 Deliveries is planed Q2 next year
- 1.1 and 1.2 builds are similar with Bazel
  - Bazel assembles the build chain and starts the build

14.17.4 Q&A

- Problem in the VM with connectivity. What do you run this on? VW network or Boundary Client?
  - For external: Via Boundary Client and be in the VPN
  - Tutorial and documentation about set up:
  - https://devstack.vwgroup.com/confluence/display/BSWM/%5BHow+to%5D+Set+up+Development+Environment+in+Windows+and+Linux
- For support question:

  slack channel in T1 space

  #c-vwos-micro-product-topics

14.17.5 TODOs

- TODO: Something we can prepare from our side to prepare the work.
  - Maybe with vwos-microproducts/i20_base
  - Example: WUM Unit Test https://git.hub.vwgroup.com/CARIAD/vwos-micro-product/tree/master/i50_comp/Wum
    - Clone Repo
    - Execute shell.sh script => Python Virtual Environment

14.17.6 Interesting Links

- (Deprecated) CI Fuzz Integration/Training (see page 114) (Current build process for DiagMaster) - Deprecated
- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (New buildprocess)

14.18 2023.10.26 Sync OTA Master Fuzzing 1.2

14.18.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [ ] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF)

OTA Master

Unknown User (ucmeqef)

Unknown User (wv4115u)

@abramova Anna

Leonov, Nikita (EXTERN: LUX)

Alekseev, Denis (EXTERN: LUX)

Unknown User (tcytzz6)

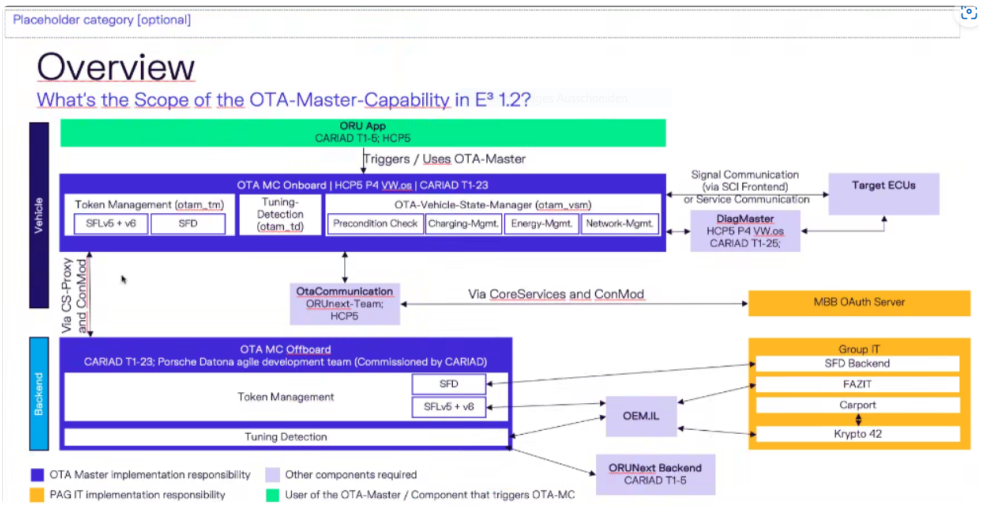
Novokshonov, Andrei (EXTERN: LUX)

Unknown User (w5mp6gu)

Ivanov, Sergei (EXTERN: LUX)

Maslennikov, Viktor (EXTERN: LUX)

14.18.2 Agenda



- Discuss the architecture of OTA Master
- Potential relevant fuzzing components:

- SCILib
- ota-vtm
- proxy (e3_comp_otam_common) => already worked with
- Interaction with ORU and Remote Repair

14.18.3 Protocol

- Heininger Tobias wants to create tasks regarding fuzzing in backlog and incorporate this in the planning of next PI
- Next steps:
  - Hands-on session with volunteer from OTA Master (Heininger Tobias will discuss with the team and find a slot for the next session)
  - Create a list of what fuzz. Rule of thumb:
    - Follow this process 1.2 - Guideline to extract a list of security relevant fuzzing items[409] or the team create a list themselves which makes sense to fuzz (usually unsanitized input from outside)
  - (Together) creating fuzz targets for that list and document effort and bugs found

14.18.4 Q&A

- Heininger, Tobias (T5-PH) The effort of writing fuzz tests? What to fuzz?
  The effort is comparable to writing an unit test, but the fuzz test can cover way more test cases.
  What to fuzz?
  => Security relevant Functions / Interfaces / Classes receiving and processing unsanitized input from outside (3rd party components, other component)
- Unknown User (ucmeqef) How long should it take to run a fuzz test?
  It depends on the fuzz test coverage or if a bug is identified.
  => If a bug is identified => then there you have your bug. The bug should be fixed and the fuzz test rerun
  => If no bug is identified, then the decision depends on a lot of factors like resources of the server, the coverage of the code you want to fuzz, how long you have run the fuzz test for. More to this 1.1 +1.2 - VW.os - Workflow of fuzzing & release criteria/definition of done[410]
- Ivanov, Sergei (EXTERN: LUX) Could it be considered a failure of a test if no bug is found?
  => Quite the opposite. If no bug is found and the fuzz test is "well-writen", it could be considered a success and the code and be considered "secure".

409 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

410 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239380/1.1+1.2+-+VW.os+-+Workflow+of+fuzzing+release+criteria+definition+of+done

14.18.5 TODOs

- [x] Heininger, Tobias (T5-PH) will come back to us with a date for the next hands-on session
- [x] Heininger, Tobias (T5-PH) will discuss among OTA-Master team who would work together with Security team to fuzz.
- [x] Dang, Quang Anh (EXTERN: ITK) will prepare the OTA Master POCs and new build and push it to the repo as preparation for the next session

14.18.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing sofar with the new buildprocess)

14.19 2023.11.14 Hands-on OTA Master Fuzzing 1.2

14.19.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [x] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF)

OTA Master

Unknown User (ucmeqef)

Abramova, Anna (EXTERN: LUX)

Ivanov, Sergei (EXTERN: LUX)

Unknown User (tcytzz6)

Novokshonov, Andrei (EXTERN: LUX)

Alekseev, Denis (EXTERN: LUX)

Heininger, Tobias (T5-PH)

Maslennikov, Viktor (EXTERN: LUX)

14.19.2 Agenda

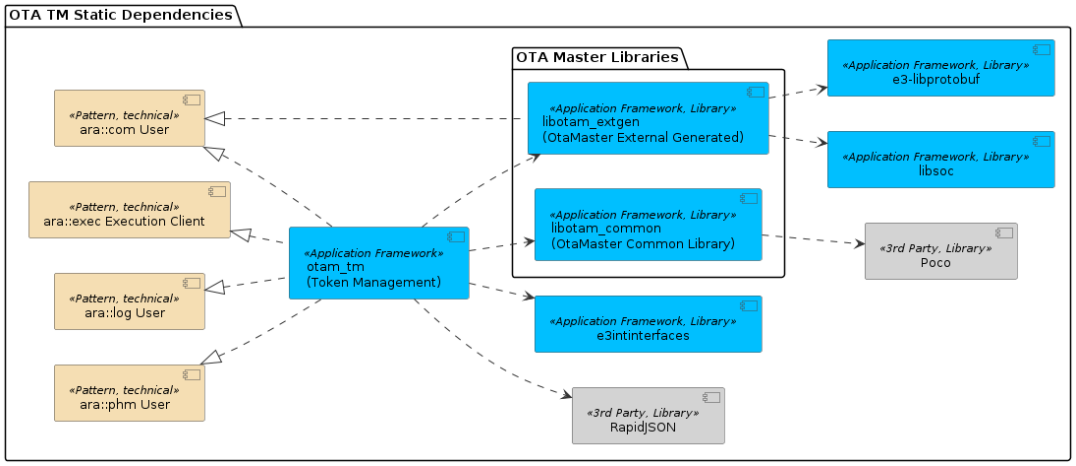
- Hands-on: Execute on the machine of a developer (Ahmed Balbaa volunteered)
- Define what to fuzz with team OTA Master

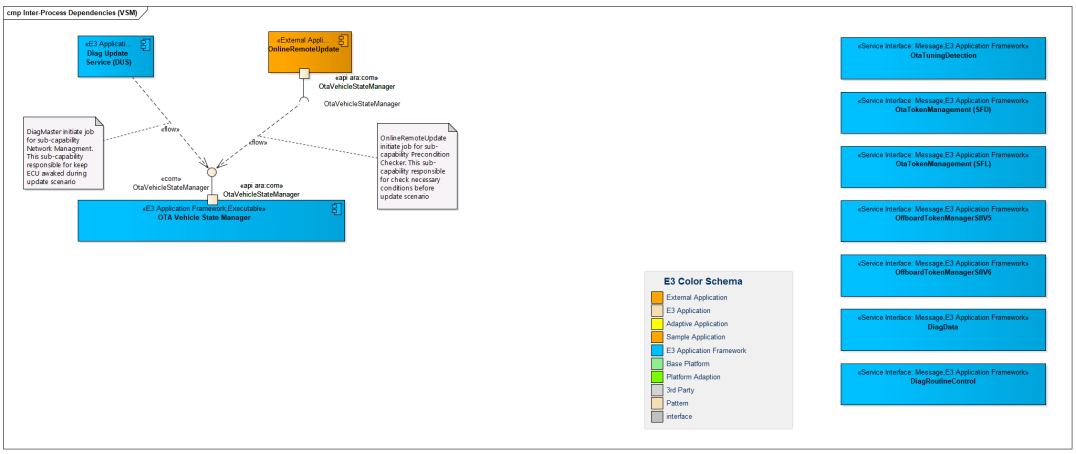
14.19.3 Protocol

- We executed :

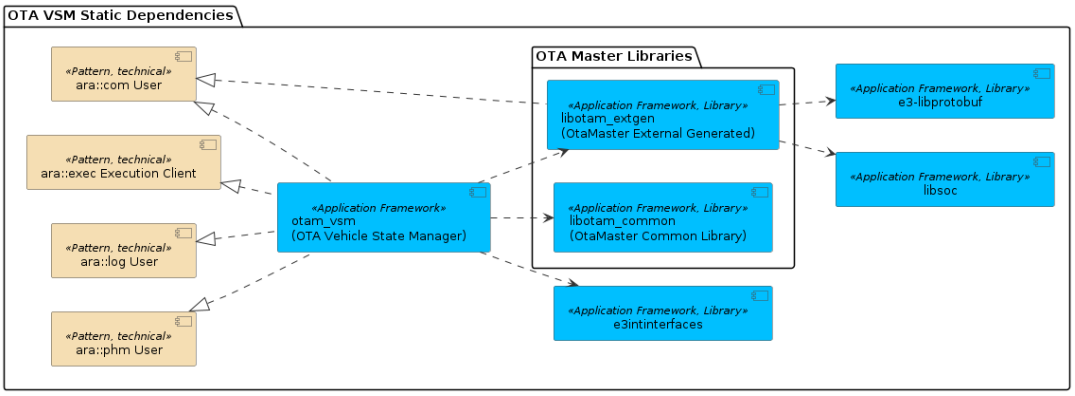
```
./build.sh --docker --target fuzzing --profile fuzzing-x86_64 --buildfolder=build
--verbose --fuzzing -j2
```

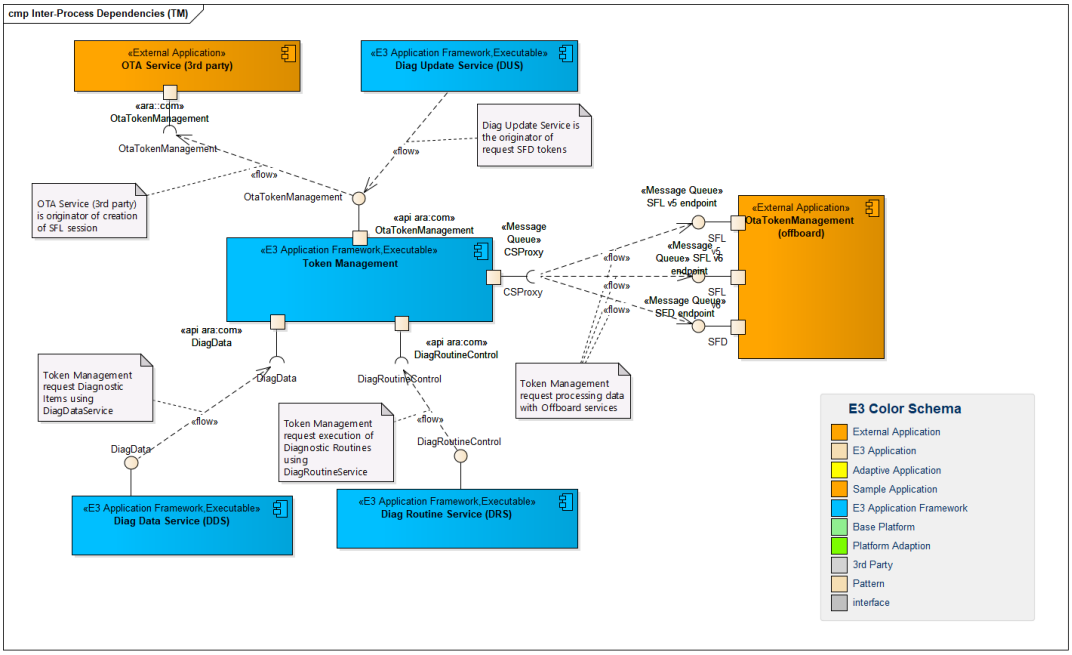
  - pulls docker container, dependencies and runs build script.
- Quang explained what was modified in the e3_comp_otam_common repo as part of a POC for fuzzing. Link to repo brach where changes where made[411]
- Quang explained risk driven fuzzing. TODO: Dang, Quang Anh (EXTERN: ITK) add Risk driven fuzzing slides.
- Structure of OTA Master was discussed
  - **Click here to see OTA Master diagrams**





411 https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_comp_otam_common/browse?at=refs%2Fheads%2Fcustom%2Fcifuzz_new_buildsystem





14.19.4 List of fuzz targets for OTA Master

Repository	Path to file	Function to fuzz	What the function does	Comments
e3_comp_otam_common	src/common/ ConfigurationSCIWrapper.cpp	createTransmittedSignal		It seems it doesn't make sense to fuzz this, since this is called by OTAM themselves.

Repository	Path to file	Function to fuzz	What the function does	Comments
"	"	getRawValue()		It seems it doesn't make sense to fuzz this, since this is called by OTAM themselves.
	"	createReceivedSignal		It seems it doesn't make sense to fuzz this, since this is called by OTAM themselves.
"	"	getReceivedSignal		It seems it doesn't make sense to fuzz this, since this is called by OTAM themselves.
"	tests/unit/SignalEventHandlerProxy_test.cpp[412]	signalHandler	This file mocks the signalHandler which provides the callback which is triggered by outside components. This callback mechanism should be fuzzed: The reference method to take a look: MOCK_METHOD(void, onEvent, (vwg::e3p::com::sci::EventType, IReceivedSignalProxy const&), (override, noexcept));	Examine how the outside component triggers this callback. Is there any parameters , unsanitized input provided?

412 https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_comp_otam_common/browse/tests/unit/common/signalproxies/SignalEventHandlerProxy_test.cpp

Repository	Path to file	Function to fuzz	What the function does	Comments
e3_comp_otam_vsm[413]	e3_comp_otam_vsm/src/precondition_checker/PreconditionCheckerSubSystem.hpp	checkPreconditionChangeJob checkPreconditionStartJob	Some functions being called from outside with some parameters specified which seem to be good target for fuzzing	Fuzzing the parameters of the funciton
"	VehicleStateManager.cpp	All	This is the APIs (Called from outside) which is only top layer file, functions that go down from here are good targets.	
e3_comp_otam_tm[414]	TokenManagementSkeletonImpl.cpp	secureVehicleDiagnosisLoginStartJob => calls: sfl::IImmobilizerSpecialFunctionLoginSubSystem::startJob	The Skeleton contains the API which is called from outside. Depends on the parameter specified, different function / service from a lower layer will be triggered.	

- Ivanov, Sergei (EXTERN: LUX) will work together with us to create fuzz targets

14.19.5 Q&A

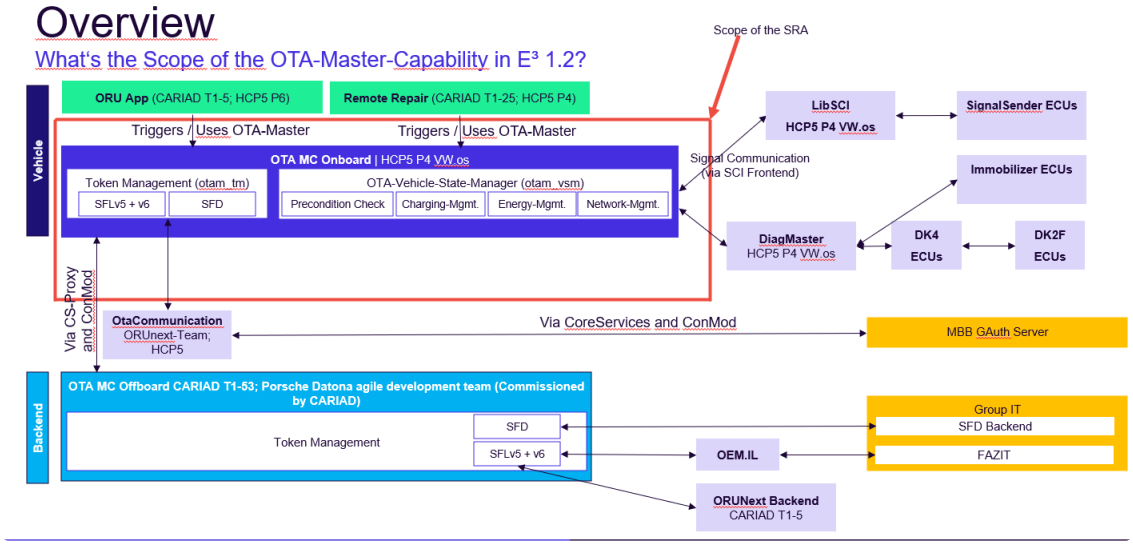
14.19.6 TODOs

- [x] Heininger, Tobias (T5-PH) will come back to us with a date for the next hands-on session
- [x] Heininger, Tobias (T5-PH) will discuss among OTA-Master team who would work together with Security team to fuzz.
- [x] Dang, Quang Anh (EXTERN: ITK) will prepare the OTA Master POCs and new build and push it to the repo as preparation for the next session
- [ ] OTA-Master team will gather a list of functions / interfaces / API which makes sense to fuzz and come back to us
- [ ] Security Team will start creating fuzz targets for the 4 candidates that we discussed today
- [ ] Organize a follow-up hands-on session to finish executing fuzz target on one of the developer's machine

413 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_otam_vsm/browse
414 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_otam_tm/browse

14.19.7 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing sofar with the new buildprocess)
- Discuss the architecture of OTA Master



14.20 2023.11.24 Sync fuzzing topic team DRM

14.20.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF)

- Belur Nataraj, Navya (EXTERN: Embitel)
- Putty, Radhika (EXTERN: Embitel)
- Yang, Juan (T5-PF)
- Lim, Meike (T5-PF)

14.20.2 Agenda

- Discussed DRM's needs on fuzzing
- Established Navya as contact person for fuzzing topic.
- Worked on building fuzzing targets with Navya

14.20.3 Protocol

14.20.4 Q&A

14.20.5 TODOs

- [x] Muckenschnabl, David Jose (EXTERN: ITK) fix this build https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_drm/browse?at=refs%2Fheads%2Fsetup_fuzz
- [x] Muckenschnabl, David Jose (EXTERN: ITK) set up meeting with Navya
- [x] Muckenschnabl, David Jose (EXTERN: ITK) Contact Meike about BSrv teams requiring fuzzing

14.20.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- E3APP013-5374[415] - [DRM] Establish CI-Fuzz Testing for all external components (follow up) DONE
- VWOS-11094[416] - Support DRM on fuzzing topic DONE
- Browse E3 APP / e3_app_DRM - DevStack Bitbucket (vwgroup.com)[417]
  Browse E3SP SubModules / e3_subm_drmcore - DevStack Bitbucket (vwgroup.com)[418]
  Browse E3SP SubModules / e3_subm_drmgateway - DevStack Bitbucket (vwgroup.com)[419]
  Browse E3SP SubModules / e3_subm_drmmethods - DevStack Bitbucket (vwgroup.com)[420]

14.21 2023.12.14 Sync Sanitizer Enabled Fuzzing

14.21.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)

---

415 https://devstack.vwgroup.com/jira/browse/E3APP013-5374?src=confmacro
416 https://devstack.vwgroup.com/jira/browse/VWOS-11094?src=confmacro
417 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_drm/browse?at=refs%2Fheads%2Fsetup_fuzz
418 https://devstack.vwgroup.com/bitbucket/projects/E3SPSUBM/repos/e3_subm_drmcore/browse?at=refs%2Fheads%2Fsetup_fuzz
419 https://devstack.vwgroup.com/bitbucket/projects/E3SPSUBM/repos/e3_subm_drmgateway/browse?at=refs%2Fheads%2Fsetup_fuzz
420 https://devstack.vwgroup.com/bitbucket/projects/E3SPSUBM/repos/e3_subm_drmmethods/browse?at=refs%2Fheads%2Fsetup_fuzz

- [x] Unknown User (ap86eaz)
- [x] Unknown User (gdhrom1)

14.21.2 Agenda

- How to build component binary with sanitizer flags enabled on the pipeline?
  Examples for LUM:
  - Cluster 6 test version: https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-components-snapshot/e3swp/lum/testing/lum_testing_6000.3.2.tar.gz
  - Cluster 41 stable version: https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-components-release/e3swp/lum/stable/lum_stable_20.2.30.tar.gz

14.21.3 Protocol

- How component testing works? Example https://devstack.vwgroup.com/bitbucket/projects/E3INT/repos/e3_mvp/browse/conanfile.py?at=refs%2Fheads%2FLUM-componentTests. We have 3 options.
  - 1) LUM package is specified => "LUM_PACKAGE = "lum/6000.3.6@e3swp/testing"" => This conan package will be pulled from artefactory and used in that component test
  - 2) Build the conan package ourselves locally and specify the name of the local package like so "lum/6000.3.6@e3swp/local"
  - 3) Build the Binary ourselves. Then login to QEMU with SSH and during runtime, change the binary to use.

14.21.4 TODOs

- Identify where the binaries are
- Try to build this locally with sanitizer flags. If successful => do it on the pipeline

14.21.5 Interesting Links

-

14.22 2024.01.08 Sync Sanitizer Enabled Fuzzing

14.22.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [x] Unknown User (ap86eaz)
- [ ] Unknown User (gdhrom1)

14.22.2 Agenda

- How to build component binary with sanitizer flags enabled on the pipeline?
  Examples for LUM:
  - Cluster 6 test version: https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-components-snapshot/e3swp/lum/testing/lum_testing_6000.3.2.tar.gz
  - Cluster 41 stable version: https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-components-release/e3swp/lum/stable/lum_stable_20.2.30.tar.gz

14.22.3 Protocol

- How component testing works? Example https://devstack.vwgroup.com/bitbucket/projects/E3INT/repos/e3_mvp/browse/conanfile.py?at=refs%2Fheads%2FLUM-componentTests. We have 3 options.
  - 1) LUM package is specified => "LUM_PACKAGE = "lum/6000.3.6@e3swp/testing"" => This conan package will be pulled from artefactory and used in that component test
  - 2) Build the conan package ourselves locally and specify the name of the local package like so "lum/6000.3.6@e3swp/local"
  - 3) Build the Binary ourselves. Then login to QEMU with SSH and during runtime, change the binary to use.
- Current approach to build the binary that Carlos knows of:
  - Build the binary with sanitizer flags and run unit test on it
  - Example of how sanitizer is built for a component:
    Higher layer depends and uses lower layer. See example of Jenkins build Pipeline:
    https://jenkins-2.ioi.imfa.hub.vwgroup.com/blue/organizations/jenkins/E3SP_SDK%2FSystem_Builds%2Faux%2F1704441642419%2Fpkgs%2Fe3inthcp3vkmsadapter/detail/e3inthcp3vkmsadapter/1/pipeline/2310
    ⚠️ Note: In the last step, a TARBALL file is built and uploaded to artefactory => Can we use this TARBALL file to do component testing?
- How to build a conan package:
  - https://devstack.vwgroup.com/bitbucket/projects/E3INT/repos/docker_e3sdk_commonpart_build_bp1/browse/files/e3sdk-cli/commands/create_conan_package.sh?at=refs%2Fheads%2Fdevelop
  - Currently, there is no --sanitizer flags for create_conan_package.sh like for build.sh as specified here E³ Standard Sanitizer configurations[421]

14.22.4 TODOs

- [ ] Dang, Quang Anh (EXTERN: ITK) Muckenschnabl, David Jose (EXTERN: ITK) discuss with Bernd and Andreas regarding component test https://devstack.vwgroup.com/bitbucket/projects/

421 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1035888190/E%C2%B3+Standard+Sanitizer+configurations

E3INT/repos/e3_mvp/browse/conanfile.py?at=refs%2Fheads%2FLUM-componentTests and binary vs conan package for component test.

14.22.5 Interesting Links

-

14.23 2024.01.11 Sync fuzzing topic team DRM (2. Part)

14.23.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF)

- Belur Nataraj, Navya (EXTERN: Embitel)
- Hohensohn, Joerg (K-DMNX/C)
- Hebbar, Kabbinale Shrikrishna (EXTERN: Embitel)
- Bhagat, Deepak (EXTERN: Embitel)
- Dam, Rajorshi (EXTERN: Embitel)
- Swamy, Rahul Raj (EXTERN: Embitel)
- Putty, Radhika (EXTERN: Embitel)
- Yang, Juan (T5-PF)
- Lim, Meike (T5-PF)

14.23.2 Agenda

- Hands-on execution of fuzz target Belur Nataraj, Navya (EXTERN: Embitel)
- David presents fuzzing presentation
- Established Navya as contact person for fuzzing topic.

14.23.3 Protocol

- David presented the introduction slide
- David explained the code and fuzzing setup via code comparison
- David showed the build, runs the fuzz command, builds the coverage binary and generated the coverage file

- Navya build was successful.
- Navya presented the screen
- my_fuzz_test_3 couldnt go through the regex check
- We created a new fuzz target live with Navya "fuzz_isStringDigit.cpp" => it works
- Heap buffer overflow error in my_fuzz_test_2
- Showed the team how to debug with GDBSERVER and debug the bug

14.23.4 Q&A

- Rajorshi Dam: What is the relevance of data and size provided by FUZZ_TEST function?
  data and size contains random data provided by the fuzzing engine => you are free to manipulate, transform the data however you want
  Good library to help transform the data is => FuzzedDataProvider (library from libfuzzer)

14.23.5 TODOs

- [ ] Maybe it makes sense for the team to download the docker container first prior to the meeting
- [ ] Execution per second is kinda low. What is the cause and how to improve this?

```
Storing generated corpus in .cifuzz-corpus/build/bin/my_fuzz_test_3
Starting from an empty corpus
  paths: 5762 - last new path: 0s - exec/s: 271 (1m10s)
```

- [ ] Ask in jourfix => Fuzzer reports a lot of new paths, but in the coverage report, there is not really a lot of paths

14.23.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
  Fuzzing Presentation to Management Slides (see page 434)
- Link to documentation on how to improve the coverage and fuzzing performance:
  https://devstack.vwgroup.com/confluence/display/E3SEC/CI-Fuzz+Proof+of+Concept#CIFuzzProofofConcept-Improvebysettingcifuzzruntimeflag (see page 78)
- E3APP013-5374[422] - [DRM] Establish CI-Fuzz Testing for all external components (follow up) DONE
- VWOS-11094[423] - Support DRM on fuzzing topic DONE
- Browse E3 APP / e3_app_DRM - DevStack Bitbucket (vwgroup.com)[424]

422 https://devstack.vwgroup.com/jira/browse/E3APP013-5374?src=confmacro
423 https://devstack.vwgroup.com/jira/browse/VWOS-11094?src=confmacro
424 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_drm/browse?at=refs%2Fheads%2Fsetup_fuzz

Browse E3SP SubModules / e3_subm_drmcore - DevStack Bitbucket (vwgroup.com)[425]
Browse E3SP SubModules / e3_subm_drmgateway - DevStack Bitbucket (vwgroup.com)[426]
Browse E3SP SubModules / e3_subm_drmmethods - DevStack Bitbucket (vwgroup.com)[427]

14.24 2024.01.12 Sync fuzzing topic team Remote Repair (1. Part)

14.24.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [x] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF)

- Gvardijan, Bernard (EXTERN: CETITEC) (Scrum Master)
- Unknown User (fqrdwlq) (Developer) **CIFuzz responsible**
- Unknown User (suzfula)

14.24.2 Agenda

- Introduction to fuzzing for those who wasn't participating in the past meeting 06 Dec 2022 and 27 Jan 2023
- In the meantime, a developer from Remote Repair can prepare the followings:

Preparation for Hands-On

`git clone ssh://git@devstack.vwgroup.com:7999/e3app/e3_app_remote_repair.git`
`git checkout custom/cifuzz_newbuild`
`./build.sh --docker --target fuzzing --profile fuzzing-x86_64 --buildfolder=build --verbose --fuzzing -j2`
`./build.sh --docker --target fuzzing --coverage --profile fuzzing-x86_64 --buildfolder=build-cov --verbose --fuzzing -j2`

- Explain the changes we have done to the code https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/compare/diff?

425 https://devstack.vwgroup.com/bitbucket/projects/E3SPSUBM/repos/e3_subm_drmcore/browse?at=refs%2Fheads%2Fsetup_fuzz
426 https://devstack.vwgroup.com/bitbucket/projects/E3SPSUBM/repos/e3_subm_drmgateway/browse?at=refs%2Fheads%2Fsetup_fuzz
427 https://devstack.vwgroup.com/bitbucket/projects/E3SPSUBM/repos/e3_subm_drmmethods/browse?at=refs%2Fheads%2Fsetup_fuzz

sourceBranch=refs%2Fheads%2Fcustom%2Fcifuzz_newbuild&targetRepoId=17332&targetBranch=refs%2Fheads%2Fdevelop#tests/CMakeLists.txt

- Show case build / run the fuzz test / create coverage and debug with GDB Server
- Discuss future activities
  - Risk Driven Fuzzing Pilot with Remote Repair
  - In order to do so, we need support from Remote Repair => A developer who is familiar with the code to
    - Gather a list of targets that make sense to fuzz
    - Support in writing more complex fuzz targets

14.24.3 Protocol

- Ask a volunteer developer to prepare the environment first to save time
- Ask the round if we need to do the presentation / introduction to fuzzing again since it has been a while
- 8 SP is planned for S24.1.IP for fuzzing of external libraries
  - Acceptance Criteria: Fuzz Testing is established for 5 external components
  - List of function exposed to outside to fuzz:
    3.2.1. openSession
    3.2.2. changeSessionType
    3.2.3. closeSession
    3.2.4. getSessionStatus
    3.2.5. resetDevice
  - [E3APP007-805] [RemoteRepair] Establish CI-Fuzz Testing - DevStack Jira (vwgroup.com)[428]
- Unknown User (fqrdwlq) is assigned for writing fuzz target. Activity can start in the IP Sprint S24.1.IP the earliest
- Build process for Remote Repair fuzz target is currently not that optimal
  - Linking fuzz target to ${STATIC_TARGET_LIB} => Build is slow since we are building everything, also the code that we don't fuzz
  - Adding the src file / include directory manually is a lot of work

14.24.4 Q&A

14.24.5 TODOs

- [ ] Dang, Quang Anh (EXTERN: ITK) set up a meeting for S24.1.IP

428 https://devstack.vwgroup.com/jira/browse/E3APP007-805

- [ ] Muckenschnabl, David Jose (EXTERN: ITK) Look into vscode preLanuchTasks or vscode running in a docker container https://stackoverflow.com/questions/43836861/how-to-run-a-command-in-visual-studio-code-with-launch-json

14.24.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
  Fuzzing Presentation to Management Slides (see page 434)
- Link to documentation on how to improve the coverage and fuzzing performance:
  https://devstack.vwgroup.com/confluence/display/E3SEC/CI-Fuzz+Proof+of+Concept#CIFuzzProofofConcept-Improvebysettingcifuzzruntimeflag (see page 78)

14.25 2024.01.15 Sync fuzzing topic TM SP

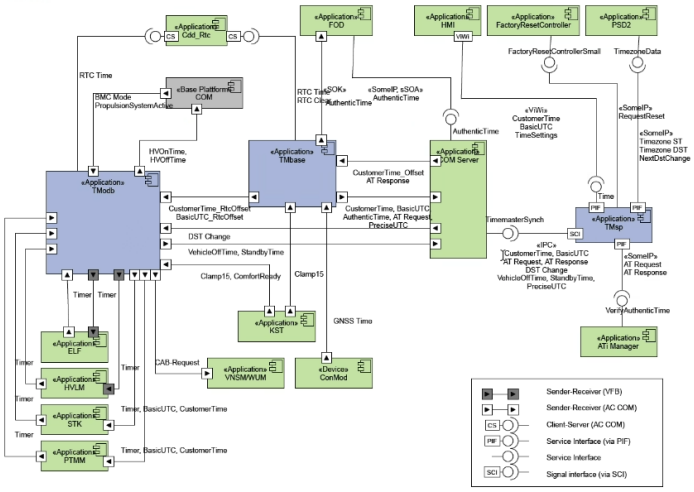
14.25.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi) (Securiy Owner Fuzzing Topic Shared Service Security Team)
- [x] Muckenschnabl, David Jose (EXTERN: ITK) (Securiy Owner Fuzzing Topic Shared Service Security Team)
- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF) (PO of Shared Service Security Team)
- [x] Dostal, Jiri (Digiteq Automotive)
- [x] Golovanov, Ilya (Digiteq Automotive)
- [x] Culik, Jan (Digiteq Automotive) (Software Test Related topic)
- [x] Ercegovcevic, Luka (EXTERN: Synchrotek)
- [x] Unknown User (f6cchom)
- [x] Pulec, Jiri (Digiteq Automotive)
- [x] Schwarz, Karel (Digiteq Automotive)
- [x] Smejkalova, Andrea (Digiteq Automotive)
- [x] Bashtannik, Oleksandr (Digiteq Automotive)
- [x] Kilik, Pavel (Digiteq Automotive)
- [x] Vastl, Zbynek (Digiteq Automotive)
- [x] Golovanov, Ilya (Digiteq Automotive) (Architect Related topic)
- [ ] Bouzek, Ondrej (Digiteq Automotive) (Architect Related topic)

14.25.2 Agenda

- Present Fuzzing
- Show example
- Understand architecture
- Obtain 3 functions for creating POCs

14.25.3 Protocol

- David presenting the management slide trimmed down for the teams
- David shows a Demo of DRM code
- Golovanov, Ilya (Digiteq Automotive) presents on TMsp side
- The BIG Time Master team consists of 3 Components (3 teams) responsible for time service and different timer.
  - TMbase
  - TMobd
  - TMsp
  - 

- The current team (TMsp) has API
  - SCI which works on:
    - Customer Time, BasiUTC, AT Request etc
  - PIF communicating with HMI for setting up customer time. This works on data such as
    - CustomerTime, BasicUTC, TimeSetting
  - PIF with PSD2 processing global time and GPS, calculating time based on time zone and location
  - PIF VerifyAuthenticTime service using for time verification
- All in 1 repository:

https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_timemaster_au416/browse
branch:

- Some POC functions for us:
  - Submodule/.../utils/<chack what they use>
  - src/TMSP/CoreModules/Services/T
  - src/TMSP/CoreModules/Services/TimeServiceModule
  - AppConfiguration.cpp => Easiest to consider fuzzing right now. Using JSON input. The team has some tests for JSON file already => Culik, Jan (Digiteq Automotive) can provide us with the example https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_timemaster_au416/browse/tests/swTests/config/test?at=refs%2Fheads%2Fdevelop

14.25.4 Q&A

- **We are also testing invalid input with unit test. In what way is fuzzing different?**
  - With UNIT test, you only test limited amount of input, scenarios. You will cover way more scenarios in a way shorter time with fuzzing
- **How does the fuzzer evaluate the fuzz target?**
  - The binary is built with Instrumentation and Sanitizer. It will get the information about coverage and Sanitizer (Memory, Address, Undefined behavior, Thread...)
- **Statistic how much time needed for one test case**
  - Currently, it is meant to be run independently and overnight. It also depends on a lot of factor:
    - the complexity of the function (fuzz test must run longer if the function is more complex).
    - function / code / line coverage of fuzz test
    - how long has the fuzz test been running?
    - how much time has passed since the last discovery of a new branch/path?
- **Is FUZZ testing planned to be automated (part of regular Triton pipeline or DSB)?**
  - Yes, it is meant to be automated on a CICD pipelien. Upon commit, push => Jenkins server will build fuzz target => Run it on CI Sense (fuzz server) => Jenkins will observe and automatically create bug tickets if any
- **Is reporting format already defined?**
  - Bug ticket will be automatically created as a JIRA ticket. There is also a Dashboard on CISense server to check all of these bugs. As for checking of coverage requirement for release, we are currently working on a solution (probably will be extracted by jenkins and reported by jenkins)
- **Time Traveler external input might be platform component.**
  - Yes, if it comes from a different component, you should consider fuzzing it

- **Is there possibility to test only public and static function or private function should be tested as well?**
  - Depends on the private function, whether or not it processes unsanitized input. If it does, then yes. Sometimes
- **What if we find bug from a different team and it is not up to our team to fix it?**
  - Create a bug ticket and propagate to the team. Depends on the criticality of the bug, the decision should be made whether or not it can be released. Rule of thumb:
    - Critical and High bugs should be fixed before release
    - Difficult decision should be propagated to the Test Management board for decisions
    - Currently, there is no hard requirements for fuzzing
- **Who should write fuzz test?**
  - Developer or Tester

14.25.5 TODOs

- [ ] Create POC and plan a hands-on session with the team

14.25.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
  - Fuzzing Presentation to Management Slides (see page 434)

14.26 2024.01.17 Sync fuzzing topic TMBase & TMObd

14.26.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF)

- Munz, Vladimir (Digiteq Automotive)
- Ivankovic, Olivera (EXTERN: Synchrotek) (Contact: from 26.01.2024 regarding all)
- Menjic, Srecko (EXTERN: Synchrotek)
- Rudaljevic, Tamara (EXTERN: Synchrotek) (Contact: from 26.01.2024 regarding all)
- Govorcin, Mihajlo (EXTERN: Synchrotek)

14.26.2 Agenda

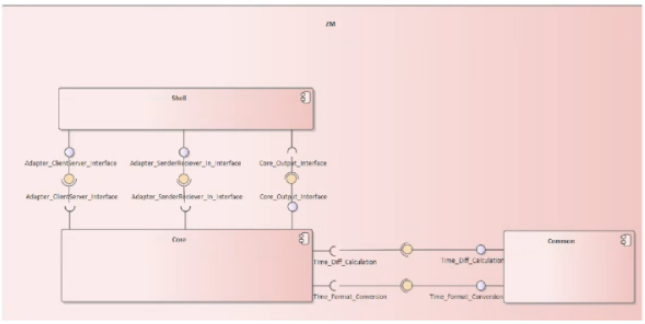
- Present Fuzzing
- Show example
- Understand architecture
- Obtain 3 functions for creating POCs

14.26.3 Protocol

- David presents trimmed Fuzzing Management Slide
- Ivankovic, Olivera (EXTERN: Synchrotek) states that they found a lot of bugs through SCA

  TM_Base and TM_OBD modules

  - Source code is separated in 3 parts:
    - Shell – Modules that handle inputs and outputs.
    - Core – Modules that handle base algorithms and functionality of component
    - Common – Some mathemathical calculations

- Cryptographic functions:
- Common Functions with less logic:
  - e3_tm_base/common/Time_Diff_Calculation.cpp
  - e3_tm_base/common/Helper_Functions.cpp
  - Some of the **common functions are affected from outside**
- Repo:
  - e3_tm_base
  - e3_tm_obd
- e3_tm_base
  - Shell Component which is accessible from outside:
    - TM_Base_Adapter_SenderReceiver_In.cpp
    - Unit tests is under shell/tests/TestAdapter_ClientServer.cpp
      e.g TCASE Adapter_ClientServer_getRTCDiagOBDWakeUp_return_value_error:
      Provides value for the component => Shell check if the expected values are set if certain mocked values are provided

        - Unit test example where the component takes the customer time:
            - handleGNSSInput => Try out this one. It doesnt have any parameters => it will, however, get the return values.
    - Unit Test from the team is not USUAL. I tests a ROUTINE step by step instead of the classic example to test 1 specific function by function
        - MOCKING:
            - Rte_ZM.cpp mocks all the return values for handleGNSSInput
- e3_tm_obd
    - Core: Contains Statemachines e.g
        - e.g Fail_Safe_State_Machine.cpp more feature complex modules

14.26.4 Q&A

- handleGNSSInput => handle GPS time

14.26.5 TODOs

- [ ] Identify which function of Common Lib is affected from outside

14.26.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
    - Sync meeting slides (see page 434)
- What to fuzz? How to create own list of fuzzing items?
  1.2 - Guideline to extract a list of security relevant fuzzing items[429]

14.27 2024.01.30 Sync fuzzing topic RPC+

14.27.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF)

RPC+ Team

429 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

Bidwe, Nitish (EXTERN: LUX)

Sasi, Arun (EXTERN: LUX)

Sreedhara, Hithesh (EXTERN: LUX)

Bin Junaidi, Muhammad Sabree (EXTERN: LUX)

Meshcheryakov, Nikita (EXTERN: LUX) (Developer Supporter)

zsolt.farkas@dxc.com[430] (not present) (System Architect - most likely take over the task to create fuzz target)

14.27.2 Agenda

- **Recap**
  - Last onboarding hands-on was 05 May 2023
  - Nitish Bidwe was the last volunteer
  - 4 SP for 3rd Sprint of SP24.1
- **Demo**
  - No need to do demo because the demo was done on 05 May 2023
- **Status**

14.27.3 Protocol

- David presents how to extract / create a list of fuzz items
  - Risk driven process
  - What to fuzz and what not to fuzz?
- The architecture of RPC+. RPC+ is a complex proxy which receives request from external components, parses and does some logic to it and then trigger the corresponding component e.g. DiagMaster
  - Main functionalitiy is **Base Line Update** functionality. Request request via interface e.g. HTTP Request.
    see more on this entry page RPC+ Entry Page regarding Diag Update[431]
    - Then acting like a proxy, it redirect the request to other component like Diag Update
    - Baseline update has state machines and can accepts a list of commands e.g.
      - initialize
      - download
      - checkPrecondition (is vehicle ready or not)
      - start (start the update when everything is ready)

430 mailto:zsolt.farkas@dxc.com
431 https://devstack.vwgroup.com/confluence/spaces/E3APP012/pages/489189524/DiagUpdate

    - OTX Functionalities like diagnostics
    - Download / Upload file from RPC+
- Prioritize Base Line Update
- Timeline
    - This Sprint RPC+ has other pipeline tasks
- Ticket to track progress and status
  https://devstack.vwgroup.com/jira/browse/E3APP012-4922

14.27.4 Q&A

- Hithesh Sreedhara - what happen after we have identified all the functions to fuzz?
  => We create fuzz targets and perform fuzzing for all those functions
- Bidwe, Nitish Nagnath - does fuzz testing discover runtime error?
  => Yes, fuzz testing specifically targets runtime error more effectively than unit test
- Hithesh Sreedhara - assuming we find some issue / bugs => How fast do we have to fix this bug?
  => Depends on the severity of the bug identified
  See 1.1 +1.2 - VW.os - Workflow of fuzzing & release criteria/definition of done[432] for more information regarding decision making
- Hithesh Sreedhara - cluster 3 is already released, if there is a critical bug, what is the process what to do?
  => The decision is up for the test management board

14.27.5 TODOs

- [ ] Update the build system of custom/cifuzz branch to the current status
- [ ] Create a list of functions to fuzz for RPC+. Contact zsolt.farkas@dxc.com[433] to ask if he can mark the list of fuzzing required functions with our new doxygen approach

14.27.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
    - Sync meeting slides (see page 434)

432 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239380/1.1+1.2+-+VW.os+-+Workflow+of+fuzzing+release+criteria+definition+of+done#id-1.1+1.2VW.osWorkflowoffuzzing&releasecriteria/definitionofdone-1.1ADA+1.2+2.0VW.osWorkflowoffuzzing&releasecriteria/definitionofdone-ReleaseCriteria

433 mailto:zsolt.farkas@dxc.com

- What to fuzz? How to create own list of fuzzing items?
  1.2 - Guideline to extract a list of security relevant fuzzing items[434]

14.28 2024.02.01 Sync fuzzing topic Lost Pilot

14.28.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF)

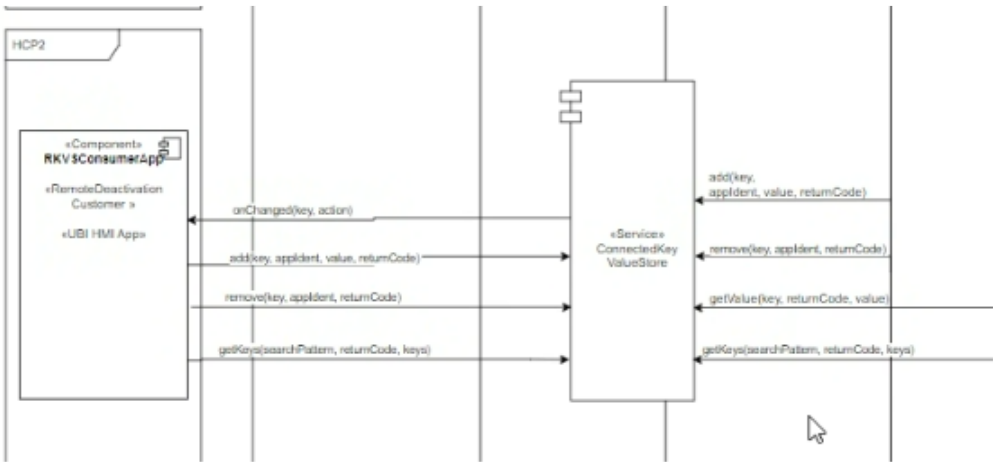
Lost Pilot Team

- Touil, Haythem (T5-PF)
- Morgenstern, Kamyar (EXTERN: Tagueri)
- Lauschkin, Maik (EXTERN: ITPower) (Responsible for fuzzing)
- Lüthen Hendrik (Responsible for fuzzing)

14.28.2 Agenda

- Recap
  - First Sync on 12 Jun 2023
  - Second sync on 16 Jun 2023
- Organisnational Topics
- Status
- Dang, Quang Anh (EXTERN: ITK) Presented fuzzing
  - Motivation
  - Comparison Fuzzing vs Unit-tests
  - Effort
  - Status with other teams
- **Restructuring Lost Pilot => 1.2**
- Next Sprint story points are planned for fuzzing.
  - Contact PO for planning
- Haythem explained the architecture
  - HCP5 RemoteKey Value Store
  - Main Goal Service for all components to store presistent data. Notify client if a change happened.

434 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items



-
  - ADD
  - REMOVE
  - GET
  - Fuzz testin only relevant for THIS EXTERNAL Interface
- 96% Unit test coverage
- e3_app_lostpilots_RKVS
  - branch = develop
  - implementation : https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_lostpilots_rkvs/browse/src/rkvs/e3p/corbos?at=refs%2Fheads%2Fdevelop
  - Unit-Test and Mocking of persistency to DB: https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_lostpilots_rkvs/browse/tests/unit/persistency?at=develop
    - Instead of writing to DB, LostPilot unit test currently write it to a file instead
  - Ticket from Lost Pilot: https://devstack.vwgroup.com/jira/browse/E3APP001-2451

14.28.3 Q&A

- **Is there a difference between Unit Tests and Fuzzing.**
  - In unit tests you are testing for functionality. In fuzzing you test for unexpected behavior/security problems.
  - It could be seen as stress-testing
- **Would fuzzing be necessary if we use a memory safe language?**
  - Yes, it would be less relevant tho, because fuzzing's main strong point is finding memory errors.

14.28.4 TODOs

- [ ] Goal is to fuzz the interface by the end of this PI.

- [ ] Lost Pilot will start working starting NEXT Sprint 4 (5SP). Next hands-on session is already set up for 13 Feb 2024

14.28.5 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
  - Sync meeting slides (see page 434)
- What to fuzz? How to create own list of fuzzing items?
  1.2 - Guideline to extract a list of security relevant fuzzing items[435]

14.29 2024.02.09 Sync fuzzing topic RPC+ regarding effort for fuzzing

14.29.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [ ] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF)

RPC+ Team

zsolt.farkas@dxc.com[436] (not present) (System Architect - most likely take over the task to create fuzz target)

Unknown User (drv1e8x) (responsible for building)

14.29.2 Agenda

- **Recap**
  - Last onboarding hands-on was 05 May 2023
  - Nitish Bidwe was the last volunteer
  - 4 SP for 3rd Sprint of SP24.1
- **Demo**
  - No need to do demo because the demo was done on 05 May 2023
- **Status**

435 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

436 mailto:zsolt.farkas@dxc.com

14.29.3 Protocol

📅 09 Feb 2024

- Farkas: Fuzz test doesnt make sense.
- Security Mode
  - ON: Everything is trusted
  - OFF: Operating in unsafe environment. Works in a container, kinda sandbox, double encrypted
    - Failure comes from 3rd party component
    - Once this is successfully decrypted
- Code Review will say that fuzz testing will fail 95% of the times, will not pass the first 3 lines.
- How does the key checking work?
  - Key in trusted storage
  - We get encrypted MOSA Container
  - Ask for decryption, signature, encryption, etc..
  - The key is preconfigured and provided by 3rd party as config and is kinda fix
  - You can upload malicious file, but **the decryption will fail if the format of the file doesn't match**
- Currently, testing:
  - QEMU mocked environment
  - Mocked functions
- RPC+ is more of a proxy and doesn't really have much logic. Does it make sense to fuzz?
  - RPC+ is a common name for 5 different orthogonal independent functions
- Issue:
  - Difficult without the backend => Can't generate random update file
  - RPC+ has issues generating encrypted input too because they don't have the key
- What can we fuzz?
  - ⚠️ Most of the functionalities has a SCHEME Validation. The input has to follow the SCHEME
  - File Download Functionalities
- 4SP for the 3rd sprint might not be up to date

14.29.4 Q&A

14.29.5 TODOs

- [x] Update the build system of custom/cifuzz branch to the current status

- [ ] Unknown User (drv1e8x) will forward a list of functions to fuzz for RPC+. Contact zsolt.farkas@dxc.com[437] to ask if he can mark the list of fuzzing required functions with our new doxygen approach
  According to 1.2 - Guideline to extract a list of security relevant fuzzing items[438] rules => Sometimes next week.

14.29.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
  - Sync meeting slides (see page 434)
- What to fuzz? How to create own list of fuzzing items?
  1.2 - Guideline to extract a list of security relevant fuzzing items[439]
- How to mark the function as fuzzing relevant with doxygen
  CI-Fuzz Proof of Concept?

14.30 2024.02.13 Hands-on creating fuzz targets for Lost Pilot

14.30.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF)

Lost Pilot Team

- Touil, Haythem (T5-PF)
- Morgenstern, Kamyar (EXTERN: Tagueri)
- Lauschkin, Maik (EXTERN: ITPower) (Responsible for fuzzing)
- Lüthen Hendrik (Responsible for fuzzing)

14.30.2 Minutes

- Maik was the volunteer to create and run the target
  - Created a new brach from develop

437 mailto:zsolt.farkas@dxc.com
438 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items
439 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

    - Used the cifuzz.sh script to setup the repo
- Quang showcased what the cifuzz.sh script does and what/how the build system works and does.
- Quang explaned how the target looks like and how the fuzzer uses the functions.
- Maik continues with the target
    - Finished setting up the repo
    - Started writing the target from where Quang got stuck.
    - fixed a problem with a flag conan was setting

14.30.3 Q&A

- how to fix: Could not find toolchain file: /opt/e3sdk/toolchain/toolchainfile_wrapper.cmake
    - in the conan.conf set cmake_toolchain_file = ""

14.30.4 TODOs

- [ ] Goal is to fuzz the interface by the end of this PI.
- [ ] Lost Pilot will start working starting NEXT Sprint 4 (5SP). Next hands-on session is already set up for 13 Feb 2024

14.30.5 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
    - Sync meeting slides (see page 434)
- What to fuzz? How to create own list of fuzzing items? 1.2 - Guideline to extract a list of security relevant fuzzing items[440]

14.31 2024.02.19 Sync fuzzing topic TM SP

14.31.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi) (Securiy Owner Fuzzing Topic Shared Service Security Team)
- [x] Muckenschnabl, David Jose (EXTERN: ITK) (Securiy Owner Fuzzing Topic Shared Service Security Team)

440 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF) (PO of Shared Service Security Team)
- [ ] Dostal, Jiri (Digiteq Automotive)
- [ ] Golovanov, Ilya (Digiteq Automotive)
- [ ] Culik, Jan (Digiteq Automotive) (Software Test Related topic)
- [x] Ercegovcevic, Luka (EXTERN: Synchrotek)
- [ ] Unknown User (f6cchom)
- [ ] Pulec, Jiri (Digiteq Automotive)
- [ ] Schwarz, Karel (Digiteq Automotive)
- [ ] Smejkalova, Andrea (Digiteq Automotive)
- [ ] Bashtannik, Oleksandr (Digiteq Automotive)
- [ ] Kilik, Pavel (Digiteq Automotive)
- [ ] Vastl, Zbynek (Digiteq Automotive)
- [x] Golovanov, Ilya (Digiteq Automotive) (Architect Related topic)
- [ ] Bouzek, Ondrej (Digiteq Automotive) (Architect Related topic)
- [x] Culik, Jan (Digiteq Automotive)
- [x] Oscadal, Marek (Digiteq Automotive)

14.31.2 Agenda

14.31.3 Protocol

- The build wasn't successful due to different style of CMake configuration.
- Golovanov, Ilya: The repository has submodules. It should be init and populated first.
  => This is done by David
- Added options for Jenkin Pods to not crash the pipeline
- Golovanov, Ilya tried the build and also had linking error
- David tried building their target with our container
- The issue was:
  - **wrong order of file**

```
set(CIFUZZ_SRC_FILES
    ${CMAKE_SOURCE_DIR}/src/TMSP/AppConfiguration.cpp // the path to the fuzz target cpp file should goes first
    fuzz_AppConfiguration.cpp
  )
```

14.31.4 Q&A

- Golovanov, Ilya: The jenkin build fails because there is a jenkin pod missing

14.31.5 TODOs

- [ ] Create POC and plan a hands-on session with the team

14.31.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
  - Fuzzing Presentation to Management Slides (see page 434)

14.32 2024.02.20 Fuzzing session with TMBase

14.32.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF)

- Munz, Vladimir (Digiteq Automotive)
- Ivankovic, Olivera (EXTERN: Synchrotek) (Contact: from 26.01.2024 regarding all)
- Rudaljevic, Tamara (EXTERN: Synchrotek) (Contact: from 26.01.2024 regarding all)
- Havlik, Jiri (Digiteq Automotive)
- Menjic, Srecko (EXTERN: Synchrotek)

14.32.2 Agenda

- hands-on fuzzing with the team

14.32.3 Protocol

20 Feb 2024

- David presented the fuzz target fuzz_Time_Diff_Calculation.cpp => fuzzed some of the utils functionalities
- Trying to create another fuzz target fuzz_TM_BASE_Adapter_SenderReceiver_In.cpp
- TMBase unit test is feature based

    - Fuzz test should rather be based on functions
- David, Olivera and Tamara trying to fix the fuzz target

14.32.4 Q&A

14.32.5 TODOs

- [ ] Muckenschnabl, David Jose (EXTERN: ITK) Meeting will be pushed to March when David is back from Vacation to continue with the fuzz targets

14.32.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
    - Sync meeting slides (see page 434)
- What to fuzz? How to create own list of fuzzing items?
  1.2 - Guideline to extract a list of security relevant fuzzing items[441]

14.33 2024.02.20 Hands-on creating fuzz targets for Lost Pilot 2

14.33.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF)

Lost Pilot Team

- Touil, Haythem (T5-PF)
- Lauschkin, Maik (EXTERN: ITPower) (Responsible for fuzzing)
- @Lüthen Hendrik (Responsible for fuzzing)

14.33.2 Minutes

- Quang pushed new changes
- Quang explained why the fuzzer was not reporting paths
    - The binary had a missing flag "-fsanitizer=fuzzer"

441 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

- Quang explained the corpus
  - It is a state save for the fuzzer to reach the maximum coverage it has registered.
- Quang explained how to build the coverage binary
  - ./build.sh --docker --target fuzzing --coverage --profile fuzzing-x86_64 --buildfolder=build-cov --verbose --fuzzing -j2
- Quang explained where he was stuck
  - He wanted to increase the coverage of the target
- Maik tried the build in his dev-environment
  - ASAN_OPTIONS=strict_string_checks=1:detect_stack_use_after_return=1:check_initialization_order=1:strict_init_order=1:abort_on_error=0:halt_on_error=0 ./build.sh --docker --target fuzzing --profile fuzzing-x86_64 --verbose --fuzzing --sanitizer=address:undefined --sanitizer-args=-fsanitize-recover=address:-fsanitize-address-use-after-scope -j3
  - Maik built and ran the target successfully
  - Maik built and generated the coverage report successfully (commands in fuzz_target file)
- Quang continued to improve the fuzz-target
- Quang pushed changes
- Maik successfully built and ran the improved target

14.33.3 Q&A

- Touil: The blue parts on the coverage mean we have executed this part of the code, can we see the output of the function under test.
  - Yes in theory you could execute the target with a specific input from the corpus, but this is not the intention of fuzzing. With fuzzing we want to test the secure execution of the code (i.e no crashes or undefined behavior). Unit test are there to check the functionality of the function.

14.33.4 TODOs

- [ ] Lauschkin, Maik (EXTERN: ITPower) & Touil, Haythem (T5-PF) will improve the target on their side and ask Dang, Quang Anh (EXTERN: ITK) is anything is not working

14.33.5 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
  - Sync meeting slides (see page 434)

- What to fuzz? How to create own list of fuzzing items?
  1.2 - Guideline to extract a list of security relevant fuzzing items[442]

14.34 2024.02.23 Hands-on creating fuzz targets for Lost Pilot 3

14.34.1 Participants

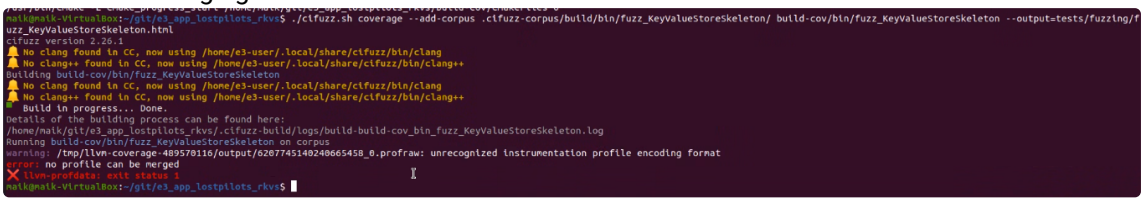
- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF)

Lost Pilot Team

- Touil, Haythem (T5-PF)
- Lauschkin, Maik (EXTERN: ITPower) (Responsible for fuzzing)
- Luethen, Hendrik, Dr. (T5-PF) (Responsible for fuzzing)

14.34.2 Minutes

- Maik had one question about why we divided
- Jenkins pod for metadata
- Issue with coverage generation



  - Solution: remove the html directory and remove build-cov file and do a clean build

14.34.3 Q&A

- Touil: The blue parts on the coverage mean we have executed this part of the code, can we see the output of the function under test.
  - Yes in theory you could execute the target with a specific input from the corpus, but this is not the intention of fuzzing. With fuzzing we want to test the secure execution of the code (i.e no crashes or undefined behavior). Unit test are there to check the functionality of the function.

442 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

14.34.4 TODOs

- [x] Lauschkin, Maik (EXTERN: ITPower) & Touil, Haythem (T5-PF) will improve the target on their side and ask Dang, Quang Anh (EXTERN: ITK) if anything is not working

14.34.5 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
  - Sync meeting slides (see page 434)
- What to fuzz? How to create own list of fuzzing items?
  1.2 - Guideline to extract a list of security relevant fuzzing items[443]

14.35 2024.03.01 Support session Lost Pilot

14.35.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [ ] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF)

Lost Pilot Team

- [x] Touil, Haythem (T5-PF)
- [x] Lauschkin, Maik (EXTERN: ITPower) (Responsible for fuzzing)
- [x] Luethen, Hendrik, Dr. (T5-PF) (Responsible for fuzzing)
- [x] Popescu, Constantin (T5-PF)

14.35.2 Minutes

- The fuzzing build on CICD pipe fails without the pod line. Constantin added the line in e3sdk_metadata
- Lost Pilot is working on new fuzz targets and more in the next PI as well
- Decision regarding branch and pipeline:
  - Currently, to not block the develop branch from building, we would keep the fuzzing activities on feature branch for now

443 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

14.35.3 Q&A

14.35.4 TODOs

14.35.5 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
  - Sync meeting slides (see page 434)
- What to fuzz? How to create own list of fuzzing items?
  1.2 - Guideline to extract a list of security relevant fuzzing items[444]

14.36 2024.03.05 Sync fuzzing topic team FlashMob (VUM)

14.36.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Hack, Christian (T5-PH) (PO - Team FlashMob)
- [x] Kulikov, Yuriy (T1-KD) (FlashMob's software engineer / QA / Team Architect) => For technical questions
- [x] Hoerner, Moritz (T1-KD) (FlashMob's Software Engineer) => For technical question

14.36.2 Agenda

- Introduction to fuzzing
- Introduction to FlashMob Code Architecture and what it does?
- Identify which FlashMob's fuzzing needs. Which interfaces / functions to fuzz?
- Access to repo / code. How to build?

14.36.3 Protocol

- Joris Bauer (2nd PO) for technical topic
- What must be done?
  - List of fuzzing items
- For FlashMob relevant:
  - File parser and processor

444 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

    - API endpoints
- Repo:
    - https://git.hub.vwgroup.com/VWOSMacro/vum
    - Readme Markdown documentation is up to date
- VUM SRA result:
    - S.1 RPC+ -> VUM :
      D.1 Vehicle Manifest:
        VTR: Moderate [FOO]
        VFG: Minor [STO]
        AUT: Moderate [NFU | STO | SRV]

      S.2 ORU -> VUM :
      D.1 Vehicle Manifest:
        VTR: Moderate [FOO]
        VFG: Minor [STO]
        AUT: Moderate [NFU | STO | SRV]

      S.5 LUM_HCP5 -> VUM :
      D.3 File Name:
        VFG: Moderate [STO | SRV]
        AUT: Moderate [STO | SRV]

      S.10 RPC+ -> VUM :
      D.4 File URL:
        VFG: Moderate [STO | SRV]
        AUT: Moderate [STO | SRV]

      S.11 ORU -> VUM :
      D.4 File URL:
        VFG: Moderate [STO | SRV]
        AUT: Moderate [STO | SRV]

      S.15 RPC+ -> VUM :
      D.5 Cryptographic Parameters:
        VTR: Moderate [FOO]
        VFG: Moderate [STO | SRV]
        AUT: Moderate [STO | SRV]

      S.16 ORU -> VUM :
      D.5 Cryptographic Parameters:
        VTR: Moderate [FOO]
        VFG: Moderate [STO | SRV]
        AUT: Moderate [STO | SRV]

S.26 DoIP Router -> VUM :
D.7 Diag Assembly Status:
  VTR: Moderate [FOO]
  VFG: Moderate [STO | SRV]
  AUT: Severe [NFU | STO | SRV | LEB]

S.30 LUM_HCP5 -> VUM :
D.8 Precondition(s) (non-safety):
  VFG: Moderate [STO | SRV]
  AUT: Severe [NFU | STO | SRV | LEB]

S.34 DoIP Router -> VUM :
D.8 Precondition(s) (non-safety):
  VFG: Moderate [STO | SRV]
  AUT: Severe [NFU | STO | SRV | LEB]

S.37 RPC+ -> VUM :
D.9 DAS Hash:
  VTR: Moderate [FOO]
  VFG: Minor [STO]
  AUT: Minor [STO]

S.38 ORU -> VUM :
D.9 DAS Hash:
  VTR: Moderate [FOO]
  VFG: Minor [STO]
  AUT: Minor [STO]

S.43 DoIP Router -> VUM :
D.10 SFD / SFL Anfragestruktur:
  VFG: Minor [STO]

S.50 ORU -> VUM :
D.12 IVD :
  VFG: Moderate [NFU | STO | SRV]
  AUT: Moderate [NFU | STO | SRV]

S.55 DoIP Router -> VUM :
D.12 IVD :
  VFG: Moderate [NFU | STO | SRV]
  AUT: Moderate [NFU | STO | SRV]

S.59 DiagAccessManager -> VUM :
D.13 Response for diagonstic access:
VFG: Minor [STO]
AUT: Minor [STO]"

- Sebastian Lutz comment:

  Das sind eigentlich so alle Interfaces zu VUM, die man beachten sollte für eine ordnungsgemäße Funktionalität.

  Am kritischsten ist wahrscheinlich TargetECUs -> DoIP Router -> VUM, da am CAN-Bus schnell mal ein manipuliertes Gerät verbunden werden kann, dass Garbage Input schickt.

  Wenn wir unsere Trust Boundary aber wirklich nur um den VUM ziehen, sollten aber auch Applikationen auf dem HCP5, die mit dem VUM kommunizieren, als tendenziell nicht vertrauenswürdig ansehen. Sprich: wir sollten schauen, dass die Interfaces die von ORU, RPC+, DiagAccessManager und LUM_HCP5 mit Inter-Prozess-Kommunikation angesprochen werden, bei Garbage-Inputs den VUM nicht abschießen oder anderweitiges Fehlverhalten verursachen.

14.36.4 Q&A

- What is the result for fuzzing?
  => List of fuzzing items
- Is it one time effort?
  => if the code doesn't change much => then yes, pretty much one time effort
- What happen if file parser code is open source lib?

14.36.5 TODOs

- [ ] Kulikov, Yuriy (T1-KD) will create a list of fuzzing items according to the rules specified here 1.2 - Guideline to extract a list of security relevant fuzzing items[445] and send it to the round

14.36.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
  Fuzzing Presentation to Management Slides (see page 434)

445 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

- Link to documentation on how to improve the coverage and fuzzing performance: https://devstack.vwgroup.com/confluence/display/E3SEC/CI-Fuzz+Proof+of+Concept#CIFuzzProofofConcept-Improvebysettingcifuzzruntimeflag (see page 78)
- Java Fuzzing example
  - Andreas example for JAVA: https://github.com/CodeIntelligenceTesting/jazzer/tree/main/examples
  - Cifuzz example for JAVA: https://github.com/CodeIntelligenceTesting/java-example-old

14.37 2024.03.06 Sync fuzzing topic TM SP

14.37.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi) (Securiy Owner Fuzzing Topic Shared Service Security Team)
- [x] Muckenschnabl, David Jose (EXTERN: ITK) (Securiy Owner Fuzzing Topic Shared Service Security Team)
- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF) (PO of Shared Service Security Team)
- [x] Dostal, Jiri (Digiteq Automotive)
- [x] Golovanov, Ilya (Digiteq Automotive)
- [x] Culik, Jan (Digiteq Automotive) (Software Test Related topic)
- [x] Ercegovcevic, Luka (EXTERN: Synchrotek)

14.37.2 Agenda

- Discussed planned time for fuzzing.
  - This sprint the topic will be continued. Next PI will have fuzzing planned into it.
- We worked on the null dereference the target was chrashing in, it was a missing logger. Fuzzerd furder and we found an assert in rapidjason

14.37.3 LIST OF FUZZ RELEVANT FUNCTIONS

List location: Adaptive 1.2 Fuzz Testing - Target list[446]

14.37.4 TODOs

- [ ] @TMSP team Create List of fuzz-relevant functions in the component TMSP
- [ ] Muckenschnabl, David Jose (EXTERN: ITK) Finish checking the target and report the bug.

446 https://devstack.vwgroup.com/confluence/spaces/E3APP004/pages/3053269142/Adaptive+1.2+Fuzz+Testing+-+Target+list

14.37.5 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
  - Fuzzing Presentation to Management Slides (see page 434)

14.38 2024.03.08 Sync fuzzing topic TMBase & TMObd

14.38.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF)

- Ivankovic, Olivera (EXTERN: Synchrotek) (Contact: from 26.01.2024 regarding all)
- Menjic, Srecko (EXTERN: Synchrotek)
- Rudaljevic, Tamara (EXTERN: Synchrotek) (Contact: from 26.01.2024 regarding all)

14.38.2 Agenda

- The team works on windows.
- It looks like the team does not have rights to pull the docker image

14.38.3 Protocol

14.38.4 Q&A

14.38.5 TODOs

- [ ] Muckenschnabl, David Jose (EXTERN: ITK) Decide if we modify the set-up so it works on windows or if the team should use a VM for fuzzing.
- [ ] Muckenschnabl, David Jose (EXTERN: ITK) Check if the team has rights for pulling docker image. Ask for usernames from the team.

14.38.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:

  - Sync meeting slides (see page 434)
- What to fuzz? How to create own list of fuzzing items?
  1.2 - Guideline to extract a list of security relevant fuzzing items[447]

14.39 2024.03.12 Sync fuzzing topic team Remote Repair (2. Part)

14.39.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [ ] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Weichslgartner, Andreas, Dr. (TX-NS)
- [x] Schaller, Bernd (T5-SF)

- Unknown User (fqrdwlq) (Developer) **CIFuzz responsible**

14.39.2 Agenda

- Tackle creating fuzz targets for the 5 interface functions
  3.2.1. openSession
  3.2.2. changeSessionType
  3.2.3. closeSession
  3.2.4. getSessionStatus
  3.2.5. resetDevice

14.39.3 Protocol

- Capacity is reduced to 3 SP (see E3APP007-805)

14.39.4 Q&A

14.39.5 TODOs

- [ ] Dang, Quang Anh (EXTERN: ITK) set up a meeting for S24.1.IP
- [ ] Muckenschnabl, David Jose (EXTERN: ITK) Look into vscode preLanuchTasks or vscode running in a docker container https://stackoverflow.com/questions/43836861/how-to-run-a-command-in-visual-studio-code-with-launch-json

447 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

14.39.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
  Fuzzing Presentation to Management Slides (see page 434)
- Link to documentation on how to improve the coverage and fuzzing performance:
  https://devstack.vwgroup.com/confluence/display/E3SEC/CI-Fuzz+Proof+of+Concept#CIFuzzProofofConcept-Improvebysettingcifuzzruntimeflag (see page 78)

14.40 2024.03.15 [DRM] Session with Rahul to clarify fuzzing related questions

14.40.1 Participants

Swamy, Rahul Raj (EXTERN: Embitel)

Muckenschnabl, David Jose (EXTERN: ITK)

14.40.2 FYI

Dang, Quang Anh (EXTERN: ITK) (Bi)

Weichslgartner, Andreas, Dr. (TX-NS)

Schaller, Bernd (T5-SF)

Belur Nataraj, Navya (EXTERN: Embitel)

Putty, Radhika (EXTERN: Embitel)

Yang, Juan (T5-PF)

Lim, Meike (T5-PF)

14.40.3 Minutes & Questions

- We went over the new Target for json parsing
- Coverage report questions
  - The command does not work -> This is the correct command
    - `/cifuzz.sh coverage --add-corpus <path to corpus directory> <path to coverage libfuzzer binary file> --output=<path to output coverage file>.html`
- Can we have multiple FUZZ_TEST functions in the target file ?

    - No, but you can call multiple functions in the FUZZ_TEST function
- How to test multiple functions efficiently?
    - We recommend a switch case for that.
- How to make the fuzzer use a particular input
    - Create a file in the .cifuzz-corpus folder with the input you want it to use.
- Can we use the gMocks for fuzzing?
    - We are working on that.
    - Example https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_comp_otam_common/browse/tests/fuzzing/fuzz_OTACommunicationCommHelper.cpp?at=custom%2Fcifuzz_2024_01
- How to suppress coverage, when functions cannot be fuzzed .i.e they do not have input.
    - No way of doing this atm.
- Can we access private variables in a class?
    - No, we should modify the fuzz_target so it itself can reach this variable.
- How to exclude Files from the coverage report, we only want relevant files for the report.
    - Remove them from the build of the coverage

14.40.4 TODOs

- [x] Muckenschnabl, David Jose (EXTERN: ITK) ADD and example to the report page of using a switch case for testing multiple functions. https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_tm_base/browse/tests/fuzzing/fuzz_Time_Diff_Calculation.cpp?at=refs%2Fheads%2Fcustom%2Ffuzzing-setup
- [x] Muckenschnabl, David Jose (EXTERN: ITK) Add how to add inputs to corpus, and how the fuzzer uses it.
- [x] Muckenschnabl, David Jose (EXTERN: ITK) Ask Bi about the gMock experience and try to make it work out.
- [x] Muckenschnabl, David Jose (EXTERN: ITK) Find out ways to modify the coverage to: suppress functions or files.
- [x] Muckenschnabl, David Jose (EXTERN: ITK) How to or can we even access private variables in a class ?

14.40.5 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Documentation of cifuzz's **FuzzedDataProvider** https://docs.code-intelligence.com/ci-fuzz/references/fuzzed-data-provider/fuzzed-data-provider-cpp

14.41 2024.03.27 [TMBase & TMObd] Setting up the dev environment for fuzzing

14.41.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Ivankovic, Olivera (EXTERN: Synchrotek)

Menjic, Srecko (EXTERN: Synchrotek)

Rudaljevic, Tamara (EXTERN: Synchrotek)

Havlik, Jiri (Digiteq Automotive)

14.41.2 FYI

Dang, Quang Anh (EXTERN: ITK)

Weichslgartner, Andreas, Dr. (TX-NS)

Schaller, Bernd (T5-SF)

14.41.3 Minutes

- Went over some known difficulties when setting up the evironment
  - All the solution of what we did can be found here: Q&A and Troubleshooting CIFuzz (see page 440)
- Shared direct link for Debugging information:
  - https://devstack.vwgroup.com/confluence/display/E3SEC/CI-Fuzz+Proof+of+Concept#Debug-process-with-script

14.41.4 Q&A

14.41.5 TODOs

- [ ] Team will work on the building of fuzz_TM_Base_Adapter_SenderReceiver_In target.

14.41.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far)
- Our introduction to fuzzing slide:
  - Sync meeting slides (see page 434)

- What to fuzz? How to create own list of fuzzing items?
  1.2 - Guideline to extract a list of security relevant fuzzing items[448]
- FAQ Q&A and Troubleshooting CIFuzz (see page 440)

14.42 2024.04.05 [TMBase & TMObd] Sync with Oliviera

14.42.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Ivankovic, Olivera (EXTERN: Synchrotek)

Rudaljevic, Tamara (EXTERN: Synchrotek)

Havlik, Jiri (Digiteq Automotive)

Dang, Quang Anh (EXTERN: ITK)

14.42.2 FYI

Weichslgartner, Andreas, Dr. (TX-NS)

Schaller, Bernd (T5-SF)

14.42.3 Minutes

- Went over news on the target
- We improved the target
- Discussed next steps
  - List will exist in the teams confluence space and we will link to it afterwards

14.42.4 Q&A

14.42.5 TODOs

- [ ] Ivankovic, Olivera (EXTERN: Synchrotek) Sync with TimeTravelers Classic part on making a list of targets for fuzzing

448 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

14.42.6 Interesting Links

E3APP004-5134[449] - [TMbase] Provide support for Fuzz test PoC implementation - Client/Server module DONE

- 1.1 ADA + 1.2 + 2.0 - Guideline to extract a list of security relevant fuzzing items[450] (How to create a list of fuzzing targets)
- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far)
- Our introduction to fuzzing slide:
  - Sync meeting slides (see page 434)
- What to fuzz? How to create own list of fuzzing items?
  1.2 - Guideline to extract a list of security relevant fuzzing items[451]
- FAQ Q&A and Troubleshooting CIFuzz (see page 440)

14.43 2024.04.11 [TMBase & TMObd] Review of Fuzz-Target List

14.43.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Ivankovic, Olivera (EXTERN: Synchrotek)

Rudaljevic, Tamara (EXTERN: Synchrotek)

Havlik, Jiri (Digiteq Automotive)

Menjic, Srecko (EXTERN: Synchrotek)

Oscadal, Marek (Digiteq Automotive)

14.43.2 FYI

Weichslgartner, Andreas, Dr. (TX-NS)

Dang, Quang Anh (EXTERN: ITK)

Schaller, Bernd (T5-SF)

449 https://devstack.vwgroup.com/jira/browse/E3APP004-5134?src=confmacro
450 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1378239468
451 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

14.43.3 Minutes

- Oliviera explained the list the team decided internally https://devstack.vwgroup.com/confluence/x/YrxQsw
- We decided to include the following, as they are used for diagnosis of the component via injecting values and seeing how it reacts
  - TM_Base_Shell.cpp
  - TM_Base_Connector.cpp
  - TM_Base_Adapter_Diagnostic.cpp

14.43.4 Q&A

14.43.5 TODOs

14.43.6 Interesting Links

✅ E3APP004-5134[452] - [TMbase] Provide support for Fuzz test PoC implementation - Client/Server module DONE

- 1.1 ADA + 1.2 + 2.0 - Guideline to extract a list of security relevant fuzzing items[453] (How to create a list of fuzzing targets)
- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far)
- Our introduction to fuzzing slide:
  - Sync meeting slides (see page 434)
- What to fuzz? How to create own list of fuzzing items?
  1.2 - Guideline to extract a list of security relevant fuzzing items[454]
- FAQ Q&A and Troubleshooting CIFuzz (see page 440)
- Target List https://devstack.vwgroup.com/confluence/x/YrxQsw

14.44 2024.04.16 Sync session regarding OTA-Master Story Points in PI 24.2

14.44.1 Participants

Dang, Quang Anh (EXTERN: ITK)

---

452 https://devstack.vwgroup.com/jira/browse/E3APP004-5134?src=confmacro
453 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1378239468
454 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

Schaller, Bernd (T5-SF)

Maslennikov, Viktor (EXTERN: LUX)

Muckenschnabl, David Jose (EXTERN: ITK)

Heininger, Tobias (T5-PH)

Neeb, Daniel (T5-SG)

14.44.2 Minutes

- Went over list of targets OTA-Master created last PI
- Tobias explained: The team is working on high prio development for Release of DiagMaster
- Idea: Shared team security work on the targets
  - Possibility: We can fuzz util functions
  - Drawbacks: We are not familiarized with Mocks and code base, it would take us a lot more effort
- Idea: Support on fuzzing critical code area of DiagMaster to minimize the amount of bugs
  - Background: Currently DiagMaster has hunderts of bugs
- DiagMaster is currently running other security tools as well such as sanitizer etc.

14.44.3 Result

- OTA-Master will be fuzzed by Shared Service Security team until further notice
- Shared Service Security Team is available to help with fuzzing DiagUpdate code, specially critical functions

14.44.4 Q&A

14.44.5 TODOs

- [ ] Dang, Quang Anh (EXTERN: ITK) Go over OTA-Master list and fuzz simple functions
- [x] Heininger, Tobias (T5-PH) will identify critical functions / code area in DiagMaster code and we will support hardening these areas to minimize #bugs

14.44.6 Interesting Links

- SNEAK PEEK Page: https://devstack.vwgroup.com/confluence/display/E3SPVMAN/%5B%21SNEAKPEEK%21%5D+Fuzzing+Function+Overview

- Information on how to extract relevant fuzzing targets: 1.2 - Guideline to extract a list of security relevant fuzzing items[455]

14.45 2024.04.17 [DRM] Sync session regarding DRM's Story Points in PI 24.2

14.45.1 Participants

Dang, Quang Anh (EXTERN: ITK)

Schaller, Bernd (T5-SF)

Muckenschnabl, David Jose (EXTERN: ITK)

Neeb, Daniel (T5-SG)

Putty, Radhika (EXTERN: Embitel)

14.45.2 Minutes

- Explained status on DRM regarding fuzzing
- Fuzzing was an option during planning
- Due to low code coverage being a priority fuzzing activities ceases for this PI
- We can use fuzzing as a way to test the negative cases while testing the code

14.45.3 Result

- We can help test the negative Test cases using fuzzing instead of unit tests

14.45.4 Q&A

14.45.5 TODOs

- [ ] Putty, Radhika (EXTERN: Embitel) Will sync with DRM team and decide if to use fuzzing as negative test coverage
- [ ] Putty, Radhika (EXTERN: Embitel) Will set up a follow up meeting for next week 22 Apr 2024

455 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

14.45.6 Interesting Links

- Information on how to extract relevant fuzzing targets: 1.2 - Guideline to extract a list of security relevant fuzzing items[456]

14.46 2024.04.22 Sync fuzzing topic team FlashMob (VUM) Part 2: Building and defining targets

14.46.1 Participants

- [x] Dang, Quang Anh (EXTERN: ITK) (Bi)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Hack, Christian (T5-PH) (PO - Team FlashMob)
- [x] Kulikov, Yuriy (T1-KD) (FlashMob's software engineer / QA / Team Architect) => For technical questions
- [x] Hoerner, Moritz (T1-KD) (FlashMob's Software Engineer) => For technical question

14.46.2 Agenda

- How to build?
- What would be good targets for fuzzing? Progress on the list of fuzz targets?

14.46.3 Protocol

- gradle.properties should be in ~/.gradle
- there are 2 Jfrogs devstack and normal one
- Code to start fuzzing:
  - https://git.hub.vwgroup.com/VWOSMacro/vum/blob/main/UpdateService/src/updateservice/flashing/DiagUpdateImpl.java implements the interface https://git.hub.vwgroup.com/VWOSMacro/vum/blob/main/UpdateService/src/updateservice/defaults/DiagUpdateStub.java
  - The test class for this is UpdateService/tests/updateservice/flashing/UdsFlashingSmallerMediumTest.kt
  - Problem: This example doesn't have much logic. Mostly passes data around
- More interesting to fuzz would be some underlying function e.g.
  - UpdateService/src/updateservice/diagnostics/manifest/UpdatePlanConverter.java "public static UpdatePlan convertVehicleManifestToUpdatePlan("
    - Receives a manifest file, parses it and then convert it

456 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

        - Example for Manifest File to feed:
            - UpdateService/tests/resources/VehicleManifest-DK2F-small.json
            - To mock LoggerFactory, just create a system out logger factory
- Organisational:
    - No capacity planned for fuzzing this PI

14.46.4 Q&A

- What is the result for fuzzing?
  => List of fuzzing items
- Is it one time effort?
  => if the code doesn't change much => then yes, pretty much one time effort
- What happen if file parser code is open source lib?

14.46.5 TODOs

- [ ] Kulikov, Yuriy (T1-KD) will create a list of fuzzing items according to the rules specified here 1.2 - Guideline to extract a list of security relevant fuzzing items[457] and send it to the round

14.46.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
  Fuzzing Presentation to Management Slides (see page 434)
- Link to documentation on how to improve the coverage and fuzzing performance:
  https://devstack.vwgroup.com/confluence/display/E3SEC/CI-Fuzz+Proof+of+Concept#CIFuzzProofofConcept-Improvebysettingcifuzzruntimeflag (see page 78)
- Java Fuzzing example
    - Andreas example for JAVA: https://github.com/CodeIntelligenceTesting/jazzer/tree/main/examples
    - Cifuzz example for JAVA: https://github.com/CodeIntelligenceTesting/java-example-old

14.47 2024.05.02 Sync fuzzing topic [TMSP]

14.47.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK) (Securiy Owner Fuzzing Topic Shared Service Security Team)

---

457 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

Culik, Jan (Digiteq Automotive) (Software Test Related topic)

14.47.2 FYI

Dang, Quang Anh (EXTERN: ITK) (Bi) (Securiy Owner Fuzzing Topic Shared Service Security Team)

Weichslgartner, Andreas, Dr. (TX-NS)

Schaller, Bernd (T5-SF)

14.47.3 Agenda

- Discussed the list and next steps
  - next steps in todo section

14.47.4 LIST OF FUZZ RELEVANT FUNCTIONS

List location: Adaptive 1.2 Fuzz Testing - Target list[458]

14.47.5 TODOs

- [ ] @TMSP continue writing fuzz tests using the list as a guide on what to fuzz next
- [ ] Muckenschnabl, David Jose (EXTERN: ITK)add example of Gmock mock to the guide page

14.47.6 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- Our introduction to fuzzing slide:
  - Fuzzing Presentation to Management Slides (see page 434)

14.48 2024.05.08 Analyse Risk of Buffer Overflow in TLR

14.48.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Schaller, Bernd (T5-SF)

Neeb, Daniel (T5-SG)

Eiser, Daniel (T5-VF)

---

458 https://devstack.vwgroup.com/confluence/spaces/E3APP004/pages/3053269142/Adaptive+1.2+Fuzz+Testing+-+Target+list

Maier, Marc (T5-VF)

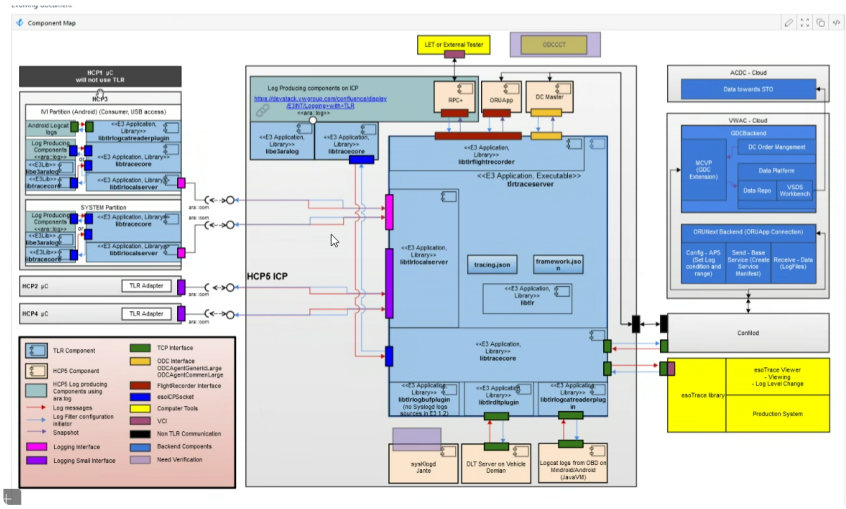
Schmon, Fabian (T5-VF)

14.48.2 FYI

Dang, Quang Anh (EXTERN: ITK) (Bi)

Weichslgartner, Andreas, Dr. (TX-NS)

14.48.3 Agenda

- This TRL lib is linked to many components, any user of the lib can trigger this
- All component using TLR
  - 

- It is an out of bounds read
- we do not have enough time to analyze the reach of this bug as Monday is late for rebuilding all TRL Components
- Can this be reached easily through HCP3
  - We would need to discuss this with someone from HCP3
- A meeting on Monday will be planned to further understand the impact from a HCP3 point of view

14.48.4 TODOs

- [x] Maier, Marc (T5-VF) Prepare Pull request to be ready for a fix

14.49 2024.05.14 [DRM] Sync session regarding DRM's Story Points in PI 24.2 Part 2

14.49.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Putty, Radhika (EXTERN: Embitel)

Dam, Rajorshi (EXTERN: Embitel)

Schaller, Bernd (T5-SF)

14.49.2 FYI

Neeb, Daniel (T5-SG)

Dang, Quang Anh (EXTERN: ITK)

14.49.3 Minutes

- Discussed Team DRMs Priorities for this PI
  - The team is working on extending Unit Test coverage. This could be supplemented with Fuzzing when the server set-up is working and we have an automated way to use the fuzzing infrastructure for regression testing.

14.49.4 Result

- We will discuss having a fuzzing epic on DRMs plans for next PI (PI24.3)

14.49.5 Q&A

14.49.6 TODOs

- [ ] Muckenschnabl, David Jose (EXTERN: ITK) Contact DRM during PI Planing for PI24.3

14.49.7 Interesting Links

- Information on how to extract relevant fuzzing targets: 1.2 - Guideline to extract a list of security relevant fuzzing items[459]

459 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

14.50 2024.05.14 Analyse Risk of Buffer Overflow in TLR for HCP3 Part 2

14.50.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Schaller, Bernd (T5-SF)

Schmon, Fabian (T5-VF)

Dang, Quang Anh (EXTERN: ITK) (Bi)

Maier, Marc (T5-VF)

Salich, Sebastian (T2-DG)

Huebner, Andreas, Dr. (T2-1)

Desai, Abhijeet (T5-VF)

14.50.2 FYI

Weichslgartner, Andreas, Dr. (TX-NS)

Neeb, Daniel (T5-SG)

Eiser, Daniel (T5-VF)

14.50.3 Analysis

A fix for this bug already exists in CL6 the analysis happened to decide the risk for supporting the decision of including the fix in CL3 or not

Link to Pull Request → https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_tlrlib/pull-requests/574/diff#src%2FMessages%2FTLRMessageCreator.hpp

Bug Fix implemented in CL6

```
inline std::shared_ptr<TLRBlobMessage>
createTLRBlobMessage(std::uint8_t const buffer[], TLRMessageLengthType const
 bufferSize, size_t& currentBufferOffset)
{
    if (bufferSize < currentBufferOffset + kAttributeHeaderSize) {
    if (bufferSize < (currentBufferOffset + kAttributeHeaderSize)) { // fix
        throw TLRParseException("Message too short for contained BLOB payload.");
    }

    TLRAttributeSizeType attributeSize{networkToLocalByteOrder<kAttributeSizeLength>(
```

```
        buffer + (currentBufferOffset + kAttributeTypeLength + kAttributeKeyLength))};
    size_t const         kReadDelta{static_cast<size_t>(attributeSize) + static_cast<si
ze_t>(kAttributeHeaderSize)};

    size_t const kReadDelta{static_cast<size_t>(attributeSize) + static_cast<size_t>(kA
ttributeHeaderSize)};

    if (bufferSize < (currentBufferOffset + kReadDelta)) {  // fix
        throw TLRParseException("Attribute size of BLOB payload exceeds message buffer
size.");
    }

    std::shared_ptr<TLRBlobMessage> result = std::make_shared<TLRBlobMessage>(
        ByteVector(buffer + (currentBufferOffset + kAttributeHeaderSize), buffer +
(currentBufferOffset + kReadDelta)),
        TLRBlobContentFormat::kTLR);

    currentBufferOffset += kReadDelta;
    currentBufferOffset += getPadding(currentBufferOffset, kAttributePadding);

    return result;
}
```

In TLRLocalServer the Message is created here

Link to source code → https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_tlrlocalserver/browse/src/RemoteServer/RemoteServer.hpp#288

Creation of message

```
template <typename LogMessageChunksDataType>
    void
    rebuildMessages(LogMessageChunksDataType logMessageChunks) noexcept
    {
        ...
        size_t                      wholeTLRMessageSize{0U}; // Size init
        ...
        for (auto const& logMessageChunk : logMessageChunks) {
            if (logMessageChunk.chunkMetaInformation.has_value()) {
                chunkMetaInformationPair.first  = kRemoteServerUuid;
                chunkMetaInformationPair.second = logMessageChunk.chunkMetaInformation-
>logMessageIndex;
                wholeTLRMessageSize             = logMessageChunk.chunkMetaInformation-
>wholeMessageSize;  // Size is acquired from MetadataInformation
                numberOfChunks                  = logMessageChunk.chunkMetaInformation-
>numberOfChunks;
                chunkIndex                      = logMessageChunk.chunkMetaInformation-
>chunkIndex;
            }
            ...
```

```
                try {
                    auto tlrMessage =
TLRMessage::fromBuffer(newMessageEntry.first.data(), wholeTLRMessageSize);  // Used to
create message here
                    logD() << "forward message To TraceCore";
                    this->forwardMessageToTraceCore(*tlrMessage);
                    wholeTLRMessageSize    = 0U;
                    numberOfChunks         = 0U;
                    chunkIndex             = 0U;
                    newMessageEntry.second = 0U;
                }
            ...
        }
    }
```

This size is also checked by TLRLocalServer to filter out inconsistencies

Link to source code → https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_tlrlocalserver/browse/src/RemoteServer/RemoteServer.hpp#405

Cheking of size perfomed by LocalServer

```
inline bool
RemoteServer::checkMessageSizeAndNumberOfChunks(std::uint16_t const
 wholeNumberOfChunks,
                                                size_t const
wholeTLRMessageSize) const
{
    // Check if the wholeTLRMessageSize is correct
    if ((wholeTLRMessageSize < 1U)) {
        logE() << "wholeTLRMessageSize is incorrect";
        return false;
    }

    // Check if the wholeNumberOfChunks is correct
    if (wholeNumberOfChunks < 1U) {
        logE() << "numbre of chunk is incorrect";
        return false;
    }

    // Check if wholeNumberOfChunks and wholeTLRMessageSize match
    if ((wholeNumberOfChunks
         != static_cast<std::uint16_t>((wholeTLRMessageSize / kChunkDataSizeMax)
                                       +
static_cast<std::uint16_t>((wholeTLRMessageSize % kChunkDataSizeMax) != 0U)))
        || ((wholeNumberOfChunks * kChunkDataSizeMax) < wholeTLRMessageSize)) {
        logE() << "numberOfChunks or wholeTLRMessageSize is incorrect";
        return false;
    }
    return true;
```

```
}
```

This alone would not give a conclusion on the risk.

It was also discussed during this meeting that a 3rd Party app in the Android environment residing in HCP3 could trigger this bug by using the logging infrastructure.

After further discussion it was concluded that the 3rd party could not control the buffer size as this is calculated internally, using the size of the message provided by the user.

In this case there could not be a difference in message size and buffer size, making this attack vector invalid.

14.50.4 Result

As the user cannot modify the buffer size that will be allocated, the risk of a user exploiting this is **very low.**

14.51 2024.06.03 [TMBase] Discussion on where to store corpus for TMBase

14.51.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Dang, Quang Anh (EXTERN: ITK)

Weichslgartner, Andreas, Dr. (TX-NS)

Pulec, Jiri (Digiteq Automotive)

Golovanov, Ilya (Digiteq Automotive)

14.51.2 FYI

Schaller, Bernd (T5-SF)

14.51.3 Minutes

- Discussed if we should store the corpus
- Explained why we should store it
  - It is needed for coverage progress of the fuzzer

14.51.4 Result

We will store the corpus in a different branch of the project repository

Corpus was removed from Pull Request

The current corpus will be stored in the following branch temporarily

https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_timemaster_au416/browse?at=refs%2Fheads%2Fcustom%2Fcl46-fuzzing_corpus

14.51.5 TODOs

- [x] Muckenschnabl, David Jose (EXTERN: ITK)Create a new branch where we store the corpus
- [x] Muckenschnabl, David Jose (EXTERN: ITK) remove the corpus from the Pull Request

14.51.6 Interesting Links

- Information on how to extract relevant fuzzing targets: 1.2 - Guideline to extract a list of security relevant fuzzing items[460]
- https://llvm.org/docs/LibFuzzer.html#corpus definition of corpus

14.52 2024.06.24 Sync with Team Remote Repair regarding upcoming task in PI 24.3

14.52.1 Participants

Dang, Quang Anh (EXTERN: ITK

Unknown User (fqrdwlq)

Unknown User (suzfula)

14.52.2 Agenda

- Discussion regarding the 16 external interfaces
  - Result extracting from Remote Repair SRA are the following sequences of data
    - S.23 Transmission of D.36 Request Data from Backend [K.9 Vehicle Connector -> K.1 Remote Repair Onboard App HCP5],
      Following data is part of D.36

460 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239468/1.2+-+Guideline+to+extract+a+list+of+security+relevant+fuzzing+items

    - D.2 Open Session - Request (Session Type, Pin, Preconditions) D.36
      D.5 Change Session Type - Request (Session ID, Session Type) D.36 **(excluded for cl6)**
      D.7 Close Session - Request (Session ID) D.36
      D.9 Get Session Status - Request API Call without Input Data D.36
      D.11 Reset Device - Request (Seesion ID, DiagNode, reset type) D.36
      D.13 Diag Data - Request Session ID, Command, diagDataKeys, diagDataKeyType,..... D.36
      D.15 Diag Events - Request (sessionId, command, diagNodes, diagNodeType, dtcStatusMask, dtcMaskRecord, resultOptions, minDeltaTriggerTime, timeout, readRequestID) D.36
      D.19 Diag Routines - Request (sessionId, command, key, routineOptions, resultOptions, timeout, readRequestID) D.36
      D.21 Diag Procedure Interpreter - Request (sessionId, command, otxPackage, additionalFiles, controlFiles, timeout, forJobID, forExecutionID, parameters, otxDocument, inData) D.36
      D.23 assemblyStatus - Request (sessionId, command, detailLevel, hashAlgorithm, diagNodeFilter, timeout, reqadRequestID) D.36 **(excluded for cl6)**
  - S.30 Transmission of D.17 Diag Entry - Request [K.9 Vehicle Connector -> K.1 Remote Repair Onboard App HCP5],
    "D.17 **not yet implemented**"
  - S.68 D.35 HMI action towards Remote Repair [K.7 -> K.1] ("Ethernet")
    - ViWi service, call_services, OTP1 service located in HCP5 (ServiceManagementODPOne)
  - S.72 Transmission of D.27 Check Preconditions Start - Response [K.2 OTA Master Capability -> K.1 Remote Repair Onboard App HCP5],
  - S.73 Transmission of D.29 Check Preconditions Poll - Response [K.2 OTA Master Capability -> K.1 Remote Repair Onboard App HCP5],
  - S.74 Transmission of D.31 Check Preconditions Stop - Response [K.2 OTA Master Capability -> K.1 Remote Repair Onboard App HCP5],
  - S.77 Transmission of D.32 serviceList [K.11 Core Services -> K.1 Remote Repair Onboard App HCP5],
  - S.80 Transmission of D.37 Configuration File JSON [K.12 HCP5 -> K.1 Remote Repair Onboard App HCP5]" (Interface "Read JSON file" somewhere)

Remote Repair documentation, here we can find the code that relates to the risk artifacts[461]

461 https://pages.git.hub.vwgroup.com/Cariad/arch-vwos-raas-published-doc/latest-develop/raas/subsystems/applications/remoterepair/remoterepair.html#_system_overview

Risk code	Code Interface
D.2	https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/events/OpenSessionRequest.hpp?at=refs%2Fheads%2Fdevelop-cl46#48
D.7	https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/events/CloseSessionRequest.hpp#47
D.9	https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/events/SessionStatusRequest.hpp#47
D.11	https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/events/ResetDeviceRequest.hpp#57
D.13	https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/gen/inc/vwg/services/diagnostic/diagdata/diagdata
D.15	https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/events/DiagEventsRequest.hpp#78
D.19	Did not find
D.21	https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/gen/inc/vwg/services/diagnostic/diagprocedureinterpreter/diagprocedureinterpreter
D.35	https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/gen/pif/includes/vwg/e3p/pif/servicemanagementodpone
D.27	
D.29	
D.31	
D.32	https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/events/ServiceListRequest.hpp#41

Risk code	Code Interface
D.37	

- **Future work for fuzzing:**
  - **Define fuzzing-relevant functions to tackle**: according to our guidelines defined 1.1 ADA + 1.2 + 2.0 - Guideline to extract a list of security relevant fuzzing items[462]
  - **Document the fuzzing relevant items with Doxygen approach** as described in In-code Documentation of Fuzzing Relevant Items[463]
  - **Write fuzz tests for this list of functions**
- Remote Repair Architecture:

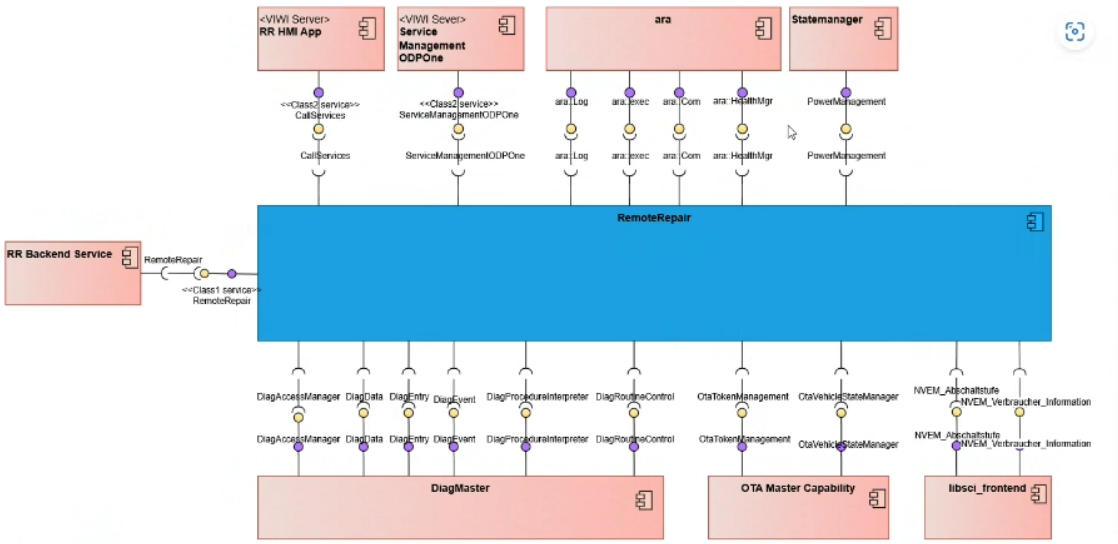


Figure 2. Remote Repair architecture

- List of all fuzz items to be tackled include:
  - List of all external interfaces and functions RemoteRepair Service (v4 - E3-107893- K-Matrix 12.08.04 / Cluster 6)[464]. (ca. 13 functions)
    Suggestion: 2-3 Hours per function => Meaning 40 hours
  - List of all call back functions or functions processing reply / response from other component.

---

462 https://devstack.vwgroup.com/confluence/pages/releaseview.action?pageId=1378239468#id-1.1ADA+1.2+2.0Guidelinetoextractalistofsecurityrelevantfuzzingitems-FuzzingRelevantCriteriafuzzing_relevant_criteria

463 https://devstack.vwgroup.com/confluence/pages/releaseview.action?pageId=1378239468#id-1.1ADA+1.2+2.0Guidelinetoextractalistofsecurityrelevantfuzzingitems-In-codeDocumentationofFuzzingRelevantItems

464 https://devstack.vwgroup.com/confluence/spaces/E3APP017/pages/1085901603/RemoteRepair+Service+v4+-+E3-107893-+K-Matrix+12.08.04+Cluster+6#RemoteRepairService(v4E3107893KMatrix12.08.04/Cluster6)-Messages

Approximately 15 callback functions
Suggestion: 2-3 Hours per function => Meaning 45 hours
- Total: 85 hours ~ 10.6 SP
  Assuming:
  - Unit tests already exist
  - No further mocks to be implemented
- Discussion of priorities:
  - 1 Priority: SAST
  - 2 Priority: Dynamic Testing e.g. Sanitizer, fuzzing
- Result of prioritization will come 01 Jul 2024

14.52.3 Minutes

14.52.4 Result

14.52.5 TODOs

- [ ] Map down from SRA to code level. Reduce the list of fuzzing items for Moderate or higher risk only
- [ ] Muckenschnabl, David Jose (EXTERN: ITK) Dang, Quang Anh (EXTERN: ITK) Prepare "left side" of linking table and make a meeting where the team can review the linkage

14.52.6 Interesting Links

- Jira Tickets involved:
  - E3APP007-1244

14.53 2024.06.28 Sync with Team [Flashmob] - Alternatives to fuzzing

14.53.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Dang, Quang Anh (EXTERN: ITK)

Weichslgartner, Andreas, Dr. (TX-NS)

Kulikov, Yuriy (T1-KD)

Kurugod, Basavaraj Kotrappa (TX-SB)

14.53.2 FYI

Schaller, Bernd (T5-SF)

14.53.3 Agenda

- Discuss activating **CodeQL** in VUM's repo - https://codeql.github.com/
  - This is a Static code analysis tool provided by Github, as it is so easy to activate and run, we think it should be used.
  - License already part of Github Advanced
- Discuss **property based testing -** https://kotest.io/docs/proptest/property-based-testing.html
  - As we cannot continue with fuzzing, we are looking at alternatives for dynamic testing.
  - Property based testing workflow
    - Write a property your function needs to comply with.
    - Use or create a generator for inputs.
    - Let the tool execute a finite amount of inputs that fit a description (the generator describes the description).
    - The tool checks if the function fulfills the property for all this inputs.
      - If it does not fulfill the property it finds the smallest/simplest input that triggers the violation
  - Example with **jqwik**: https://github.com/jqwik-team/jqwik-samples
  - **Kotest** also has its own property based testing project

14.53.4 Minutes

- Team Flashmob has tight schedule. Not to overload the team with additional work.
- Additional support for configuration of CodeQL would be appreciated
- Andreas shows a demo of CodeQL for https://git.hub.vwgroup.com/TS/secure_coding_training/security/code-scanning
  - codeql.yml file to be added to the .github/workflows/codeql.yml
- CodeQL:
  - easy to setup
  - low false-positive rate
- Jqwik is harder to integrate
- Flashmob is currently using Junit 5 which has parametized testing
  - Provide generator which provide generated permutation of input
  - Not as sophiscicated as property testing with edge cases

    - Flashmob team has limited amount of functions to use parametized (15 cases sofar) or property based testing:
        - Up to this point, not much use since VUM component mostly work with enum (#input values are limited)
        - VUM is not doing much intepreting the data. The processing of data from outside is quite "strict"
            - e.g. the JSON parser (Domain backend mbb) made sure that the scheme is correct
    - A few upcoming features can be elligible for property base testing
- Yuriy created a pull request for CodeQL https://git.hub.vwgroup.com/VWOSMacro/vum/pull/298

14.53.5 Result

14.53.6 TODOs

- [ ] Weichslgartner, Andreas, Dr. (TX-NS) Dang, Quang Anh (EXTERN: ITK) Muckenschnabl, David Jose (EXTERN: ITK) should provide support to set up CodeQL
- [ ] Onboarding/tech talk regarding property based testing

14.53.7 Interesting Links

Property based testing from **Kotest** - https://kotest.io/docs/proptest/property-based-testing.html

Github's CodeQL - https://codeql.github.com/

14.54 2024.07.08 Sync with Team [TMSP-TimeTravelers]

14.54.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Dang, Quang Anh (EXTERN: ITK)

Schaller, Bernd (T5-SF)

Lim, Meike (T5-PF)

Culik, Jan (Digiteq Automotive)

Ercegovcevic, Luka (EXTERN: Synchrotek)

14.54.2 FYI

Weichslgartner, Andreas, Dr. (TX-NS)

14.54.3 Agenda

- Short overall report and presentation about our progress from fuzz testing side
- Time of execution per testcase, define and how to configure it
- Jenkins upgrade and possibilities regarding including/excluding fuzz tests in build
- Reporting of fuzz tests

14.54.4 Minutes

- We discussed the status of the team
  - There are 4 test cases
  - they have planned to do 5 more test cases this PI
- Discussion on the test plan for fuzzing
  - When to run fuzz test
    - on every merge to develop
    - with the time of the longest pipe that is running parallel to it
    - We will only fuzz when merging into any develop branch
  - Test reporting of fuzzing
    - A new column will be added into the test report to summarize fuzzing tests
  - reporting
    - will be based on Doxygen markings
  - longer overnight fuzzing
    - this will be happening in the future

14.54.5 TODO

- [ ] Muckenschnabl, David Jose (EXTERN: ITK) communicate that developers want to be able to run fuzzing on demand on any branch

14.54.6 Links

1.1 ADA + 1.2 + 2.0 - Guideline to extract a list of security relevant fuzzing items[465]

Example of how the report looks like when Doxygen markings are in a develop branch[466]

465 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1378239468
466 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=3061303501

14.55 2024.07.10 Refinement of Targets List [TimeTravelers-TMBase]

14.55.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK) (Securiy Owner Fuzzing Topic Shared Service Security Team)

Dang, Quang Anh (EXTERN: ITK) (Bi) (Securiy Owner Fuzzing Topic Shared Service Security Team)

Oscadal, Marek (Digiteq Automotive)

Rudaljevic, Tamara (EXTERN: Synchrotek)

@Muenz, Vladimir

14.55.2 FYI

Weichslgartner, Andreas, Dr. (TX-NS)

Schaller, Bernd (T5-SF)

14.55.3 Agenda

- Discussed how the team should mark the fuzzing targets
  - Mark all the functions that have contact with external inputs
  - make sure these functions are covered by your fuzz-tests
  - All the documentation and tracking of fuzz-test will be based on the Doxygen tags

14.55.4 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- How to mark functions with Doxygen 1.1 ADA + 1.2 + 2.0 - Guideline to extract a list of security relevant fuzzing items[467]

14.56 2024.07.18 Onboarding Radu Troubleshoot conan issue

14.56.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK) (Securiy Owner Fuzzing Topic Shared Service Security Team)

Dang, Quang Anh (EXTERN: ITK) (Bi) (Securiy Owner Fuzzing Topic Shared Service Security Team)

467 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1378239468

Unknown User (e7hqlnm)

14.56.2 Agenda

- Fixing issue since Radu doesn't have conan installed locally
- Step to fix

- **How to set up conan**

```
# Install conan
# Specify ~/.conan/remote.json with following content
{
 "remotes": [
  {
   "name": "e3-release",
   "url": "https://jfrog.devstack.vwgroup.com/artifactory/api/conan/e3-e3sp-conan-
release",
   "verify_ssl": true
  },
  {
   "name": "e3-snapshot",
   "url": "https://jfrog.devstack.vwgroup.com/artifactory/api/conan/e3-e3sp-conan-
snapshot",
   "verify_ssl": true
  },
  {
   "name": "e3-app-release",
   "url": "https://jfrog.devstack.vwgroup.com/artifactory/api/conan/e3-e3sp-conan-
app-release",
   "verify_ssl": true
  },
  {
   "name": "e3-app-snapshot",
   "url": "https://jfrog.devstack.vwgroup.com/artifactory/api/conan/e3-e3sp-conan-
app-snapshot",
   "verify_ssl": true
  },
  {
   "name": "e3-vwos-production",
   "url": "https://jfrog.devstack.vwgroup.com/artifactory/api/conan/e3-e3sp-conan-
vwos-production",
   "verify_ssl": true
  },
  {
   "name": "e3-vwos-test",
   "url": "https://jfrog.devstack.vwgroup.com/artifactory/api/conan/e3-e3sp-conan-
vwos-test",
   "verify_ssl": true
  }
 ]
```

```
}

# Also add this to ~/.conan/setting.yml
    Linux:
        target: [local]
```

- Radu doesn't have access to jfrog repo e3swp3rdparty. Request access for jfrog repo here: https://devstack.vwgroup.com/jira/servicedesk/customer/portal/6/create/158

14.56.3 Interesting Links

- CI-Fuzz Proof of Concept (Guide and Documentation of process) (see page 101) (Everything about fuzzing so far with the new build-process)
- How to mark functions with Doxygen 1.1 ADA + 1.2 + 2.0 - Guideline to extract a list of security relevant fuzzing items[468]

14.57 2024.07.30 Sync with Team Remote Repair regarding Refining linking of Dataflows and code

14.57.1 Participants

Dang, Quang Anh (EXTERN: ITK

Muckenschnabl, David Jose (EXTERN: ITK)

Hartung, Martin (EXTERN: ITK)

Unknown User (fqrdwlq)

Unknown User (v2utdu9)

14.57.2 FYI

Schaller, Bernd (T5-SF)

Unknown User (suzfula)

Lutz, Sebastian (EXTERN: ITK)

14.57.3 Agenda

- Go over linking list, complete it and check if it is correct

468 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1378239468

- Remote Repair documentation[469]
- Result extracting from Remote Repair SRA are the following sequences of data
  - S.23 Transmission of D.36 Request Data from Backend [K.9 Vehicle Connector -> K.1 Remote Repair Onboard App HCP5],
    Following data is part of D.36
    - D.2 Open Session - Request (Session Type, Pin, Preconditions) D.36
      - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#184
    - D.5 Change Session Type - Request (Session ID, Session Type) D.36 **(excluded for cl6)**
      - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#197
    - D.7 Close Session - Request (Session ID) D.36
      - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#214
    - D.9 Get Session Status - Request API Call without Input Data D.36
      - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#245
    - D.11 Reset Device - Request (Seesion ID, DiagNode, reset type) D.36
      - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#231
    - D.13 Diag Data - Request Session ID, Command, diagDataKeys, diagDataKeyType,..... D.36
      - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#296
    - D.15 Diag Events - Request (sessionId, command, diagNodes, diagNodeType, dtcStatusMask, dtcMaskRecord, resultOptions, minDeltaTriggerTime, timeout, readRequestID) D.36
      - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#332
    - D.19 Diag Routines - Request (sessionId, command, key, routineOptions, resultOptions, timeout, readRequestID) D.36

469 https://pages.git.hub.vwgroup.com/Cariad/arch-vwos-raas-published-doc/latest-develop/raas/subsystems/applications/remoterepair/remoterepair.html#_system_overview

    - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#370
  - D.21 Diag Procedure Interpreter - Request (sessionId, command, otxPackage, additionalFiles, controlFiles, timeout, forJobID, forExecutionID, parameters, otxDocument, inData) D.36
    - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#398
  - D.23 assemblyStatus - Request (sessionId, command, detailLevel, hashAlgorithm, diagNodeFilter, timeout, reqadRequestID) D.36 **(excluded for cl6)**
    - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#262
- S.30 Transmission of D.17 Diag Entry - Request [K.9 Vehicle Connector -> K.1 Remote Repair Onboard App HCP5],
  "D.17 **not yet implemented**"
  - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#352
- S.68 D.35 HMI action towards Remote Repair [K.7 -> K.1] ("Ethernet")
  - ViWi service, call_services, OTP1 service located in HCP5 (ServiceManagementODPOne)
    - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/proxy/HMICommunicator.hpp?at=refs%2Fheads%2Fdevelop#101
    - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/proxy/HMICommunicator.hpp?at=refs%2Fheads%2Fdevelop#113
- S.72 Transmission of D.27 Check Preconditions Start - Response [K.2 OTA Master Capability -> K.1 Remote Repair Onboard App HCP5],
  - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/proxy/OtaVSMCommunicator.hpp?at=refs%2Fheads%2Fdevelop-cl46#123
- S.73 Transmission of D.29 Check Preconditions Poll - Response [K.2 OTA Master Capability -> K.1 Remote Repair Onboard App HCP5],
  - **Not present**
- S.74 Transmission of D.31 Check Preconditions Stop - Response [K.2 OTA Master Capability -> K.1 Remote Repair Onboard App HCP5],
  - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/proxy/OtaVSMCommunicator.hpp?at=refs%2Fheads%2Fdevelop-cl46#134

- S.77 Transmission of D.32 serviceList [K.11 Core Services -> K.1 Remote Repair Onboard App HCP5],
  - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/proxy/CoreServicesGreyCommunicator.hpp?at=develop#141
- S.80 Transmission of D.37 Configuration File JSON [K.12 HCP5 -> K.1 Remote Repair Onboard App HCP5]" (Interface "Read JSON file" somewhere)
  - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/utils/Configuration.hpp?at=refs%2Fheads%2Fdevelop#69

14.57.4 Minutes

- Went over list and found code related to fuzz-relevant Data Flows
- Clarified how to link interfaces to fuzz in code with req{} tag. This should be linked to APP Requirement, not Stakeholder Requirement.
  Suggestion for APP requirements:
  2024-02-14 Meeting Minutes: How to write Fuzzing-SW/Component Requriments[470]

14.57.5 TODOs

- [ ] Map down from SRA to code level. Reduce the list of fuzzing items for Moderate or higher risk only
- [x] Mark the new interfaces with doxygen tag @e3fuzz
- [ ] Link the requirement with req{} tag once we have APP requirement for Remore Repair as suggested here 2024-02-14 Meeting Minutes: How to write Fuzzing-SW/Component Requriments[471]

14.57.6 Interesting Links

- Jira Tickets involved:
  - E3APP007-1244
  - E3APP007-1469

14.57.7 Information form previous meetings

- **Future work for fuzzing:**

470 https://devstack.vwgroup.com/confluence/spaces/E3APP017/pages/2732207314/2024-02-14+Meeting+Minutes+How+to+write+Fuzzing-SW+Component+Requriments
471 https://devstack.vwgroup.com/confluence/spaces/E3APP017/pages/2732207314/2024-02-14+Meeting+Minutes+How+to+write+Fuzzing-SW+Component+Requriments

  - **Define fuzzing-relevant functions to tackle**: according to our guidelines defined 1.1 ADA + 1.2 + 2.0 - Guideline to extract a list of security relevant fuzzing items[472]
  - **Document the fuzzing relevant items with Doxygen approach** as described in In-code Documentation of Fuzzing Relevant Items[473]
  - **Write fuzz tests for this list of functions**
- Remote Repair Architecture:

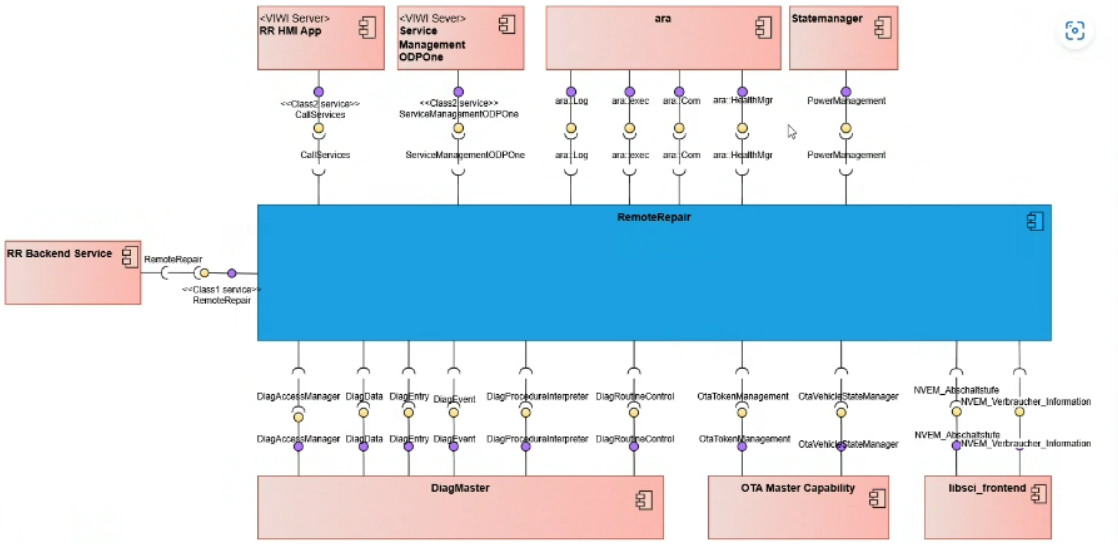


Figure 2. Remote Repair architecture

- List of all fuzz items to be tackled include:
  - List of all external interfaces and functions RemoteRepair Service (v4 - E3-107893- K-Matrix 12.08.04 / Cluster 6)[474]. (ca. 13 functions)
    Suggestion: 2-3 Hours per function => Meaning 40 hours
  - List of all call back functions or functions processing reply / response from other component.
    Approximately 15 callback functions
    Suggestion: 2-3 Hours per function => Meaning 45 hours
  - Total: 85 hours ~ 10.6 SP
    Assuming:
    - Unit tests already exist
    - No further mocks to be implemented
- Discussion of priorities:

472 https://devstack.vwgroup.com/confluence/pages/releaseview.action?pageId=1378239468#id-1.1ADA+1.2+2.0Guidelinetoextractalistofsecurityrelevantfuzzingitems-FuzzingRelevantCriteriafuzzing_relevant_criteria

473 https://devstack.vwgroup.com/confluence/pages/releaseview.action?pageId=1378239468#id-1.1ADA+1.2+2.0Guidelinetoextractalistofsecurityrelevantfuzzingitems-In-codeDocumentationofFuzzingRelevantItems

474 https://devstack.vwgroup.com/confluence/spaces/E3APP017/pages/1085901603/RemoteRepair+Service+v4+-+E3-107893-+K-Matrix+12.08.04+Cluster+6#RemoteRepairService(v4E3107893KMatrix12.08.04/Cluster6)-Messages

    - 1 Priority: SAST
    - 2 Priority: Dynamic Testing e.g. Sanitizer, fuzzing
- Result of prioritization will come 01 Jul 2024

14.58 2024.08.20 Sync with Team Remote Repair Fuzzing Requirement

14.58.1 Participants

Dang, Quang Anh (EXTERN: ITK

Hartung, Martin (EXTERN: ITK)

Veziryan, Victoria (T5-SG)

Unknown User (uw9yiis)

Gvardijan, Bernard (EXTERN: CETITEC)

Unknown User (suzfula)

14.58.2 FYI

Lutz, Sebastian (EXTERN: ITK)

Muckenschnabl, David Jose (EXTERN: ITK)

14.58.3 Information

- Go over linking list, complete it and check if it is correct
- Remote Repair documentation[475]
- Result extracting from Remote Repair SRA are the following sequences of data
    - S.23 Transmission of D.36 Request Data from Backend [K.9 Vehicle Connector -> K.1 Remote Repair Onboard App HCP5],
      Following data is part of D.36
        - D.2 Open Session - Request (Session Type, Pin, Preconditions) D.36
          https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#184
        - D.5 Change Session Type - Request (Session ID, Session Type) D.36 **(excluded for cl6)**

475 https://pages.git.hub.vwgroup.com/Cariad/arch-vwos-raas-published-doc/latest-develop/raas/subsystems/applications/remoterepair/remoterepair.html#_system_overview

- https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#197

- D.7 Close Session - Request (Session ID) D.36
  - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#214
- D.9 Get Session Status - Request API Call without Input Data D.36
  - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#245
- D.11 Reset Device - Request (Seesion ID, DiagNode, reset type) D.36
  - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#231
- D.13 Diag Data - Request Session ID, Command, diagDataKeys, diagDataKeyType,….. D.36
  - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#296
- D.15 Diag Events - Request (sessionId, command, diagNodes, diagNodeType, dtcStatusMask, dtcMaskRecord, resultOptions, minDeltaTriggerTime, timeout, readRequestID) D.36
  - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#332
- D.19 Diag Routines - Request (sessionId, command, key, routineOptions, resultOptions, timeout, readRequestID) D.36
  - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#370
- D.21 Diag Procedure Interpreter - Request (sessionId, command, otxPackage, additionalFiles, controlFiles, timeout, forJobID, forExecutionID, parameters, otxDocument, inData) D.36
  - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#398
- D.23 assemblyStatus - Request (sessionId, command, detailLevel, hashAlgorithm, diagNodeFilter, timeout, reqadRequestID) D.36 **(excluded for cl6)**
  - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#262

- S.30 Transmission of D.17 Diag Entry - Request [K.9 Vehicle Connector -> K.1 Remote Repair Onboard App HCP5],
  "D.17 **not yet implemented**"
  - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop#352
- S.68 D.35 HMI action towards Remote Repair [K.7 -> K.1] ("Ethernet")
  - ViWi service, call_services, OTP1 service located in HCP5 (ServiceManagementODPOne)
    - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/proxy/HMICommunicator.hpp?at=refs%2Fheads%2Fdevelop#101
    - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/proxy/HMICommunicator.hpp?at=refs%2Fheads%2Fdevelop#113
- S.72 Transmission of D.27 Check Preconditions Start - Response [K.2 OTA Master Capability -> K.1 Remote Repair Onboard App HCP5],
  - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/proxy/OtaVSMCommunicator.hpp?at=refs%2Fheads%2Fdevelop-cl46#123
- S.73 Transmission of D.29 Check Preconditions Poll - Response [K.2 OTA Master Capability -> K.1 Remote Repair Onboard App HCP5],
  - **Not present**
- S.74 Transmission of D.31 Check Preconditions Stop - Response [K.2 OTA Master Capability -> K.1 Remote Repair Onboard App HCP5],
  - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/proxy/OtaVSMCommunicator.hpp?at=refs%2Fheads%2Fdevelop-cl46#134
- S.77 Transmission of D.32 serviceList [K.11 Core Services -> K.1 Remote Repair Onboard App HCP5],
  - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/proxy/CoreServicesGreyCommunicator.hpp?at=develop#141
- S.80 Transmission of D.37 Configuration File JSON [K.12 HCP5 -> K.1 Remote Repair Onboard App HCP5]" (Interface "Read JSON file" somewhere)
  - [x] https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/utils/Configuration.hpp?at=refs%2Fheads%2Fdevelop#69

14.58.4 Minutes

- Suggestion:

    - Stakeholder
      SW-component's interfaces must be hardened by appropriate measures to support robustness of the software and functional availability.
    - Software (APP) Requirement:
      The robustness of the software interface which processes data item D.36 Request Data from Backend (refering to Security Risk Analysis) must be verified by fuzz testing according to fuzz testing's release criteria (1.1 +1.2 - VW.os - Workflow of fuzzing & release criteria/definition of done[476]).
- Implementation:
  - [ ] We should create 17 Software Requirements for each of the Data Point which needs to be fuzzed as the result of SRA risks and linkage to code level:
    - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/RRCommunicator.hpp?at=refs%2Fheads%2Fdevelop-cl46#177,190,207,224,239,256,291,328,350,376,405
    - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/proxy/CoreServicesGreyCommunicator.hpp?at=refs%2Fheads%2Fdevelop-cl46#149
    - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/proxy/HMICommunicator.hpp?at=refs%2Fheads%2Fdevelop-cl46#109,122
    - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/core/proxy/OtaVSMCommunicator.hpp?at=refs%2Fheads%2Fdevelop-cl46#122,134
    - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse/src/utils/Configuration.hpp?at=refs%2Fheads%2Fdevelop-cl46#68

14.58.5 TODOs

- [ ] TODO

14.58.6 Interesting Links

- Jira Tickets involved:
  - E3APP007-1244
  - E3APP007-1469

476 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1378239380/1.1+1.2+-+VW.os+-+Workflow+of+fuzzing+release+criteria+definition+of+done

14.59 2024.09.20 Fuzzing Activites recap for 24.3

14.59.1 Remote Repair

- Risk driven fuzzing pilot carried out
  - Markings in develop-cl46
  - Got 17 functions from risk analysis that required fuzzing
  - Fuzzed the 17 functions
  - VWOS-15070[477] - Pilot risk driven fuzzing carried out with RemoteRepair DONE E3APP 007-1168

14.59.2 TM Base

- Some markings here https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_tm_base/browse/TM_Base/Shell/TM_Base_Adapter_ClientServer.h#124,145,173
- Tickets created but not acted upon
  E3APP004-5498[478] - [TM-Classic] CI-Fuzz Testing for TMBase/TMOBD - R24.4 - Part 1 DONE

14.59.3 TM OBD

- No markings
- No Tickets created as of now

14.59.4 TM SP

- Marking here https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_timemaster_au416/commits/820569e5d0eb445fb959077565b8f5f2030eb193#doc%2Fdoxygen%2FSWP-DOC-CONFIG.dox
- Tickets created and work started on several fuzz tests
  E3APP004-5483[479] - [TM-Adaptive] DAST update/maintenance R24.3 DONE

14.59.5 LostPilot

- Markings https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_lostpilots_rkvs/pull-requests/634/overview

477 https://devstack.vwgroup.com/jira/browse/VWOS-15070?src=confmacro
478 https://devstack.vwgroup.com/jira/browse/E3APP004-5498?src=confmacro
479 https://devstack.vwgroup.com/jira/browse/E3APP004-5483?src=confmacro

14.59.6 Team Adaptive

- Markings and tests implemented as seen in ticket
  E3-146393[480] - [POC] Implement Fuzz Tests DONE

14.59.7 RPC+

- Ticket created but not acted upon
  E3APP012-6128[481] - [RPC+]DAST update/maintenance - PI24.3 COMMITTED

14.60 2024.09.30 Discuss E3-122194 - DiagMaster Risk

14.60.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Heininger, Tobias (T5-PH)

Schaller, Bernd (T5-SF)

Neeb, Daniel (T5-SG)

14.60.2 Agenda

- What components are relevant?
- Who is responsible for these components?

14.60.3 Minutes

- Synced situation
- Discussed who is responsible for the components
- PM of Update components is contact person for this situation
  marcus.meyerhoefer@cariad.technology[482]
- Heininger, Tobias (T5-PH) is responsible diag_update
  - The software is active in autos cl3, might not be active in the future
- The **Diagnosis** components from DiagMaster will keep existing in the future
  - diag_procedure_interpreter_service Unknown User (wycxwma)
  - diag_routine_service Unknown User (wycxwma)
  - diag_data_service Unknown User (wycxwma)

480 https://devstack.vwgroup.com/jira/browse/E3-146393?src=confmacro
481 https://devstack.vwgroup.com/jira/browse/E3APP012-6128?src=confmacro
482 mailto:marcus.meyerhoefer@cariad.technology

    - diag_access_manager Unknown User (wycxwma)
- The **Update** components will be replaced by VUM java code
    - diag_update Heininger, Tobias (T5-PH)
    - diag_update_slave_uds Heininger, Tobias (T5-PH)
    - ota_master token management (partially) Heininger, Tobias (T5-PH)
        - test planned

14.61 2024.10.07 Discuss E3-122194 - DiagMaster Risk

14.61.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Neeb, Daniel (T5-SG)

Meyerhoefer, Marcus, Dr. (T5-PH)

14.61.2 FYI

Schaller, Bernd (T5-SF)

Heininger, Tobias (T5-PH)

Dang, Quang Anh (EXTERN: ITK)

14.61.3 Agenda

- Discuss Risk
- What are the following actions?

14.61.4 Minutes

- Gave background of Risk to Markus
- The **Diagnosis** components from DiagMaster will keep existing in the future Eberspaecher, Michael (T5-PH
    - **Next Steps** discuss with Team DiagNinjas about what inputs should be tested. Koese, Muhammed (T5-PH)
    - diag_procedure_interpreter_service (prio)
    - diag_routine_service
    - diag_data_service
    - diag_access_manager
- The **Update** components will be replaced by VUM java code Heininger, Tobias (T5-PH)

- Efforts in the Update components should not be priority as it will be replaced
- Following components will be replaced by JAVA in the future, hence, the priority is low for now
  - diag_update
  - diag_update_slave_uds
  - ota_master token management (partially)
    - test planned

14.61.5 TODO

- [ ] Muckenschnabl, David Jose (EXTERN: ITK) Send Mail to Koese, Muhammed (T5-PH) for a sync meeting

14.62 2024.10.15 SW Requirement interface-hardening / Fuzz Testing Remote Repair

participants:

- Dang, Quang Anh (EXTERN: ITK)
- Schaller, Bernd (T5-SF)
- Hoche, Matthias (T5-PF)
- Unknown User (uw9yiis)
- Lutz, Sebastian (EXTERN: ITK)
- Hartung, Martin (EXTERN: ITK)

- E3APP007-1584
- Proposal (One SW for all D.x):
  - The implementation of the interfaces which process data items whose availability is critical to assure remote repair functionality (referring to Security Risk Analysis (SRA)) shall properely handle invalid input.
  - Alternativ:
    - shall handle invalid input with proper error handling method.
- Proposal for each data item D.x (decided way to go: see 2024.08.20 Sync with Team Remote Repair Fuzzing Requirement (see page 318))
  - **The implementation of the interfaces which process data item "D.2 Open Session - Request" whose availability is critical to assure remote repair functionality (referring to Security Risk Analysis (SRA)) shall properely handle invalid input.**

- Special case:
  - Config File "D.37 JSON Config File" read from system

- How to test the requirement and what to put into "Verification Criteria level" field in codebeamer
  - Proposal 2:
    - Fuzz-Test is part of dynamic test framework
      - If Fuzz-test is "green" → Requirement gets "green"
  - Proposal 1:
    - Tagged as "Test by review" (or similiar): Review protocol should contain link to or documentation about successful fuzz test. A successful fuzz-test report shows correct handling of invalid input.

14.63 2024.10.24 Sync with Team DiagNinja regarding fuzzing

14.63.1 Participants

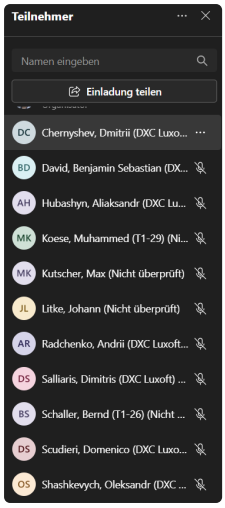
Dang, Quang Anh (EXTERN: ITK)

Schaller, Bernd (T5-SF)

Kutscher, Max (EXTERN: PTG)

Litke, Johann (EXTERN: Bertrandt)

Chernyshev, Dmitrii (DXC Luxoft) (Contact for questions about code and unit tests)



14.63.2 FYI

Developers from DiagNinjas

14.63.3 Agenda

- Short presentation to Fuzzing?

- Answer the question: What is to be fuzzed for DiagNinja?
  - SRA Fuzzing recommendation?
  - Repos for:
    - diag_procedure_interpreter_service (prio)
    - diag_routine_service
    - diag_data_service

14.63.4 Minutes

- DiagNinjas responsible for following components/functionalities:
  - https://devstack.vwgroup.com/confluence/display/E3DIAG/DiagMaster+Services%27+Contacts?src=contextnavpagetreemode
- DiaGate
  - DiagMaster Component's Contacts[483]
- Presentation of fuzzing
- Q&A:
  - Is there overview for fuzzing result => https://devstack.vwgroup.com/confluence/pages/viewpage.action?spaceKey=E3SPVMAN&title=%5B%21DEV-DB%21%5D+Fuzzing+List
- Code Architecture
  - Chernyshev, Dmitrii (EXTERN: LUX) presented the code architecture
  - Common Repository is perfect to start with, since it is used by a lot of other components
  - REPO: https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_diagmaster_common/browse?at=refs%2Fheads%2Fdevelop
    - ConfigurationReader.cpp
      Some Configuration parser
    - TopologyMapper.cpp
      Parsing happens in the Constructor. Then all other methods
      Example in TopologyMapperTest.cpp
      - Write fuzz data to a file and hand it to the Constructor
    - To test locally, look into gtest_main.cpp

14.63.5 TODO

- [ ] Kutscher, Max (EXTERN: PTG) Litke, Johann (EXTERN: Bertrandt)
  Dang, Quang Anh (EXTERN: ITK) create POC for DiagNinja components
- [ ] Invite for a separate meeting with DiaGate team for diag_access_manager component

483 https://devstack.vwgroup.com/confluence/spaces/E3DIAG/pages/1892095250/DiagMaster+Component+s+Contacts

14.63.6 Links

1.1 ADA + 1.2 + 2.0 - Guideline to extract a list of security relevant fuzzing items[484]

Example of how the report looks like when Doxygen markings are in a develop branch[485]

Fuzzing Bible: CI-Fuzz Proof of Concept (see page 136)

14.64 2024.11.13 Sync with Team Adaptive regarding fuzzing status

14.64.1 Participants

Kutscher, Max (EXTERN: PTG)

Litke, Johann (EXTERN: Bertrandt)

Sobocinski, Maciej (EXTERN: EB)

Koskiranta, Eppu (EXTERN: EB)

14.64.2 FYI

Developers from Adaptive

14.64.3 Agenda

- What is the status on that?
- Is the Team using fuzzing already actively?
- Is the team planning to use fuzzing in this PI?
- Does the team need a refresh regarding the fuzzing topic?

484 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1378239468
485 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=3061303501

14.64.4 Minutes

- Adaptive is onboarded regarding fuzzing
- First fuzzing test already exists and plan to use fuzzing in this PI
  - Methods and classes for fuzzing are identified and considered in this PI
  - [E3-152764] Identify methods and classes which shall be fuzz tested - DevStack Jira[486]
  - [E3-146392] Identify methods and classes which shall be fuzz tested - DevStack Jira[487]
- Additional fuzzing targets will be created by the Adaptive team
- Team was provided with information how to run fuzzing targets locally
- They will have a look onto it and if further support is needed they will contact us

14.64.5 TODO

- [x] Kutscher, Max (EXTERN: PTG) update CI-Fuzz List of Onboarding Teams - E3SP Security - DevStack Confluence (see page 158) with this information for team Adaptive

14.64.6 Links

1.1 ADA + 1.2 + 2.0 - Guideline to extract a list of security relevant fuzzing items[488]

Example of how the report looks like when Doxygen markings are in a develop branch[489]

Fuzzing Bible: CI-Fuzz Proof of Concept (see page 136)

Remote Repair where fuzzing is established[490]

CI-Fuzz List of Onboarding Teams - E3SP Security - DevStack Confluence[491]

14.65 2024.11.21 Sync with Team TM SP (Adaptive Autosar) and TM Obd (Classic Autosar) regarding fuzzing status

14.65.1 Participants

Kutscher, Max (EXTERN: PTG)

Litke, Johann (EXTERN: Bertrandt)

---

486 https://devstack.vwgroup.com/jira/browse/E3-152764
487 https://devstack.vwgroup.com/jira/browse/E3-146392
488 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1378239468
489 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=3061303501
490 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse?at=refs%2Fheads%2Ffeature%2FE3APP007-1168-establish-cifuzz-testing
491 https://devstack.vwgroup.com/confluence/display/E3SEC/CI-Fuzz+List+of+Onboarding+Teams?preview=%2F1314972588%2F4076605353%2F2024_10_23_Onboarding_1st_sess.pptx

Bashtannik, Oleksandr (Digiteq Automotive)

Culik, Jan (Digiteq Automotive)

Dolejsova, Stanislava (Digiteq Automotive)

Oscadal, Marek (Digiteq Automotive)

14.65.2 FYI

Dang, Quang Anh (EXTERN: ITK)

Schaller, Bernd (T5-SF)

14.65.3 Agenda

- What is the status on that?
- Is the Team using fuzzing already actively?
- Is the team planning to use fuzzing in this PI?
- Does the team need a refresh regarding the fuzzing topic?

14.65.4 Minutes

- TM SP (Adaptive Autosar) and TM Obd (Classic Autosar) have considered and planned fuzzing for this PI
- Fuzzing targets have been identified and first test cases for targets have been already created
- Overall 70% of the test cases already created and last 30% will be also created
- Fuzzing is already included in develop-cl46 branch (→ current latest sources contain fuzzing test cases)
- The team is waiting for execution of fuzzing in the CI/CD pipeline

- Team will contact us if additional support is needed

14.65.5 TODO

1. Kutscher, Max (EXTERN: PTG) update CI-Fuzz List of Onboarding Teams - E3SP Security - DevStack Confluence (see page 158) with this information for team TM SP (Adaptive Autosar) and TM Obd (Classic Autosar)

14.65.6 Links

- TM SP (Adaptive Autosar) and TM Obd (Classic Autosar) Links:
  - Adaptive 1.2 Fuzz Testing[492]
  - Classic 1.2 Fuzz Testing[493]
  - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_timemaster_au416/browse/tests/fuzzing?at=refs%2Fheads%2Fdevelop-cl46
  - https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_tm_base/browse/tests/fuzzing?at=refs%2Fheads%2Fdevelop

1.1 ADA + 1.2 + 2.0 - Guideline to extract a list of security relevant fuzzing items[494]

Example of how the report looks like when Doxygen markings are in a develop branch[495]

Fuzzing Bible: CI-Fuzz Proof of Concept (see page 136)

Remote Repair where fuzzing is established[496]

CI-Fuzz List of Onboarding Teams - E3SP Security - DevStack Confluence[497]

492 https://devstack.vwgroup.com/confluence/spaces/E3APP004/pages/2844888219/Adaptive+1.2+Fuzz+Testing
493 https://devstack.vwgroup.com/confluence/spaces/E3APP004/pages/3008412770/Classic+1.2+Fuzz+Testing
494 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1378239468
495 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=3061303501
496 https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse?at=refs%2Fheads%2Ffeature%2FE3APP007-1168-establish-cifuzz-testing
497 https://devstack.vwgroup.com/confluence/display/E3SEC/CI-Fuzz+List+of+Onboarding+Teams?preview=%2F1314972588%2F4076605353%2F2024_10_23_Onboarding_1st_sess.pptx

14.66 2024.11.28 Hands-on session with Team DiagNinja regarding fuzzing

14.66.1 Participants

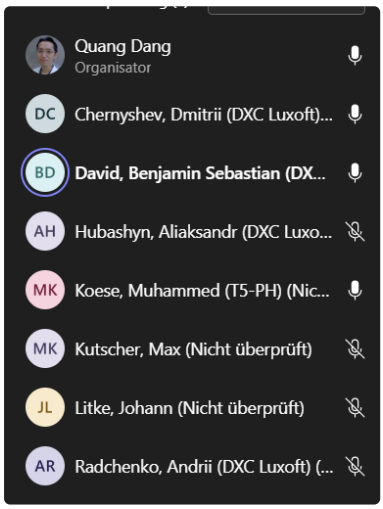
Dang, Quang Anh (EXTERN: ITK)

Schaller, Bernd (T5-SF)

Kutscher, Max (EXTERN: PTG)

Litke, Johann (EXTERN: Bertrandt)

Chernyshev, Dmitrii (EXTERN: LUX) (Contact for questions about code and unit tests)



14.66.2 FYI

Developers from DiagNinjas

14.66.3 Agenda

- A volunteer from team DiagNinja executes followings:

```
- git clone ssh://git@devstack.vwgroup.com:7999/e3comp/
  e3_comp_diagmaster_common.git
  git checkout feature/VWOS-15920-establish-fuzzing-testing
  docker pull e3-e3sp-dockerdev.docker.devstack.vwgroup.com/master/e3sdk-build-
  bp1:4.5.0

  touch ~/.conan/profiles/fuzzing_x86_64
  [env]
  CC=/usr/bin/clang-15
  CXX=/usr/bin/clang++-15
```

```
LD=/usr/bin/ld
AR=/usr/bin/ar


[settings]
# Host settings
os_build=Linux
arch_build=x86_64

# Target settings
os=Linux
os.target=local
arch=x86_64
compiler=gcc
compiler.version=9.3
compiler.libcxx=libstdc++11
build_type=Debug

./build.sh --docker --target fuzzing --profile fuzzing_x86_64 --verbose --
fuzzing --sanitizer=address:undefined --sanitizer-args=-fsanitize-
recover=address:-fsanitize-address-use-after-scope -j2
./build.sh --docker --target fuzzing --coverage --profile fuzzing_x86_64 --
buildfolder=build-cov --verbose --fuzzing -j2
```

- tmp

14.66.4 Minutes

28 Nov 2024

Which files needed to be created or updated and how the fuzzing test files looks like was explained:

- cmake
  - APfuzzing.cmake
  - CMakeLists.txt
- tests
  - fuzzing
    - fuzz_ConfigReader.cpp
    - fuzz_TopologyMapper.cpp
    - CMakeLists.txt
  - CMakeLists.txt
- CMakeLists.txt
- cifuzz.sh
- cifuzz.yaml
- e3sdk_metadata

The content of fuzzing test files (fuzzing target) was explained.

Mentioned creating fuzzing profile under ~/.conan/profiles/

- fuzzing_x86_64

How fuzzing is build was shown → commands for local build

- Execute building with ./cifuzz
- Execute fuzzing
- Execute building coverage
- Generate coverage repot

Showed a fuzzing bug in RapidJSON lib when executing fuzz_ConfigReader.cpp and mentioned the created bug ticket.

- Discussion about if bug is valid or not. → Bug can only appear if flashing container is not valid.
- Discussion what to with this bug. The team needs to decide what to do with this bug.
  - Don't accept bug → Risk management
  - Escalate and report to RapidJson team
  - Team could create their own fix

Dmitrii checked out repo and latest branch.

- Runs build
- Runs fuzzing
- Runs coverage report generation

A Second Hand on session should be planned for next PI.

- Showing again building and fuzzing execution
- Showing debugging

24 Oct 2024

- DiagNinjas responsible for following components/functionalities:
  - https://devstack.vwgroup.com/confluence/display/E3DIAG/DiagMaster+Services%27+Contacts?src=contextnavpagetreemode
- DiaGate
  - DiagMaster Component's Contacts[498]
- Presentation of fuzzing
- Q&A:

498 https://devstack.vwgroup.com/confluence/spaces/E3DIAG/pages/1892095250/DiagMaster+Component+s+Contacts

    - Is there overview for fuzzing result => https://devstack.vwgroup.com/confluence/pages/viewpage.action?spaceKey=E3SPVMAN&title=%5B%21DEV-DB%21%5D+Fuzzing+List
- Code Architecture
    - Chernyshev, Dmitrii (EXTERN: LUX) presented the code architecture
    - Common Repository is perfect to start with, since it is used by a lot of other components
    - REPO: https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_diagmaster_common/browse?at=refs%2Fheads%2Fdevelop
        - ConfigurationReader.cpp
          Some Configuration parser
        - TopologyMapper.cpp
          Parsing happens in the Constructor. Then all other methods
          Example in TopologyMapperTest.cpp
            - Write fuzz data to a file and hand it to the Constructor
        - To test locally, look into gtest_main.cpp

14.66.5 TODO

- [ ] Kutscher, Max (EXTERN: PTG) Litke, Johann (EXTERN: Bertrandt)
  Dang, Quang Anh (EXTERN: ITK) create POC for DiagNinja components
- [ ] Invite for a separate meeting with DiaGate team for diag_access_manager component

14.66.6 Links

1.1 ADA + 1.2 + 2.0 - Guideline to extract a list of security relevant fuzzing items[499]

Example of how the report looks like when Doxygen markings are in a develop branch[500]

Fuzzing Bible: CI-Fuzz Proof of Concept (see page 136)

14.67 2024.12.02 Sync with Martin OTA-Master SRA extraction

14.67.1 Participants

Hartung, Martin (EXTERN: ITK)

Muckenschnabl, David Jose (EXTERN: ITK)

14.67.2 Agenda

List Data flows related to code hardening

499 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1378239468
500 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=3061303501

14.67.3 Minutes

S.1 Transmission of D.43 SFD Token request from client [K.4 DiagMaster -> K.3 otam-tm],

S.3 Transmission of D.2 Anfragestruktur (SFD), D.8 Target Diagnostic Address [K.4 DiagMaster -> K.3 otam-tm],

S.7 Transmission of D.32 Authentication Token [K.7 Core Services / CS-Proxy -> K.3 otam-tm],

S.11 Transmission of D.4 SFD Token [K.7 Core Services / CS-Proxy -> K.3 otam-tm],

S.13 Transmission of D.5 SFL login job [K.8 ORU -> K.3 otam-tm],

S.19 Transmission of D.46 Diagnostic result from Immobilizer ECUs [K.4 DiagMaster -> K.3 otam-tm],

S.23 Transmission of D.32 Authentication Token [K.7 Core Services / CS-Proxy -> K.3 otam-tm],

S.27 Transmission of D.7 SFL Login Code [K.7 Core Services / CS-Proxy -> K.3 otam-tm],

S.31 Transmission of D.9 SFL State [K.4 DiagMaster -> K.3 otam-tm],

S.37 Transmission of D.12 Session Challenge (SFL) [K.7 Core Services / CS-Proxy -> K.3 otam-tm],

S.38 Transmission of D.40 Wakeup message [K.8 ORU -> K.10 otam_vsm],

S.40 Transmission of D.48 Request preconditions check [K.8 ORU -> K.10 otam_vsm],

S.41 Transmission of D.41 Signal input values [K.9 libSCI + Comserver -> K.10 otam_vsm],

S.43 Transmission of D.36 Request charging/energy approval [K.8 ORU -> K.10 otam_vsm],

S.45 Transmission of D.37 Charging/energy approval [K.9 libSCI + Comserver -> K.10 otam_vsm],

S.47 Transmission of D.28 Request for ORU state [K.8 ORU -> K.10 otam_vsm],

S.49 Transmission of D.48 Request preconditions check [K.18 Remote Repair -> K.10 otam_vsm],

S.50 Transmission of D.41 Signal input values [K.9 libSCI + Comserver -> K.10 otam_vsm]"

14.67.4 TODO

- [x] Muckenschnabl, David Jose (EXTERN: ITK) Set up a meeting with OTA-Master for identifying relevant code and writing Requirements

14.68 2024.12.05 Sync with OTA-Master Code to Data flow mapping and Requirements

14.68.1 Participants

Hartung, Martin (EXTERN: ITK)

Muckenschnabl, David Jose (EXTERN: ITK)

Ivanov, Sergei (EXTERN: LUX)

Maslennikov, Viktor (EXTERN: LUX)

Unknown User (wit1rpi)

14.68.2 Agenda

Link Data flows to code

Write new requirements

14.68.3 Minutes

Looked into SRS for background

Looked into SRA for background on the Data flows list

14.68.4 Data Flows

14.68.5 otam_tm

14.68.5.1 SFD

14.68.5.1.1 S.1 Transmission of D.43 SFD Token request from client [K.4 DiagMaster -> K.3 otam-tm]

```
The main interface of TM SFD
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_tm/browse/src/
TokenManagementSkeletonImpl.hpp#122-123,134-137,147-148

Other participants (for mocking), see details in other points below:
e3_comp_otam_common/src/helpers/CSProxyCommunicationHelper.cpp
e3_comp_otam_common/src/common/OffboardAuthenticator.hpp
e3_comp_otam_common/src/helpers/ImmobilizerHelper.cpp
e3_comp_otam_common/src/helpers/DRCCommHelper.cpp
```

14.68.5.1.2 S.11 Transmission of D.4 SFD Token [K.7 Core Services / CS-Proxy -> K.3 otam-tm],

```
SFD job uses csproxy (via job) and saves to results available by pollJob
and notification:
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_tm/browse/src/sfd/JobTMSFD.cpp#425,438,470,507,510-511,517
->
csproxy job
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
```

```
e3_comp_otam_common/browse/src/common/JobCSProxyHandler.hpp
->
CSProxy performs http reqest and returns token (protobuf):
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/helpers/CSProxyCommunicationHelper.hpp#108

Mock:
e3_comp_otam_tm/src/TokenManagementSkeletonImpl.hpp - input

e3_comp_otam_common/src/common/OffboardAuthenticator.hpp
e3_comp_otam_common/src/helpers/ImmobilizerHelper.cpp
e3_comp_otam_common/src/helpers/DRCCommHelper.cpp
```

14.68.5.1.3 S.3 Transmission of D.2 Anfragestruktur (SFD), D.8 Target Diagnostic Address [K.4 DiagMaster -> K.3 otam-tm],

```
SFD job uses DRC (via drc job)
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_tm/browse/src/sfd/JobTMSFD.cpp#64,273-274,318
->
DRC job
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/common/JobDRC.hpp
->
DRC start + read routine used, returns json
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/helpers/DRCCommHelper.hpp#81,91
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/helpers/DRCCommHelper.cpp#178,191,350,430

Mock:
e3_comp_otam_tm/src/TokenManagementSkeletonImpl.hpp

e3_comp_otam_common/src/common/OffboardAuthenticator.hpp
e3_comp_otam_common/src/helpers/ImmobilizerHelper.cpp
e3_comp_otam_common/src/helpers/CSProxyCommunicationHelper.cpp
```

14.68.5.2 SFL

14.68.5.2.1 S.13 Transmission of D.5 SFL login job [K.8 ORU -> K.3 otam-tm],

```
The main interface of TM SFL
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_tm/browse/src/TokenManagementSkeletonImpl.hpp#77,89,102
```

```
Other participants (for mocking):
e3_comp_otam_common/src/helpers/CSProxyCommunicationHelper.cpp
e3_comp_otam_common/src/common/OffboardAuthenticator.hpp
e3_comp_otam_common/src/helpers/ImmobilizerHelper.cpp
e3_comp_otam_common/src/helpers/DRCCommHelper.cpp
e3_comp_otam_common/src/helpers/DDSv2CommHelper.hpp
```

14.68.5.2.2 S.27 Transmission of D.7 SFL Login Code [K.7 Core Services / CS-Proxy -> K.3 otam-tm],

```
SFL job uses csproxy

SFL v5: requests token from CSProxy (via job), saves tokens to write them
via DiagData on the further step.
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_tm/browse/src/sfl/JobTMSFLv5.cpp#895,921,951,954,987

SFL v6: requests token from CSProxy (via job), saves tokens to write them
via DRC on the further step.
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_tm/browse/src/sfl/JobTMSFLv6.cpp#115,904,972,982,1007

->
csproxy job
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/common/JobCSProxyHandler.hpp
->
CSProxy performs http reqest and returns token (protobuf):
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/helpers/CSProxyCommunicationHelper.hpp#108
```

14.68.5.2.3 S.31 Transmission of D.9 SFL State [K.4 DiagMaster -> K.3 otam-tm],

```
SFL job writes requested token and checks the results

SFL v5: puts the token in json and request DiagData setKeys (via
JobDD2Write)
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_tm/browse/src/sfl/JobTMSFLv5.cpp#1153-1154,1192,1246
->
JobDD2Write calls DiagData setKeys
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/common/JobDD2Write.cpp#216
->
Proxy calls requestSetDiagData and readRequestedResults
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/helpers/
```

```
DDSv2CommHelper.cpp#273,365,401,432


SFL v6: puts the token in json and request DiagData setKeys (via JobDRC)
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_tm/browse/src/sfl/JobTMSFLv6.cpp#712,1085,1110
->
DRC job
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/common/JobDRC.hpp
->
DRC start + read routine used, returns json
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/helpers/DRCCommHelper.hpp#81,91
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/helpers/DRCCommHelper.cpp#178,191,350,430
```

14.68.5.2.4 S.37 Transmission of D.12 Session Challenge (SFL) [K.7 Core Services / CS-Proxy -> K.3 otam-tm],

```
SFL v6 job requests challenge from csproxy (via csproxy job):
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_tm/browse/src/sfl/JobTMSFLv6.cpp#548,570,581
->
csproxy job
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/common/JobCSProxyHandler.hpp
->
CSProxy performs http reqest and returns token (protobuf):
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/helpers/CSProxyCommunicationHelper.hpp#108
```

14.68.5.2.5 S.19 Transmission of D.46 Diagnostic result from Immobilizer ECUs [K.4 DiagMaster -> K.3 otam-tm],

```
SFL v5 requests items from DiagData (via JobDD2Read):
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_tm/browse/src/sfl/JobTMSFLv5.cpp#585,648
->
JobDD2Read
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/common/JobDD2Read.cpp
->
Proxy calls readRequestedResults
```

```
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/helpers/DDSv2CommHelper.cpp#303,446

SFLv6:
1) requests items from DiagData:
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_tm/browse/src/sfl/JobTMSFLv6.cpp#605,638
->
JobDD2Read
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/common/JobDD2Read.cpp
->
Proxy calls readRequestedResults
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/helpers/DDSv2CommHelper.cpp#303,446

2) requests items from DiagRoutine (DRC) - 3 requests:
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_tm/browse/src/sfl/JobTMSFLv6.cpp#706,714,747,790,830
->
DRC job
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/common/JobDRC.hpp
->
DRC start + read routine used, returns json
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/helpers/DRCCommHelper.hpp#81,91
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/helpers/DRCCommHelper.cpp#178,191,350,430
```

14.68.5.2.6 Additionally, SFL v5 and v6 request access from DiagAccessManger as one of steps

https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_comp_otam_common/browse/src/helpers/DAMProxyHelper.cpp?at=master-cl46#220,279,318

14.68.5.2.7 Additionally, SFL consumes SCI signal: "KST_KL_15":

https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_otam_tm/browse/src/sfl/renew/ClampControlConsumer.cpp#43-44,46

14.68.5.3 Common

14.68.5.3.1 S.7 Transmission of D.32 Authentication Token [K.7 Core Services / CS-Proxy -> K.3 otam-tm],

```
SFD job requests authentication and urls. Uses it for csproxy requests
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_tm/browse/src/sfd/JobTMSFD.cpp#65,217,230,436
->
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/helpers/
OTACommunicationCommHelper.cpp#100,129,150,192,214
```

14.68.5.3.2 S.23 Transmission of D.32 Authentication Token [K.7 Core Services / CS-Proxy -> K.3 otam-tm],

```
SFL v5:
requests auth and URLs for "token"
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_tm/browse/src/sfl/JobTMSFLv5.cpp#792,798,892

SFL v6:
requests auth and URLs for "challenge" and "token" requests
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_tm/browse/src/sfl/
JobTMSFLv6.cpp#118-119,481,488,514,521,544,546,898,900
```

14.68.5.3.3 Additionally, Immobilizer is used to request VIN in both SFD and SFL(v5 and v6) as one of steps

https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_comp_otam_common/browse/src/helpers/ImmobilizerHelper.cpp?at=refs%2Fheads%2Fmaster-cl46#151 -> Base PIF class https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_comp_otam_common/browse/src/pifcommon/PIFBaseConsumer.tpp?at=refs%2Fheads%2Fmaster-cl46

14.68.6 otam_vsm

14.68.6.1 Network Management

S.38 Transmission of D.40 Wakeup message [K.8 ORU -> K.10 otam_vsm],

```
Network wakeup API
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_vsm/browse/src/VehicleStateManager.hpp?
at=refs%2Fheads%2Fmaster-cl46#229,238,247,257

SCI signal consumed:
"SI_QRS_Mode"
```

14.68.6.2 Precondition Checker

S.40 Transmission of D.48 Request preconditions check [K.8 ORU -> K.10 otam_vsm],

S.49 Transmission of D.48 Request preconditions check [K.18 Remote Repair -> K.10 otam_vsm],

```
Precondition job API
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_vsm/browse/src/VehicleStateManager.hpp?
at=refs%2Fheads%2Fmaster-cl46#147,161,170,180

SCI signals (see S.41, S.50), consumed depend on the request:
"AAG_Anhaenger_erkannt"
"AB_Belegung_VF"
"BCM1_Parklicht_li_Anf"
"BCM1_Parklicht_re_Anf"
"BCM1_Standlicht_Anf"
"BM_DWA_Alarm"
"BM_Panik"
"BM_Taxi_Notalarm"
"BM_Warnblinken"
"BT_FH_Oeffnung"
"BT_Tuer_geoeffnet"
"BT_verriegelt"
"FT_FH_Oeffnung"
"FT_Tuer_geoeffnet"
"FT_verriegelt"
"HBFS_FH_Oeffnung"
"HBFS_Tuer_geoeffnet"
"HBFS_verriegelt"
"HFS_FH_Oeffnung"
"HFS_Tuer_geoeffnet"
"HFS_verriegelt"
"KST_KL_15"
"MD1_Position"
"PTAcvSrv_StPwrMd"
"SAD2_Deckel_Position"
"VDSO_ALgtOfs"
"VDSO_ALgtOfsCmp"
"VDSO_StVx3d"
```

```
"VDSO_Vx3d"
"VMM_HldMng_StPrkBrkAcv"
"VMM_HldMng_StPrkBrkCpb"
"VMM_HldMng_StPrkPosnAcv"
"VS_VD_zu_ver"
"ZV_Frontdeckel_offen"
"ZV_Heck_offen"
```

14.68.6.3 Common

S.41 Transmission of D.41 Signal input values [K.9 libSCI + Comserver -> K.10 otam_vsm],

S.50 Transmission of D.41 Signal input values [K.9 libSCI + Comserver -> K.10 otam_vsm]"

```
Common SCI signals impl:
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/common/ConfigurationSCIWrapper.cpp?
at=refs%2Fheads%2Fmaster-cl46#101
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/common/signals?at=master-cl46
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/common/signals/SignalProxy.tpp?at=master-
cl46
```

14.68.6.4 Charging/Energy Management

S.43 Transmission of D.36 Request charging/energy approval [K.8 ORU -> K.10 otam_vsm],

```
1) Charging Management API
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_vsm/browse/src/VehicleStateManager.hpp?
at=refs%2Fheads%2Fmaster-cl46#115,125,137

2) Energy Management API
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_vsm/browse/src/VehicleStateManager.hpp?
at=refs%2Fheads%2Fmaster-cl46#193,204,220
```

S.45 Transmission of D.37 Charging/energy approval [K.9 libSCI + Comserver -> K.10 otam_vsm],

```
1) Charging Management API

ChargingManagementCore PIF is used to retrieve data and recieve events
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_vsm/browse/src/charging_manager/
```

```
ChargingManagementCoreConsumer.cpp?at=master-cl46#96,180
->
Base PIF class
https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/
e3_comp_otam_common/browse/src/pifcommon/PIFBaseConsumer.tpp?
at=refs%2Fheads%2Fmaster-cl46

used SCI signals:
"HV_Bordnetz_aktiv"
"HVK_Vorladung_aktiv"

2) Energy approval
used SCI signals:
"NVEM_Freigabe_OTA"
```

14.68.6.5 ORU State

S.47 Transmission of D.28 Request for ORU state [K.8 ORU -> K.10 otam_vsm],

```
API
https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/
e3_comp_otam_vsm/browse/src/VehicleStateManager.hpp?at=master-cl46#266
```

14.68.6.6 **TODOs**

- [x] Ivanov, Sergei (EXTERN: LUX) Add information about implementation of interfaces mapped to the corresponding Dataflows

14.68.6.7 **Link**

2024.10.15 SW Requirement interface-hardening / Fuzz Testing Remote Repair (see page 325)

OtaMaster Requirements https://cb.vwgroup.com/cb/tracker/88845988?view_id=-2

OtaMaster OTA Master - Software Architectural Design and Interfaces[501] (confluence description)

14.69 2024.12.10 Sync with Code Spiders

14.69.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Schaller, Bernd (T5-SF)

501 https://devstack.vwgroup.com/confluence/spaces/E3OTA/pages/2065513891/OTA+Master+-+Software+Architectural+Design+and+Interfaces

Hruda, Petr (PECZ-OS)

Bruno, Vittorio (T5-PH)

Kutscher, Max (EXTERN: PTG)

Litke, Johann (EXTERN: Bertrandt)

Unknown User (ez1gcj4)

14.69.2 Agenda

Discuss if Code Spiders fall under DiagMaster Risk → They do

Discuss Fuzzing path

Discuss Activities to be planned for next PI25.1

14.69.3 Minutes

Discuss if Code Spiders fall under DiagMaster Risk → They do

Discussed fuzzing and what our process is.

Planned next steps for PI25.1

14.69.4 Link

2024.10.15 SW Requirement interface-hardening / Fuzz Testing Remote Repair (see page 325)

OtaMaster Requirements https://cb.vwgroup.com/cb/tracker/88845988?view_id=-2

14.70 2025.02.17 Sync/onboarding with Team CodeSpiders regarding fuzzing

14.70.1 Participants

Shared Service Team Security:

Kutscher, Max (EXTERN: PTG)

Litke, Johann (EXTERN: Bertrandt)

CodeSpiders team:

Hruda, Petr (PECZ-OS)

Bruno, Vittorio (T5-PH)

Herman, Adam (PECZ-OS)

Glowacki, Aleksander (PECZ-OS)

14.70.2 FYI

Schaller, Bernd (T5-SF)

14.70.3 Agenda

- Presentation/onboarding to Fuzzing
- Answer the question: What is to be fuzzed for CodeSpiders?
  - SRA Fuzzing recommendation?
  - Repos/functions to fuzz for:
    - https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_diagevent/browse/src/engine/EventCache/Data/Merger
    - https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_diagevent/browse/src/engine/EventCache/Persistency/FileOperations
      - POC for functions: FileOperations::read(std::string& output)
      - GzipCompressor::decompress(std::unique_ptr<std::istream> const& stream, std::string& output)
      - GzipCompressor::compress(std::unique_ptr<std::ostream> const& stream, std::string const& text)
    - PoC contact/devoper: Hruda, Petr (PECZ-OS)

14.70.4 Minutes

14.70.5 TODO

- [ ] Kutscher, Max (EXTERN: PTG) Litke, Johann (EXTERN: Bertrandt) create PoC for CodeSpiders components
- [ ] Kutscher, Max (EXTERN: PTG) Litke, Johann (EXTERN: Bertrandt) Invite for a hands on session with team/tester from CodeSpider

14.70.6 Links

1.1 ADA + 1.2 + 2.0 - Guideline to extract a list of security relevant fuzzing items[502]

Example of how the report looks like when Doxygen markings are in a develop branch[503]

Fuzzing Bible: CI-Fuzz Proof of Concept (see page 136)

Onboarding Slides: Fuzzing Presentation to Management Slides (see page 434)

502 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1378239468
503 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=3061303501

15 CI-Sparc Fuzzing Guideline to generate Fuzzing test with LLM

Version	Date	Updated by	Change log
1.0	30 Sep 2025	Schaller, Bernd (T5-SF)	First version

Protocol:

30 Sep 2025 Review

PDF:

? Unknown Attachment

PPT:

? Unknown Attachment

16 CI-Sparc Fuzzing Presentation to Management Slides

Version	Date	Updated by	Change log
1.0	30 Sep 2025	Schaller, Bernd (T5-SF)	First version

Protocol:

30 Sep 2025 Review

Table of Contents

16.1 1. Overview

Summary	• Coming down from a repository
More details	• XC.4 - Security Management[504] • 3 Guidelines | Security Management[505] • 8 Operative Deliverables | Security Management[506] • Overview of fuzz items in all component with doxygen code marking implemented[507]

16.2 2. Term Definitions

504 https://devstack.vwgroup.com/confluence/display/VWOSS/XC.4+-+Security+Management
505 https://devstack.vwgroup.com/confluence/display/VWOSS/3+Guidelines+%7C+Security+Management
506 https://devstack.vwgroup.com/confluence/display/VWOSS/8+Operative+Deliverables+%7C+Security+Management
507 https://devstack.vwgroup.com/confluence/display/E3SPVMAN/%5B%21DEV%21%5D+Fuzzing+List

16.3 3. Fuzzing Relevant Criteria

Overall, as a best practice testing method, fuzzing should be recommended to all projects, components, classes, interfaces etc.

16.3.1 3.1. Do Fuzz

We divide fuzzing relevancy in following categories:

- **Security critical functions.** This is a special category which should be fuzzed even if it doesn't handle any kind of external input
  - Functions with elevated privileges
  - Functions with cryptographic algorithms
  - Functions executing command lines of any sorts
- Functions handle any kind of external input e.g. user, file, network input
  - Input parser / processor
  - File parser / processer
  - Network protocol
  - API endpoints
  - Custom protocol
- Complex functions. This is where it becomes tricky, since some of this functions don't handle external input directly. However, in some cases, the external input can still be propagated to some of these functions in some forms. The best practice would be to do fuzzing still. However, the fuzzer can be constraint to the semantics of internal calls so fuzzing reproduces realistic scenarios.
  - Function with a lot of pointer arithmetics
  - Functions with a lot of if-else conditions / branching

16.3.2 3.2. Don't Fuzz

There are a lot of functions you don't need to waste your time fuzzing:

- Simple functions with little to no logic
  - Getter / Setter
  - Functions forwarding data
- Functions not interacting with external input
  - ⚠️ Note that security critical functions should be fuzzed even if they don't interact with external input
  - ⚠️ Note that you also have to be sure that the external input **can not propagate** to those functions **in any form**

16.4 4. Guidelines to generate a list of fuzzing tests

16.4.1 4.1. Create your own list of fuzzing items

16.5 5. Documentation of Fuzzing Relevant tests

16.5.1 5.1. Motivation why we use LLM

PDF:

? Unknown Attachment

PPT:

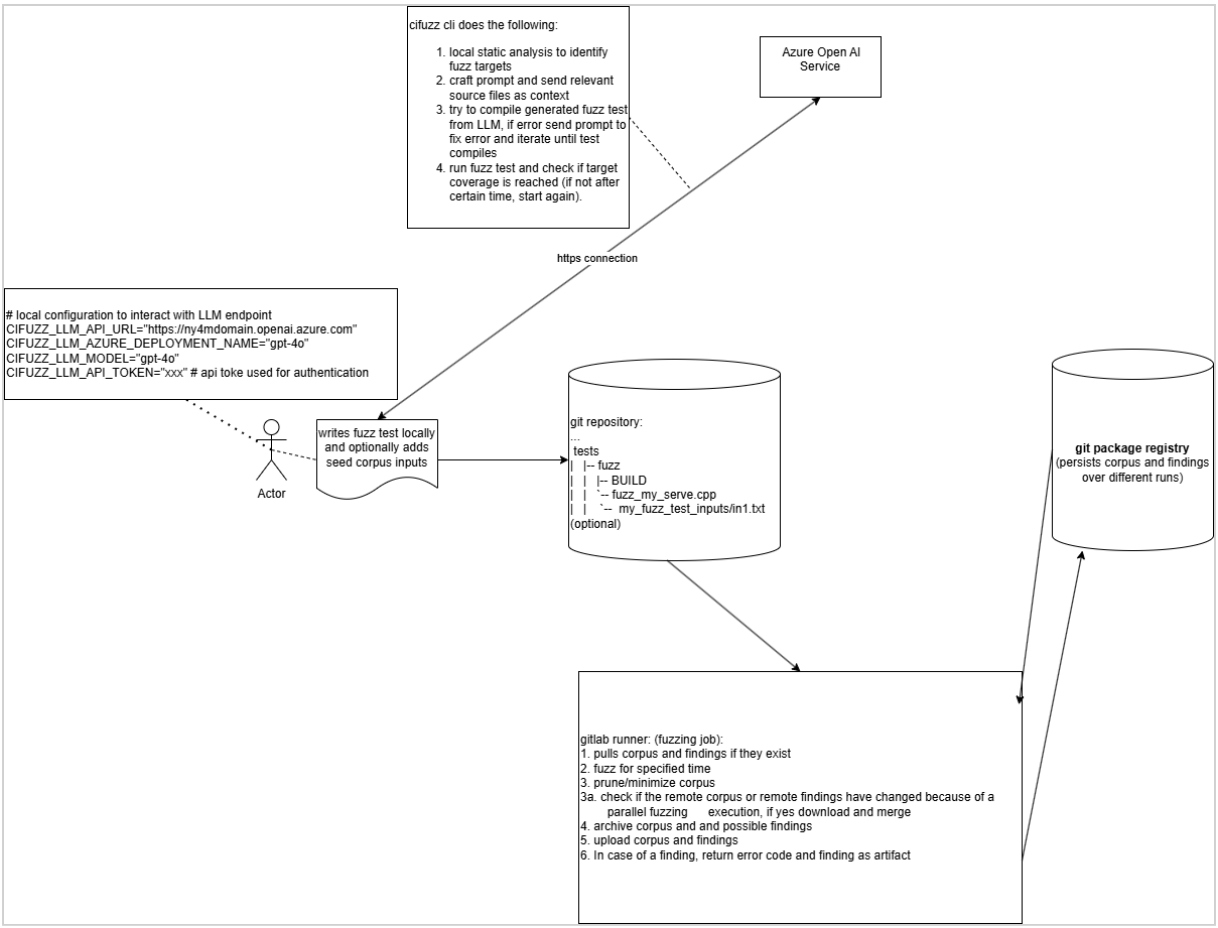
? Unknown Attachment

17 CISpark/CIFuzz generate

17.1 Official Documentation

Official documentation: LLM Setup | Code Intelligence Docs[508]

17.2 Architecture



17.3 Config with Azure OpenAI

```
export CIFUZZ_LLM_API_URL="https://ny4mdomain.openai.azure.com"
export CIFUZZ_LLM_AZURE_DEPLOYMENT_NAME="gpt-4o"
export CIFUZZ_LLM_MAX_TOKENS 40000
```

508 https://docs.code-intelligence.com/ai-test-agent/llm-setup

```
export CIFUZZ_LLM_API_TOKEN="XXXX" # replace with real token

export HTTPS_PROXY="http://127.0.0.1:9000"
export HTTP_PROXY="http://127.0.0.1:9000"
```

In WSL:

start vpnkit

```
wsl -d wsl-vpnkit --cd /app wsl-vpnkit
```

and set proxy to windows machine

```
export HTTPS_PROXY="http://192.168.127.254:9000"
export HTTP_PROXY="http://192.168.127.254:9000"
```

The firewall config in WSL can be checked with `sudo ufw status` and should contain the following.

```
192.168.127.254 9000       ALLOW OUT   Anywhere on eth0           # Allow outgoing
traffic to proxy for standard eth0
192.168.127.254 9000       ALLOW OUT   Anywhere on wsltap         # Allow outgoing
traffic to proxy for wsltap (VPN)
```

Ports of the firewall can be opened with:

```
sudo ufw allow out $PORT_NUMBER
```

There is also a script available which does the firewall settings (see Checkpoint Mobile ::[509]):

```
source /etc/profile.d/setup-proxy.sh  proxy_on --from-file=~/proxy.json
```

509 https://pages.git.hub.vwgroup.com/CICD-Tools/swf-selfservice-docs/wsl/customer-guide/networking/checkpoint-mobile-vpn/index.html

17.3.1 CiFuzz generate Validate

With `cifuzz candidates` you can rank/list possible fuzz targets:

```
cifuzz candidates

(base)
cifuzz version 3.68.0
 ✓  Generating compile commands... Done.


 ✓  Parsing source code... Done.


Found 85 potential fuzz test candidates.


 ✓  Analyzing control-flow graph... Done.



  | Function                      | Score | LoC  | ULoC | CC   | DE  | DFC | Location
1 | yaml_parser_load              | 1.00  | 2468 |  -   | 2093 | 781 | 91  | src/loader.
c:86
2 | yaml_parser_parse             | 0.94  | 2199 |  -   | 1965 | 730 | 86  | src/parser.
c:181
3 | yaml_parser_scan              | 0.72  | 1521 |  -   | 1632 | 546 | 63  | src/
scanner.c:742
4 | yaml_parser_fetch_more_tokens | 0.72  | 1506 |  -   | 1624 | 544 | 62  | src/
scanner.c:800
5 | yaml_emitter_dump             | 0.63  | 1570 |  -   | 1309 | 539 | 15  | src/dumper.
c:111

 LoC - Total lines of code reachable from this function
ULoC - Total uncovered lines of code reachable from this function
  CC - Total cyclomatic complexity reachable from this function
  DE - Total number of dangerous expressions (pointer dereferences) reachable from this
function
 DFC - Total number of dangerous function calls (e.g. memcpy, strcpy, etc.) reachable
from this function
Logs can be found here:
        /home/andreas/repos/libyaml/.cifuzz/logs/candidates.log
```

automatically try to create working fuzz testcase

```
cifuzz create yaml_parser_load --validate -o fuzz/fuzz_yaml_parser_load.cpp  --fuzz-tes
t-name fuzz_yaml_parser_load -v
```

need Cmake in folder fuzz `fuzz/CMakeLists.txt ` (generator function needs new CMake version, rewrite linking for older versions)

```
add_fuzz_test(fuzz_yaml_parser_load fuzz_yaml_parser_load.cpp)
target_link_libraries(fuzz_yaml_parser_load PRIVATE "$<LINK_LIBRARY:WHOLE_ARCHIVE,$
{CMAKE_PROJECT_NAME}>")
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.16)


project (yaml C CXX)


find_package(cifuzz NO_SYSTEM_ENVIRONMENT_PATH)


enable_fuzz_testing()
```

SDV use case:

from SDV evaluation

call cifuzz create with build folder (`-b`), e.g.:

```
 cifuzz create doip_send_diag_resp -b west_build/s32z2xxdc2_r52/s32z270/rtu0/sdv_demo/
```

Add to bazel file in test folder:

```
cc_fuzz_test(
    name = "doip_fuzz_tests",
    srcs = ["fuzz/fuzz_doip_step_non_blocking.cpp",
        "unit/fakes/doip_fakes.h",
        "unit/fakes/doip_fakes.c","//service/doip/code"],
    copts = [
        "-Iservice/doip/code/private",
        "-Iservice/doip/code/public",
        "-Iservice/doip/tests/unit/fakes",
    ],

    corpus = select({
        "@cifuzz//:collect_coverage": glob([".doip_fuzz_tests_cifuzz_corpus/**"],
allow_empty = True),
        "//conditions:default": [],
    }),

    deps = [
        "//lib/lib_fff",
        "//lib/lib_common",
        "//lib/lib_errno",
        # not the network library, but the fakes for it
        "//lib/lib_network:lib_network_export_fakes",
        "@cifuzz"],
)
```

And copy output of cifuzz create:

```
#include <cassert>
#include <cifuzz/cifuzz.h>
#include <fuzzer/FuzzedDataProvider.h>
#include <cstdint>
#include <cstdbool>
#include <cstring>

extern "C" {
    #include "doip.h"
    #include "lib_errno.h"
    #include "doip_fakes.h"
}

DECLARE_DOIP_CONNECTIONS(2, 2048, 0x4321, false);



FUZZ_TEST_SETUP() {
  // One-time initialization tasks can be done here if necessary.
}

FUZZ_TEST(const uint8_t *data, size_t size) {
  // Initialize the FuzzedDataProvider with the input data.
  FuzzedDataProvider fdp(data, size);
  int32_t err = doip_init(&doip_config, NULL, NULL);
```

```
  // Perform the doip_uds_step logic
  while (fdp.remaining_bytes() > 0) {
    auto payload = fdp.ConsumeBytes<uint8_t>(fdp.ConsumeIntegralInRange<size_t>(0,fdp.r
emaining_bytes()));
    auto address = fdp.ConsumeIntegral<uint16_t>();



      /* scenario 1 - address isn't on any socket */
    doip_config.instance->sockets[0].state =
static_cast<doip_state_e>(fdp.ConsumeIntegralInRange<uint8_t>(doip_state_none,
doip_state_finalize));
    doip_config.instance->sockets[1].state =
static_cast<doip_state_e>(fdp.ConsumeIntegralInRange<uint8_t>(doip_state_none,
doip_state_finalize)); //doip_state_active;

    doip_send_diag_resp(
        &doip_config,
        address,
        payload.data(),
        payload.size());
  }
}
```

17.4 Run with local LLMs

17.4.1 Compile Llama.cpp

Install local dependencies (assume you have gcc/g++or clang/clang++ etc already installed)

```
sudo apt install libcurl4-openssl-dev cmake
```

Compile:

```
git clone https://github.com/ggml-org/llama.cpp.git
cd llama.cpp
cmake -B build
cmake --build build --config Release
```

17.4.2 Run the LLM Server:

For example with qwen-model:

```
./build/bin/llama-server -hf Qwen/Qwen3-8B-GGUF
```

17.4.3 Configure Cifuzz spark environmental variables:

llama.cpp

```
export CIFUZZ_LLM_API_URL="http://127.0.0.1:8080"
export CIFUZZ_LLM_AZURE_DEPLOYMENT_NAME="gpt-4o"
export CIFUZZ_LLM_MODEL="gpt-4o"
export CIFUZZ_LLM_MAX_TOKENS=40000
export CIFUZZ_LLM_API_TOKEN="aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" # dummy api key
```

Ollama:

```
export CIFUZZ_LLM_API_URL="http://127.0.0.1:11434/v1"
export CIFUZZ_LLM_AZURE_DEPLOYMENT_NAME="gpt-4o"
export CIFUZZ_LLM_MODEL="qwen2.5-coder:14b"
export CIFUZZ_LLM_MAX_TOKENS=8192
export CIFUZZ_LLM_API_TOKEN="aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" # dummy api
```

These configs can be stored in .env files and loaded with, e.g., `source .env_ollama.`

Koboldcpp config (kobold_cpp.kcpps):

```
{"model": [], "model_param": null, "port": 5001, "port_param": 5001, "host": "",
"launch": true, "config": null, "threads": 7, "usecublas": null, "usevulkan": null,
"useclblast": null, "usecpu": true, "contextsize": 4096, "gpulayers": -1,
"tensor_split": null, "version": false, "analyze": "", "ropeconfig": [0.0, 10000.0],
"blasbatchsize": 512, "blasthreads": null, "lora": null, "noshift": false,
"nofastforward": false, "useswa": false, "usemmap": false, "usemlock": false, "noavx2":
false, "failsafe": false, "debugmode": 0, "onready": "", "benchmark": null, "prompt":
"", "cli": false, "promptlimit": 100, "multiuser": 1, "multiplayer": false, "websearch":
 false, "remotetunnel": false, "highpriority": false, "foreground": false,
"preloadstory": null, "savedatafile": null, "quiet": false, "ssl": null, "nocertify":
false, "mmproj": null, "mmprojcpu": false, "visionmaxres": 1024, "draftmodel": null,
"draftamount": 8, "draftgpulayers": 999, "draftgpusplit": null, "password": null,
"ignoremissing": false, "chatcompletionsadapter": "AutoGuess", "flashattention": false,
"quantkv": 0, "forceversion": 0, "smartcontext": false, "unpack": "", "exportconfig":
"", "exporttemplate": "", "nomodel": false, "moeexperts": -1, "defaultgenamt": 4096,
```

```
"nobostoken": false, "enableguidance": false, "maxrequestsize": 32, "overridekv": null,
"overridetensors": null, "showgui": false, "skiplauncher": false, "hordemodelname": "",
"hordeworkername": "", "hordekey": "", "hordemaxctx": 0, "hordegenlen": 0, "sdmodel":
"", "sdthreads": 7, "sdclamped": 0, "sdt5xxl": "", "sdclipl": "", "sdclipg": "",
"sdvae": "", "sdvaeauto": false, "sdquant": false, "sdlora": "", "sdloramult": 1.0,
"sdnotile": false, "whispermodel": "", "ttsmodel": "", "ttswavtokenizer": "", "ttsgpu":
false, "ttsmaxlen": 4096, "ttsthreads": 0, "embeddingsmodel": "", "admin": false,
"adminpassword": "", "admindir": "", "hordeconfig": null, "sdconfig": null, "noblas":
false, "nommap": false}
```

```
export CIFUZZ_LLM_API_URL="http://127.0.0.1:5001/v1"
export CIFUZZ_LLM_AZURE_DEPLOYMENT_NAME="gpt-4o"
export CIFUZZ_LLM_MODEL="gpt-4o"
export CIFUZZ_LLM_MAX_TOKENS=8192
export CIFUZZ_LLM_API_TOKEN="aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" # dummy api
```

Now cispark connects to local LLM.

```
cifuzz spark --target-coverage 70
```

17.5 Troubleshooting

17.5.1 Problem CMAKE_C_COMPILE_OBJECT

```
✗   Validating build system configuration... Error!

Working directory: "/home/g9lmeoq/repos/rapidjson"
Command: "cmake" "-DCIFUZZ_ENGINE=libfuzzer-clang" "-DCIFUZZ_TESTING:BOOL=ON" "-
DCIFUZZ_SANITIZERS=address;undefined" "-DCMAKE_BUILD_TYPE=RelWithDebInfo" "-
DCMAKE_BUILD_RPATH_USE_ORIGIN:BOOL=ON" "-B" "/home/g9lmeoq/repos/rapidjson/.cifuzz/
build/libfuzzer-clang/address+undefined" "/home/g9lmeoq/repos/rapidjson"
-- Configuring done
CMake Error: Error required internal CMake variable not set, cmake may not be built
correctly.
Missing variable is:
CMAKE_C_COMPILE_OBJECT
-- Generating done
```

```
CMake Generate step failed.  Build files cannot be regenerated correctly.
```

Solution:

add "C" to Languages in Cmake

```
PROJECT(RapidJSON VERSION "${LIB_VERSION_STRING}" LANGUAGES CXX C)
```

17.5.2 Out of Memory for 32B models in WSL

To run a 32B model in WSL you need at least 22GB of RAM. To configure this, add the following to your `.wslconfig:`

```
memory=22GB
```

After restarting WSL you can check with:

```
cat /proc/meminfo
MemTotal:       22548820 kB
...
```

18 Fuzzing Pipeline incl. JIRA bug reporting with Jenkins

status	version	date modified	owner	process area
	15 (see page 361)	2024-05-17 15:09	Unknown User (egxwokt)	SUP.9a - Defect Management[510]

further meta information ...

general page information	
page ID	1483433781
page created by / on	Dang, Quang Anh (EXTERN: ITK)[511] / 2023-05-05 15:59
page editors	• Dang, Quang Anh (EXTERN: ITK)[512]
page commenters	
used template	Template: Guideline (cooking recipe)[513]
used template version	36
page classification	
classification	4.2
confidentiality	internal
approval information	
last approval version	none
last approval date	none

510 https://devstack.vwgroup.com/confluence/x/w5vvG
511 https://devstack.vwgroup.com/confluence/display/~gchogqb
512 https://devstack.vwgroup.com/confluence/display/~gchogqb
513 https://devstack.vwgroup.com/confluence/x/PjJCDg

last approval by	none
change history	

Version	Published	Changed By	Comment
CURRENT (see page 361) **(v. 15)**	May 17, 2024 15:09	Dang, Quang Anh (EXTERN: ITK)[514]	Updated severity rating after Sabrina brought back trivial severity field
v. 14[515]	Feb 21, 2024 11:19	Dang, Quang Anh (EXTERN: ITK)[516]	Update the field after sync with Sabrin Sajonz
v. 13[517]	Feb 14, 2024 15:51	Dang, Quang Anh (EXTERN: ITK)[518]	Added 3 new mandatory fields in bug ticket
v. 12[519]	Jan 08, 2024 18:07	Dang, Quang Anh (EXTERN: ITK)[520]	Update environment
v. 11[521]	Dec 15, 2023 13:58	Dang, Quang Anh (EXTERN: ITK)[522]	Change the title
v. 10[523]	Jul 12, 2023 16:01	Dang, Quang Anh (EXTERN: ITK)[524]	Clarify some TODO fields
v. 9[525]	Jul 11, 2023 12:14	Dang, Quang Anh (EXTERN: ITK)[526]	Table in progress
v. 8[527]	Jul 11, 2023 11:09	Dang, Quang Anh (EXTERN: ITK)[528]	
v. 7[529]	Jun 30, 2023 16:13	Dang, Quang Anh (EXTERN: ITK)[530]	Update status of discussions for the bug ticket fields

514 https://devstack.vwgroup.com/confluence/display/~gchogqb
515 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=3185493472
516 https://devstack.vwgroup.com/confluence/display/~gchogqb
517 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=2765862753
518 https://devstack.vwgroup.com/confluence/display/~gchogqb
519 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=2730771020
520 https://devstack.vwgroup.com/confluence/display/~gchogqb
521 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=2539727492
522 https://devstack.vwgroup.com/confluence/display/~gchogqb
523 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=2468777895
524 https://devstack.vwgroup.com/confluence/display/~gchogqb
525 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1735140524
526 https://devstack.vwgroup.com/confluence/display/~gchogqb
527 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1721426379
528 https://devstack.vwgroup.com/confluence/display/~gchogqb
529 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1721425138
530 https://devstack.vwgroup.com/confluence/display/~gchogqb

Version	Published	Changed By	Comment
v. 6[531]	Jun 20, 2023 17:33	Dang, Quang Anh (EXTERN: ITK)[532]	

Go to Page History[533]

Extracted information based on this fuzzing bug E3-115982[534]

	JIRA	Expected values	Jenkins	Fuzzing log
Header	Title	<repo_name> <bug tilte> in <crashed_function> Example: e3_comp_tlrlib out of memory in vwg::e3p::tlr::createTLRBlobMessage	Name of the repo	Extract the string out of finding log after the finding name and without the line of code Example: [goofy_chicken] out of memory in vwg::e3p::tlr::createTLRUTF8StringMessage (src/Messages/TLRMessageCreator.hpp:315:43) cictl <server name>
Details	Type	Bug	Bug	-
	Priority	High | Medium | Low | Lowest	Manually set by the team after analysis of the bug	Based on Dashboard rating. Extractable via cictl Critical > High High > Medium Medium > Low Low > Lowest

531 https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=1692341016
532 https://devstack.vwgroup.com/confluence/display/~gchogqb
533 https://devstack.vwgroup.com/confluence/pages/viewpreviousversions.action?pageId=1483433781
534 https://devstack.vwgroup.com/jira/browse/E3-115982

	JIRA	Expected values	Jenkins	Fuzzing log
	Affects Version/s	Current PI	Current PI	
	Component/s	Name of the component as specified in the BOM file Example: E3-122191[535] libcommanifestparser	Jenkin will find this information in BOM file	
	Labels:	FuzzingBug	FuzzingBug	
	Resolution:	Unresolved	Unresolved	
	Fix Version/s	Manually set by the team after analysis of the bug	Manually set by the team after analysis of the bug	
General Details	Severity	None | Blocker | Critical | Major | Minor	According to CVSS None: 0.0 Trivial: 0.1 - 3.9 Minor: 4.0-6.9 Major: 7.0 - 8.9 Critical: 9.0 - 10.0 Blocker: Manually individually set if related to safety	Based on Dashboard rating. Extractable via cictl. Standard CVSS (Score 0 -10) Critical 9.0-10.0 High 7.0 -8.9 Medium 4.0 - 6.9 Low 0.1 - 3.9 None 0.0 Reference: https://nvd.nist.gov/vuln-metrics/cvss

535 https://devstack.vwgroup.com/jira/browse/E3-122191

	JIRA	Expected values	Jenkins	Fuzzing log
	Teams	The team responsible for the repository at the time of fuzzing trigger (SBOM)	The team responsible for the repository at the time of fuzzing trigger	
	Component Fix Version	Manually set by the team after analysis of the bug	Manually set by the team after analysis of the bug	
	Master Branch*	This field gives the branch where the issue / bug is found. E3 1.2 future cluster: If the bug exists and persists in any cluster larger than 6 E3 1.2 cluster 3: Master branch E3 1.2 cluster 5: Cluster 5 exists only for winter "Erprobung". This should be removed soon E3 1.2 cluster 6: Develop branch E3 2.0 mainline: ??? For fuzzing, if we do fuzzing on develop branch => choose cluster 6. If the bug also exists in master branch, then cluster 3.	None E3 1.2 future cluster E3 1.2 cluster 3 E3 1.2 cluster 5 E3 1.2 cluster 6 E3 2.0 mainline	None E3 1.2 future cluster E3 1.2 cluster 3 E3 1.2 cluster 5 E3 1.2 cluster 6 E3 2.0 mainline
	Affected Branch		To be filled after analysis.	
Impact Analysis	Affected Teams	Empty at time of generation	Empty at time of generation	
	Error Source	Fuzzing		
	Analysis Result	Manually set by the team after analysis of the bug	Manually set by the team after analysis of the bug	

	JIRA	Expected values	Jenkins	Fuzzing log
	Probability of Occurence	always	always	
	Root Cause	Coding	Coding	
	Reproducible	None	None	
	Code Change Impact	Manually set by the team after analysis of the bug	Manually set by the team after analysis of the bug	
	Security Relevant	Yes	Yes	
	Testcase ID*	Value: "Fuzzing"	Fuzzing	Fuzzing
Planning	Sprint	Manually set by the team after analysis of the bug	Manually set by the team after analysis of the bug	
People	Assignee	Manually set by the team after analysis of the buge	Manually set by the team after analysis of the bug	
	Reporter	1. Who created the fuzz branch? => Assign that person 2. If option 1 is not enough, then find the person who last commited change to the fuzz target => Assign that person		
Dates	Created	The date the bug is found	The date the bug is found	

	JIRA	Expected values	Jenkins	Fuzzing log
Development	Create branch?	Filled automatically if a JIRA ticket is mentioned in a commit	Filled automatically if a JIRA ticket is mentioned in a commit	

	JIRA	Expected values	Jenkins	Fuzzing log

Description	**General information**
	Contact Person : <Replace with your name to get in contact with you if there are any questions/ additional information required>
	<company>
	Phone: <optional>
	DevTeam : <Replace with your DevTeam>
	Used Component Version(s)
	<Replace with information on all used component versions>
		General information
		Contact Person : The same person like the reporter (see above)
		DevTeam : Same as the "team" field above
		Used Component Version(s)
		<Repo name>
		<Fuzzing Branch>
		<Current commit ID from the Fuzzing Branch>
		
		Environment
		Compiled for x86_64, instrumented for fuzzing with cifuzz (libfuzzer).
		(or dependent on the CISence server environment. But this will stay the same for all fuzzing bug)
		Expected behaviour
		No <summary of the bug> e.g., No <out of memory in vwg::e3p::tlr::createTLRUTF8StringMessage src/Messages/TLRMessageCreator.hpp>
		Observed behaviour
		<summary of the bug> e.g., <out of memory in vwg::e3p::tlr::createTLRUTF8StringMessage src/Messages/TLRMessageCreator.hpp>
		Steps to reproduce
		Dang, Quang Anh (EXTERN: ITK) TODO: Update this steps to the current state
		Step to build fuzz target locally
		
		`# Pull and rename the docker image for cifuzz`
		`docker pull e3-e3sp-dockerdev.docker.devstack.vwgroup.com/testing/e3_docker_ci_fuzz:fix_permission-1-fbeb7a8-6df60285 && docker tag e3-e3sp-dockerdev.docker.devstack.vwgroup.com/testing/e3_docker_ci_fuzz:fix_permission-1-fbeb7a8-6df60285 cifuzz:latest`
		
		`# Pull the <repo> repository and checkout to cifuzz custom branch`
		`git checkout custom/cifuzz`
			General information
			Contact Person : <The person who wrote the fuzz target file>
				Expected behaviour
				No <out of memory in vwg::e3p::tlr::createTLRUTF8StringMessage src/Messages/TLRMessageCreator.hpp>
				
				Observed behaviour
				<out of memory in vwg::e3p::tlr::createTLRUTF8StringMessage src/Messages/TLRMessageCreator.hpp>
				
				Traces
				<Attach logfile>
				<Attach crash input file>

	JIRA	Expected values	Jenkins	Fuzzing log

	Environment <Replace with exact information on the (build) environment the issue occurs in>
	Expected behaviour <Replace with description what you would expect instead of the observed issue>
	Observed behaviour <Replace with description what has happened instead of the expected behaviour.>
		``
		# Execute the fuzz test via
		./cifuzz.sh run <fuzz_target> --build-jobs=$(nproc)
		
		# Use Ctrl + C to stop the fuzz test. The fuzz test will find the bug automatically. If not, copy the crash input quick_kiwi to an arbitrary directory. Go inside the docker container to debug via
		./cifuzz.sh /shell
		
		# Replay with the crashing input via:
		NO_CIFUZZ=1 .cifuzz-build/libfuzzer/address+undefined/bin/<fuzz_target> [path to crash input file]
		
		# You can also replay with gdb
		UBSAN_OPTIONS='halt_on_error=1:abort_on_error=1' NO_CIFUZZ=1 gdb -ex run --args .cifuzz-build/libfuzzer/address+undefined/bin/<fuzz_target> [path to crash input file]
		``
		Healing
		n/A, what the step to do to bring it to run.
		Customers Impact
		n/A
		Traces
		<Attach logfile>
		<Attach crash input file>		

	JIRA	Expected values	Jenkins	Fuzzing log
	Steps to reproduce <Replace with all information required to reproduce the bug> **Healing** <What can be done to resolve the issue (e.g. restart, nothing …)> **Customers Impact** <Which kind of impact does the bug have on the end-customer / user> **Traces** <Please provide Logfiles> <Please provide Corefiles> <Please provide Screenshoots>			

	JIRA	Expected values	Jenkins	Fuzzing log
Attachments	**Attachments**	<Attach logfile> <Attach crash input file>		<Attach logfile fetchable with cictl tool from cifuzz server> <Attach crash input file fetchable with cictl tool from cifuzz server>

Meeting Notes:

28 Jul 2023

Participants:

Sajonz, Sabrina (VWIF G-TP/9)

Engelhardt, Maik (T5-PG)

Schaller, Bernd (T5-SF)

- There is a new field called "Affected branch". This filled is filled after analysis => It should be mandatory after analysis. For this, the field must be empty at creation time.
  - Background: For the developer to know which cluster to implement fix on, and also identification of duplicate.
  - Issue:
    - How to find out which commit is responsible for a certain bug? Difficult to automize => If we can find out which commit is responsible => Easy to find out all affected branches
  - Solution:
    - 1.A developers must analyze this.
      TODO: Engelhardt, Maik (T5-PG) will discuss this with Dietterle, Burkhard (TZ-CC) for ideas
    - 2.Build fuzz target for ALL BRANCH and test ALL BRANCH via regression test. => Get a list of affected branches in the end
      Future use: Discuss with Stefan Redmann regarding future test strategy

11 Jul 2023

Participants:

Unknown User (g8qniru)

Muckenschnabl, David Jose (EXTERN: ITK)

Dang, Quang Anh (EXTERN: ITK)

Engelhardt, Maik (T5-PG)

- Systembuild trigger fuzzing
- TODO: Invite Dietterle, Burkhard to the next meeting to discuss setup Jenkins to upload the artefact

30 Jun 2023

Participants:

Engelhardt, Maik (T5-PG)

Weichslgartner, Andreas, Dr. (TX-NS)

Schaller, Bernd (T5-SF)

Dang, Quang Anh (EXTERN: ITK)

Unknown User (egxwokt)

- CI Sense server based their scoring on CVSS Scoring Update cifuzz.yml · CICD-Runtime-1/cpp-fuzz-test2@ec39e9e (vwgroup.com)[536]
- TODO: Need follow up to discuss the rest of the fields

20 Jun 2023

Participants:

Engelhardt, Maik (T5-PG)

Muckenschnabl, David Jose (EXTERN: ITK)

Schaller, Bernd (T5-SF)

Dang, Quang Anh (EXTERN: ITK)

Status:

- Meeting will be moved to the week after

TODO:

- Move meeting to the week after and invite Slosarek, Stefan (TZ-CB)(if possible) and Dietterle, Burkhard (TZ-CC) 30 Jun 2023 at 15:15

05 May 2023 Meeting Notes with Florian Baho

536 https://git.qs.hub.vwgroup.com/CICD-Runtime-1/cpp-fuzz-test2/actions/runs/14715/jobs/42733

- Fields:
  - Error Source: Fuzzing
  - Priority depends on Severity field
  - Teams: Which team found the bug?
  - Affected Team: Bug is from which team?
- TODO:
  - Create mapping table https://devstack.vwgroup.com/confluence/display/E3SEC/Jira+Bugs+fields+mapping+table+between+E3+and+Jenkins+and+CIFuzz+Dashboard
    - JIRA
    - Jenkins
    - CIfuzz log

19 Apr 2023 Meeting Notes with Maik Engelhardt

Point of Contact: TODO
3 steps approach:

- (Automatic) Analyze the Cifuzz Result in Jenkins or GitHub
  - Idea: FuzzTest are tested locally before uploading to the CIFuzz server => False Positive rate is low => Accept all reports
  - We should create bugs for all severity. CIFuzz server gives severity rating
  - Best practice: Write several fuzz targets instead of 1 big one to run parallelly (if possible)
  - Check if this bug exists already exists
    - Use Stacktrace header to track e.g "[upbeat_lemur] out of memory in vwg::e3p::tlr::createTLRBlobMessage (src/Messages/TLRMessageCreator.hpp:377:9)"
    - Which status it has? If it is closed, then it can a new bug or the fix was not good enough
  - Attach the link to the finding on the CIFuzz Dashboard
- (Automatic) If there is a finding, double check which team to report to
  - There is a trace between Bitbucket repo and team name. Jenkins can extract this and put info into "Affected Team" teams
- GitHub Action/Jenkins

18.1 2023-12-15 Sync Fuzzing Pipeline with Jenkins

- [x] Engelhardt, Maik (T5-PG)
- [x] Dang, Quang Anh (EXTERN: ITK)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)

Agenda:

- Jenkins would have to build fuzz target upon commit/push
  - We can do POC on the Jenkins QS Server: https://jenkins.ioi.qs.imfa.hub.vwgroup.com/job/SANDBOXES/job/AutoCops/ (VPN is needed for access)
  - Later, we would have to contact CICD (PMT team?) for them to set it up on their
- After building the fuzz target, Jenkins would have to bundle the fuzzing artefact and upload it to the CI Sense Server https://cifuzz.qs.hub.vwgroup.com/
- Jenkins would have to monitor the fuzzing progress via fuzzing campaign ID
- Upon finding bug, Jenkins would have to create a Jira Bug ticket

Notes:

- Status
  - The production server for CISense is taking even longer to set up -> a POC of the pipeline will be se-up using the QS server.
- Jenkins
  - Dashboard → Sandbox → AutoCops → CIFuzz (created folder) → Choose "Pipeline" → →
    - Issue: Our container is not supported.

TODOs:

- [ ] Set-up boundary client for our VMs to access Jenkins QS Server
- [ ] How to use custom unsupported docker container?
- [ ] We need access to https://jenkins.ioi.qs.imfa.hub.vwgroup.com/job/SANDBOXES/job/AutoCops/, https://jenkins.ioi.qs.imfa.hub.vwgroup.com/job/SANDBOXES/job/AutoCops/job/CI%20Fuzz/job/CI%20Fuzz%20Pipeline%20Project/, build, configure rights. Contact: Smaoui, Yacine (TP-S2) - CICD pipeline, jenkins expert
- [ ] Discuss when the CIFUZZ pipeline should be triggered

Helpful Links

Example for Jenkinsfile:

https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3_aux_kpi_lake/browse/Jenkinsfile

Jenkins Shared library:

https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_pipeline_generic_shared_lib_premium/browse/src/de/vwg/e3/ioi/premium/nodes/Nodes.groovy

Contact: Smaoui, Yacine (TP-S2) - CICD triton pipeline, jenkins expert

Contact: Baumung, Wjatscheslav (EXTERN: MHP) -

18.2 2024-01-17 Sync Fuzzing Pipeline with Jenkins Part 2

- [x] Engelhardt, Maik (T5-PG)
- [x] Dang, Quang Anh (EXTERN: ITK)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [x] Slosarek, Stefan (TZ-CB)
- [x] Baumung, Wjatscheslav (EXTERN: MHP)
- [x] Smaoui, Yacine (TZ-CC)
- [x] Engelhardt, Maik (T5-PG)

Agenda:

- Jenkins would have to build fuzz target upon commit/push
  - We can do POC on the Jenkins QS Server: https://jenkins.ioi.qs.imfa.hub.vwgroup.com/job/SANDBOXES/job/AutoCops/ (VPN is needed for access)
  - Later, we would have to contact CICD (PMT team?) for them to set it up on their
- After building the fuzz target, Jenkins would have to bundle the fuzzing artefact and upload it to the CI Sense Server https://cifuzz.qs.hub.vwgroup.com/
- Jenkins would have to monitor the fuzzing progress via fuzzing campaign ID
- Upon finding bug, Jenkins would have to create a Jira Bug ticket

Notes:

- Jenkin is triggered by commit/push. What if Jenkins crashes, where does it pick up?
  - Problem: Handing over to CISense Server is "not feasible"
    - Wait long for CISense server
    - If Jenkins crashes? => How do we pick up where it started?
- Can the test itself run on Jenkins Server?
- Current approach for Unit Test:
  - Unit Tests is run locally on Jenkins Server
  - When it is done => The result is uploaded to Parasoft server
- Pros for the CISense server:
  - CISense is stateful => It saves the corpus (progress of fuzzing)
  - Server is Kubernetes structure => scaling
- **Solution**:

-

- No extra pipeline for this. It should be integrated to an existing pipeline which also does unit test etc.
- Build fuzz target => Set a timeout
- Request for result:
  - Coverage LCOV
  - Bug available?
- If the CISense Server crashes => consider the fuzz test FAIL
- The fuzz test should better run on a develop branch (the less regular commits the better)
- ⚠️ Timeout for the fuzz run?
  - longest path jenkin has e.g. conan package build ~ 1h
  - think about nightly build later as improvement
- Customer Need Ticket to use: https://devstack.vwgroup.com/jira/browse/PMTC-1967

Tutorial on how to fuzz locally and also on jenkins server based on the example of Remote Repair:

Tutorial on how to build Remote Repair Fuzzing example

```
// First, how to build and fuzz locally and on jenkins server
// Clone the repository
git clone ssh://git@devstack.vwgroup.com:7999/e3app/e3_app_remote_repair.git

// Switch to our fuzzing branch
git checkout custom/cifuzz_newbuild

// Build the fuzzing binary
./build.sh --docker --target fuzzing --profile fuzzing-x86_64 --buildfolder=build --
verbose --fuzzing -j2

// After that, you will find 3 fuzz targets in build/bin/ subdirectory. To execute a
fuzz target, perform this
./cifuzz.sh run ./build/bin/<name of the fuzz target>

// To build coverage binary:
./build.sh --docker --target fuzzing --coverage --profile fuzzing-x86_64 --
buildfolder=build-cov --verbose --fuzzing -j2
```

```
// To generate lcov file. Note that you will need a corpus for this, so you have to run
the fuzz test for a while before
./cifuzz.sh coverage --add-corpus .cifuzz-corpus/build/bin/fuzz_Configuration/ build-
cov/bin/<name of the fuzz target> --output=lcov.info --format=lcov

// To generate a html coverage report. Note that you will need a corpus for this, so
you have to run the fuzz test for a while before
./cifuzz.sh coverage --add-corpus .cifuzz-corpus/build/bin/<name of the fuzz target>/
build-cov/bin/<name of the fuzz target> --output=tests/fuzzing/<give a directory name>

// Note that the same tutorial/documentation can be found in each of the fuzz target
cpp file in ./tests/fuzzing/fuzz_<...>.cpp too.
```

Requirement for the building to work on Jenkins. The full tutorial how to set up the environment can be found here: CI-Fuzz Proof of Concept (see page 66), chapter 4.3.3 Setting up cifuzz environment onward. Basically, to summarize, the jenkins server will need: ,

- latest version of the cifuzz.sh script[537] which is already inside the remote repair reporsitoy branch *custom/cifuzz_newbuild*
- the docker container: *testing/e3sdk-build-bp1-commonpart:feature-vwos-7577-fuzzing-2-b1b1824-a9dc2382* which is already configured for target fuzzing inside *e3sdk_metadata*

TODOs:

- [x] Set-up boundary client for our VMs to access Jenkins QS Server
- [ ] How to use custom unsupported docker container?
- [ ] We need access to https://jenkins.ioi.qs.imfa.hub.vwgroup.com/job/SANDBOXES/job/AutoCops/, https://jenkins.ioi.qs.imfa.hub.vwgroup.com/job/SANDBOXES/job/AutoCops/job/CI%20Fuzz/job/CI%20Fuzz%20Pipeline%20Project/, build, configure rights. Contact: Smaoui, Yacine (TP-S2) - CICD pipeline, jenkins expert
- [ ] Discuss when the CIFUZZ pipeline should be triggered
- [ ] Smaoui, Yacine (TZ-CC) Jenkins Team needs a module that builds the first target Summarize the step here on this page and send it to the round
- [ ] POC page draft version is not accessible for Jenkins Team

Helpful Links

Proof Of Concept Fuzzing All in One documentation page:
CI-Fuzz Proof of Concept (see page 136)

Example for Jenkinsfile:

https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3_aux_kpi_lake/browse/Jenkinsfile

537 https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3-aux-security-repo/browse/DAST/cifuzz?at=refs%2Fheads%2Fdevelop

Jenkins Shared library:

https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_pipeline_generic_shared_lib_premium/browse/src/de/vwg/e3/ioi/premium/nodes/Nodes.groovy

Contact: Smaoui, Yacine (TP-S2) - CICD triton pipeline, jenkins expert

Contact: Baumung, Wjatscheslav (EXTERN: MHP) -

18.3 2024-05-27 Discuss running Fuzz-Targets directly in Jenkins

18.3.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Dang, Quang Anh (EXTERN: ITK)

Weichslgartner, Andreas, Dr. (TX-NS)

Schaller, Bernd (T5-SF)

Smaoui, Yacine (TZ-CC) (CICD Pipeline Team)

Niklas Henrich (Code Intelligence)

Khaled Yakdan (Code Intelligence)

Ty (Code Intelligence)

18.3.2 FYI

Slosarek, Stefan (TZ-CB)

Baumung, Wjatscheslav (EXTERN: MHP)

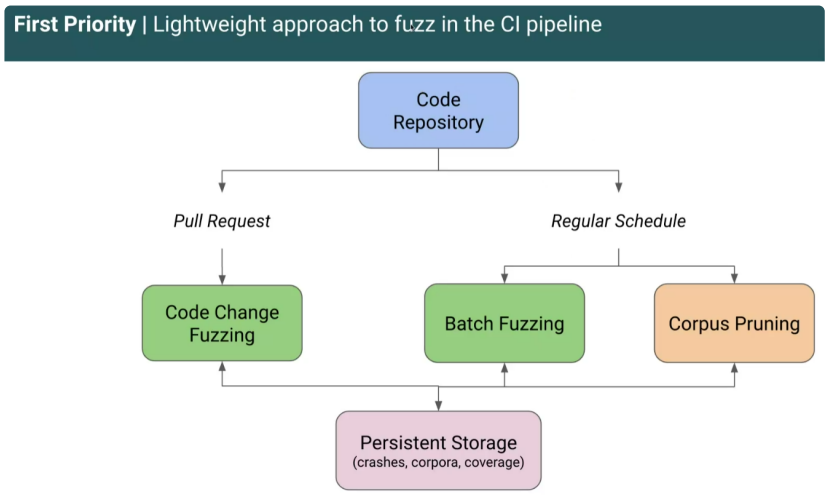
Engelhardt, Maik (T5-PG)

Slosarek, Stefan (TZ-CB)

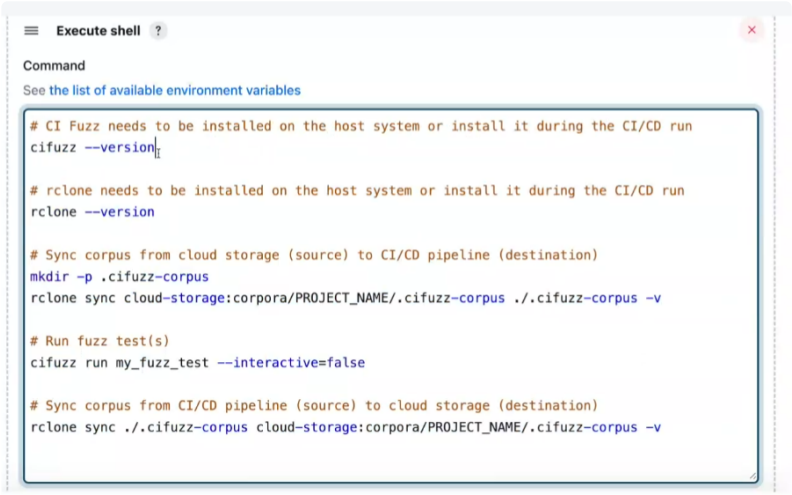
Willers, Martin (T5-PG)

18.3.3 Agenda

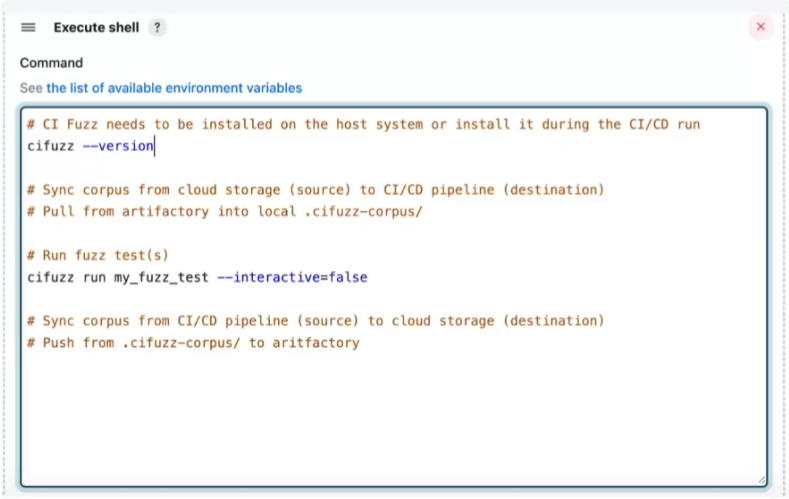
- Niklas shows overview and examples for CICD Pipeline fuzzing



- What do we need to run fuzzing?
  - (CI) Persistent Storage to sync the corpus to reuse the corpus from run to run
    Proposal: Azure cloud storage to store the corpus



  - (CI) Proposal 2: Upload corpus to artefactory



- (CICD) Current CICD Pipeline runs on Cubernetes Cluster. They try to keep Cubernetes Cluster as stateless as possible => Sometimes it would be wiped and destroyed
  Proposal: PVS persistent storage within this Cubernetest Cluster => It will be wiped after a while
  - A volume
- (Andreas) Issue: Github Runner doesn't have PVS Volume
  Proposal: Upload the corpus to the JFROG Artefactory
- (CICD) No technical blockers. It is possible.

18.3.4 TODOs

- [x] Muckenschnabl, David Jose (EXTERN: ITK) Set-Up a meeting, with Slosarek, Stefan (TZ-CB) Willers, Martin (T5-PG) Smaoui, Yacine (TZ-CC) Schaller, Bernd (T5-SF)
- [ ] Muckenschnabl, David Jose (EXTERN: ITK) When meeting for implementation invite Code Intelligence

18.3.5 Helpful Link

Proof Of Concept Fuzzing All in One documentation page:
CI-Fuzz Proof of Concept (see page 136)

Example for Jenkinsfile:

https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3_aux_kpi_lake/browse/Jenkinsfile

Jenkins Shared library:

https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_pipeline_generic_shared_lib_premium/browse/src/de/vwg/e3/ioi/premium/nodes/Nodes.groovy

Contact: Smaoui, Yacine (TP-S2) - CICD triton pipeline, jenkins expert

Contact: Baumung, Wjatscheslav (EXTERN: MHP) -

18.4 2024-06-03 Discuss running Fuzz-Targets directly in Jenkins - Planning

18.4.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Dang, Quang Anh (EXTERN: ITK)

Weichslgartner, Andreas, Dr. (TX-NS)

Schaller, Bernd (T5-SF)

Smaoui, Yacine (TZ-CC) (CICD Pipeline Team)

Willers, Martin (T5-PG)

Slosarek, Stefan (TZ-CB)

18.4.2 FYI

Baumung, Wjatscheslav (EXTERN: MHP)

Engelhardt, Maik (T5-PG)

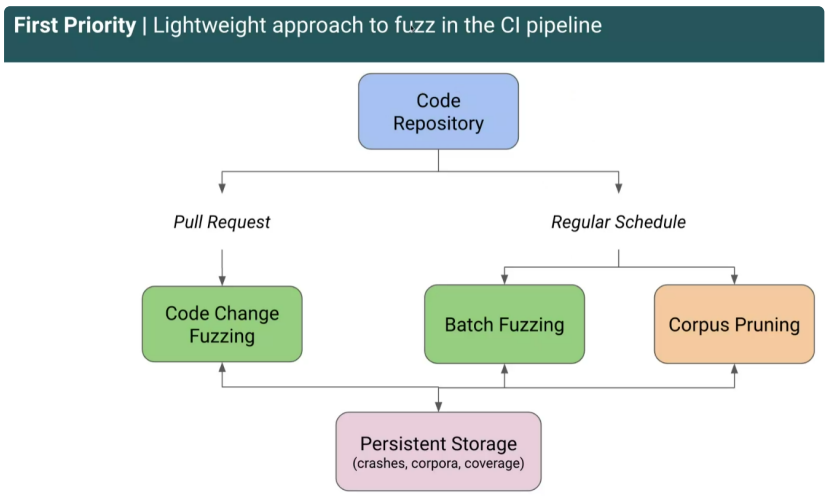
Slosarek, Stefan (TZ-CB)

18.4.3 Agenda

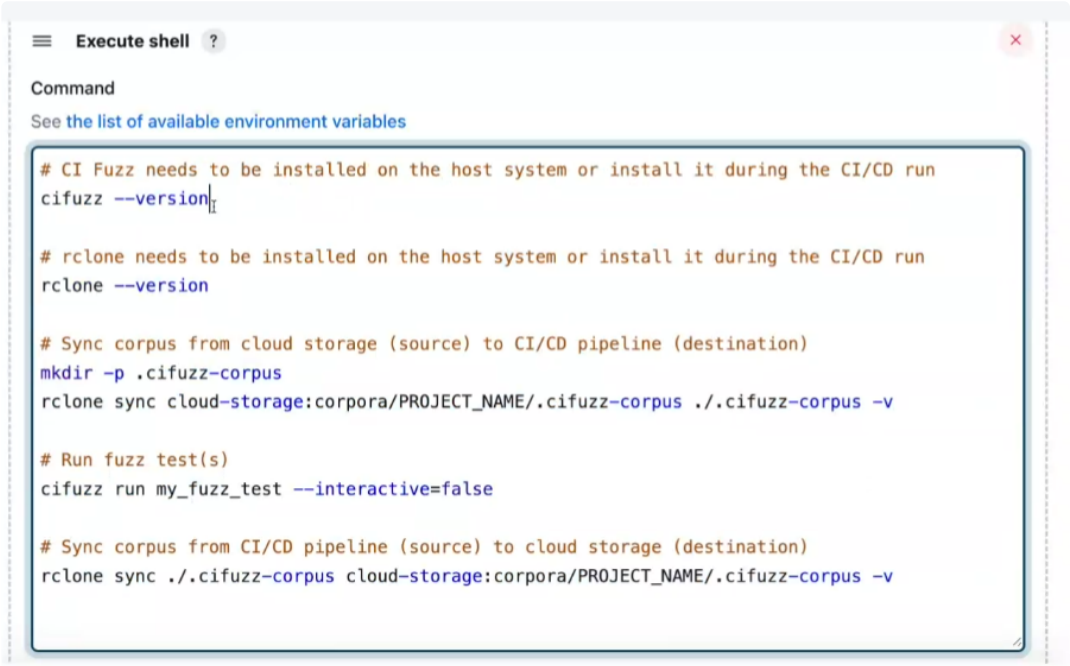
- How to implement the lightweight fuzzing on Jenkin Pipeline
  - Requirements and steps
- Clarify access and steps
- Priority
  - Discussion with Bernd Schaller, Daniel Neeb
  - Priority discussion takes place weekly in CICD
- Which tool to use to upload/download the tar corpus to the artefactory?
  - Curl
  - **How to upload/download to artefactory**

```
curl  -u $ARTIFACTORY_USER:$JFROG_API_KEY "https://jfrog.hub.vwgroup.com/
artifactory/ciids-generic-internal/cifuzz/corpus/${PROJ_NAME}_corpus.gz" |
tar -xz -C .cifuzz-corpus

tar -czf - .cifuzz-corpus | curl -u $ARTIFACTORY_USER:$JFROG_API_KEY -T -
"https://jfrog.hub.vwgroup.com/artifactory/ciids-generic-internal/cifuzz/
corpus/${PROJ_NAME}_corpus.gz"
```

18.4.4 Information from past Discussion



- What do we need to run fuzzing?
  - (CI) Persistent Storage to sync the corpus to reuse the corpus from run to run
    Proposal: Azure cloud storage to store the corpus



- (CI) Proposal 2: Upload corpus to artefactory

Execute shell ?

Command

See the list of available environment variables

```
# CI Fuzz needs to be installed on the host system or install it during the CI/CD run
cifuzz --version

# Sync corpus from cloud storage (source) to CI/CD pipeline (destination)
# Pull from artifactory into local .cifuzz-corpus/

# Run fuzz test(s)
cifuzz run my_fuzz_test --interactive=false

# Sync corpus from CI/CD pipeline (source) to cloud storage (destination)
# Push from .cifuzz-corpus/ to aritfactory
```

- Customer need ticker:
  - For the build process:
    PMTC-1682[538] - Integrate cifuzz build process into the current one IN IMPLEMENTATION
  - For the build process and CICD Pipeline
    PMTC-1967[539] - Introduce fuzzing into the build environment CLOSED

18.4.5 TODOs

- Muckenschnabl, David Jose (EXTERN: ITK) When we meet for implementation invite Code Intelligence
- Muckenschnabl, David Jose (EXTERN: ITK) Ask how does cifuzz find the .cifuzz-corpus folder and whether it is possible to parameterize the path to write and read corpus
  - Niklas Henrich commented:

    Hi David!

    Yes, a symlink should work. I just tested it locally and there were now problems.

    Cheers,

    Niklas

1. **Helpful Link**
2. 2024-05-27 Discuss running Fuzz-Targets directly in Jenkins (see page 379) (meeting where Code Intelligence explained proposal for how to execute fuzzing targets directly on the pipeline)

538 https://devstack.vwgroup.com/jira/browse/PMTC-1682?src=confmacro
539 https://devstack.vwgroup.com/jira/browse/PMTC-1967?src=confmacro

3. Proof Of Concept Fuzzing All in One documentation page:
   CI-Fuzz Proof of Concept (see page 136)
4. Example for Jenkinsfile:
5. https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3_aux_kpi_lake/browse/Jenkinsfile

6. Github Enterprise PoC:
7. cifuzz_gh_runner/.github/workflows/fuzz.yml at main · TS/cifuzz_gh_runner (vwgroup.com)[540]

```
name: fuzz

env:
  PROJ_NAME: "FUZZO"
  VERSION: "0.1.0"

# Controls when the workflow will run
on:
  # Triggers the workflow on push or pull request events but only for the main branch
  push:
  pull_request:
    branches: [master]

  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:
  # run on a release to create  adelivery
  release:
    types: [published]


jobs:
  fuzz:
    runs-on: [auto-buildah-base]
    env:
      JFROG_API_KEY: ${{ secrets.JFROG_API_KEY }}
      ARTIFACTORY_USER: ${{ secrets.ARTIFACTORY_USER }}
      PROJECT_FOLDER: "/project/"
      ARTIFACTORY_CORPUS_URL: "https://jfrog.hub.vwgroup.com/artifactory/ciids-generic-
internal/cifuzz/corpus/"
    steps:
      - uses: actions/checkout@v2
      - name: login buildah Jfrog
        run: |
          buildah login -u $ARTIFACTORY_USER -p $JFROG_API_KEY  jfrog.hub.vwgroup.com
      - name: get cifuzz container
        run: |
          containerid=$(buildah from jfrog.hub.vwgroup.com/ciids-docker-internal/
cifuzz-test:${VERSION})
          echo "containerid=$containerid" >> $GITHUB_ENV
      - name: download corpus if exists
        run: |
```

540 https://git.hub.vwgroup.com/TS/cifuzz_gh_runner/blob/main/.github/workflows/fuzz.yml

```
          HTTP_STATUS=$(curl -u "$ARTIFACTORY_USER:$JFROG_API_KEY" -o /dev/null -w "%
{http_code}" -s -I "${ARTIFACTORY_CORPUS_URL}${PROJ_NAME}_corpus.gz")
          if [ "$HTTP_STATUS" -eq 200 ]; then
            echo "Corpus found. Download it"
            curl  -u $ARTIFACTORY_USER:$JFROG_API_KEY "${ARTIFACTORY_CORPUS_URL}$
{PROJ_NAME}_corpus.gz" | tar -xz
            curl  -u $ARTIFACTORY_USER:$JFROG_API_KEY -O "${ARTIFACTORY_CORPUS_URL}$
{PROJ_NAME}_corpus.gz.sha256.txt"
          fi
      - name: fuzz
        run: buildah  run -v $(pwd):/$PROJECT_FOLDER ${{env.containerid}} /usr/bin/bash
-c "cifuzz run canfuzz --interactive=false --timeout=10s -C=$PROJECT_FOLDER"
      - name: upload corpus
        run: |
          if [ -f ${PROJ_NAME}_corpus.gz.sha256.txt ]; then
            curl  -u $ARTIFACTORY_USER:$JFROG_API_KEY "${ARTIFACTORY_CORPUS_URL}$
{PROJ_NAME}_corpus.gz.sha256.txt" -o ${PROJ_NAME}_corpus_tmp.gz.sha256.txt
            hash1=$(cut -d ' ' -f 1  ${PROJ_NAME}_corpus.gz.sha256.txt)
            hash2=$(cut -d ' ' -f 1  ${PROJ_NAME}_corpus_tmp.gz.sha256.txt)
            if [ "$hash1" != "$hash2" ]; then
              echo "Remote corpus changed. Merge it."
              mkdir -p new_corpus
              curl  -u $ARTIFACTORY_USER:$JFROG_API_KEY "${ARTIFACTORY_CORPUS_URL}$
{PROJ_NAME}_corpus.gz" | tar -xz -C new_corpus
              buildah  run -v $(pwd):/$PROJECT_FOLDER ${{env.containerid}} /usr/bin/
bash -c "cifuzz run canfuzz --interactive=false --engine-arg='-merge=1'  --engine-
arg='.cifuzz-corpus/canfuzz/' --engine-arg='new_corpus/.cifuzz-corpus/canfuzz/'
-C=$PROJECT_FOLDER"
            fi
          fi
          tar -czf ${PROJ_NAME}_corpus.gz .cifuzz-corpus
          sha256sum ${PROJ_NAME}_corpus.gz > ${PROJ_NAME}_corpus.gz.sha256.txt
          curl -u $ARTIFACTORY_USER:$JFROG_API_KEY -T ${PROJ_NAME}_corpus.gz.sha256.txt
"${ARTIFACTORY_CORPUS_URL}${PROJ_NAME}_corpus.gz.sha256.txt"
          curl -u $ARTIFACTORY_USER:$JFROG_API_KEY -T ${PROJ_NAME}_corpus.gz "$
{ARTIFACTORY_CORPUS_URL}${PROJ_NAME}_corpus.gz"
```

1. Jenkins Shared library:
2. https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_pipeline_generic_shared_lib_premium/browse/src/de/vwg/e3/ioi/premium/nodes/Nodes.groovy
3. Contact: Smaoui, Yacine (TP-S2) - CICD triton pipeline, jenkins expert
4. Contact: Baumung, Wjatscheslav (EXTERN: MHP) -

18.5 2024-07-05 Discuss Building Blocks

18.5.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Dang, Quang Anh (EXTERN: ITK)

Smaoui, Yacine (TZ-CC) **CICD Pipeline Team**

Neeb, Daniel (T5-SG)

Kahled Yakdan **Code Intelligence**

Herlambang, Afrizal (TZ-CC) **CICD Pipeline Team** provider of building block framework

Jacek Trossen **Code Intelligence**

Slosarek, Stefan (TZ-CB)

18.5.2 FYI

Baumung, Wjatscheslav (EXTERN: MHP)

Engelhardt, Maik (T5-PG)

Slosarek, Stefan (TZ-CB)

Weichslgartner, Andreas, Dr. (TX-NS)

Schaller, Bernd (T5-SF)

Willers, Martin (T5-PG)

18.5.3 Agenda

- E3ASWF-7343[541] - [Triton] Integrate CIFuzz into Triton CLOSED We are discussing this ticket
- What is the building block that CICD needs
- PMT wants to introduce fuzzing in a building block
- A building block is a block of technology that can be reused in other pipelines
  - The idea is to have fuzzing as a building block so anyone in Cariad can use it
- Afrizal introduced building blocks https://pages.git.hub.vwgroup.com/CARIAD/tools-building-blocks-central-doc-src/
- A building block for running the fuzz test will be created Muckenschnabl, David Jose (EXTERN: ITK) Dang, Quang Anh (EXTERN: ITK)
- If we need support for the building block we ask Herlambang, Afrizal (TZ-CC)
- If we need support for the pipeline we ask Smaoui, Yacine (TZ-CC)

Example https://git.hub.vwgroup.com/CARIAD/tools-building-block-sca-src

541 https://devstack.vwgroup.com/jira/browse/E3ASWF-7343?src=confmacro

18.6 2024-07-23 Discuss current cifuzz build process and merging of 2 PRs

18.6.1 Participants

Muckenschnabl, David Jose (EXTERN: ITK)

Dang, Quang Anh (EXTERN: ITK)

Baumung, Wjatscheslav (EXTERN: MHP)

Weichslgartner, Andreas, Dr. (TX-NS)

Willers, Martin (T5-PG)

18.6.2 FYI

Engelhardt, Maik (T5-PG)

Slosarek, Stefan (TZ-CB)

Schaller, Bernd (T5-SF)

18.6.3 Agenda

- Status of merging 2 PRs into develop branch:
  - docker_e3sdk_commonpart_build_bp1:https://devstack.vwgroup.com/bitbucket/projects/E3INT/repos/docker_e3sdk_commonpart_build_bp1/pull-requests/297/overview
    - Future plan for fuzzing build profile? Where and how to inject them?
    - Is the changelog.md and versioning ok? The current master branch is on 1.8.0 but develop branch still on 1.7.3?
  - templates-e3comp: https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/templates-e3comp/pull-requests/100/overview
- Fuzz test run often has segmentation fault due to the version of llvm. There are 2 solutions for this (see. https://docs.code-intelligence.com/troubleshooting)
  This could be caused by the default ASLR (Address Space Layout Randomization) entropy value set by the OS. Sanitizers in earlier LLVM versions seem to have issues with a too high value
  Solution (any feasible)?
  - Upgrade LLVM to version >= 17
  - Orset the entropy value to `28` or `30`

- **Setting entropy value**

```
sudo cat /proc/sys/vm/mmap_rnd_bits
sudo sysctl vm.mmap_rnd_bits=28
```

- A team suggested to have the option to build fuzz target on other branches as well. They would like an option to enable it optionally

18.6.4 Protocol

- What to do with profile issue?
  - Idea: Injecting profile when building fuzzing stage
  - Problem:
    - How would a local user build locally?
      => Pulling the profile from an independent repo/source and put it locally
- Regarding segmentation fault
  - Editing kernel configuration is not feasible
  - Updating clang and llvm to version >= 17?
    - If we update clang to 17, then clang-tidy would have to be updated to clang-tidy 17
    - Create a separate branch and PR
    - Upgrade to newest version of clang
    - Do a system build to see if anything break
- Versioning:
  - 1.9.0 - 23.07.2024 - Parasoft 2023.2.1 - vwos/std 1.9.0
  - Tagging 1.9.0 after merging
- Both PR can be approved and merged

18.6.5 Links

- CICD Jira tickets:
  - E3ASWF-7343[542] - [Triton] Integrate CIFuzz into Triton CLOSED
  - PMTC-1967[543] - Introduce fuzzing into the build environment CLOSED

542 https://devstack.vwgroup.com/jira/browse/E3ASWF-7343?src=confmacro
543 https://devstack.vwgroup.com/jira/browse/PMTC-1967?src=confmacro

18.6.6 TODO

- [ ] Dang, Quang Anh (EXTERN: ITK) Muckenschnabl, David Jose (EXTERN: ITK) plan a session with Martin Willers to hands-on write a fuzz target for 1 component

18.7 2024-08-05 Sync regarding future work for Fuzzing Overview page

18.7.1 Participants

Dang, Quang Anh (EXTERN: ITK)

Bartels, Bjoern (EXTERN: Datendeuter)

Unknown User (e7hqlnm)

Kutscher, Max (EXTERN: PTG)

18.7.2 FYI

Engelhardt, Maik (T5-PG)

Schaller, Bernd (T5-SF)

Muckenschnabl, David Jose (EXTERN: ITK)

18.7.3 Agenda

- The current sneak-peek page
- How to improve the Sneak-Peek page
- Automatic Jira bug creation

18.7.4 Protocol

- The official overview page which will be automatically updated regularly
  E3 1.2 - Fuzzing List[544]
- Feature requests / Feedback for the page that we would like to implement:
  - [ ] Add the team name to the page

544 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/3041828882/E3+1.2+-+Fuzzing+List

- [ ] Add tabs similiar to ICP-14 Unit Test Sanitizer Report[545] for develop branch, CL3, CL6, CL8 etc.
- [ ] Separate the page into different ARTs e.g. ICP and BSrv
- [ ] Add the repository name like a clickable name such as e3_comp_otam_common
- [ ] Status of the fuzz tests
  - We generate a fuzzing report with following information:
    - List of fuzz targets
    - Coverage information
    - Running duration and #inputs
    - Findings
    - Status of run
      - TODO: Not started nor created yet, meaning no report for the target compared to the e3fuzz.xml
      - ERROR: Anything related to build or run error
      - SUCCESSFUL: Fuzz tests ran successfully and no finding/bug was identified
      - FINDINGS: Bugs identified
- [ ] Jira bug reporting upon bug found
  - Datendeuter will generate a bug ticket based on the information we provide in the report in TARBALL
  - List of Jira Bug fields that need to be filled can be found here: Fuzzing Pipeline incl. JIRA bug reporting with Jenkins (see page 361)
- [ ] (low prio) Add relative paths to the function to fuzz

18.7.5 Links

- Sneak-peek page:
  - [!SNEAKPEEK!] Fuzzing Function Overview Team Update Core (OTA-Master)[546]
  - https://devstack.vwgroup.com/confluence/display/E3SPVMAN/%5B%21DEV%21%5D+Fuzzing+List
- Jira Tickets
  - VWOS-12019[547] - [Template] General hardening via CodeMarking WON'T FIX
  - VWOS-12020[548] - Define means to identify fuzzable functions DONE

545 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1237654038/ICP-14+Unit+Test+Sanitizer+Report
546 https://devstack.vwgroup.com/confluence/spaces/E3SPVMAN/pages/2660667242/SNEAKPEEK+Fuzzing+Function+Overview+Team+Update+Core+OTA-Master
547 https://devstack.vwgroup.com/jira/browse/VWOS-12019?src=confmacro
548 https://devstack.vwgroup.com/jira/browse/VWOS-12020?src=confmacro

- ☑ E3-134268[549] - [Data lake KPI] Create report for fuzzing and fuzzed compoents DONE

18.7.6 TODO

- [ ] Security team will start writing the fuzzing report to upload as part of the TARBALL in artefactory
- [ ] Datendeuter will start with the feature requests list above

18.8 2024-08-12 Sync Fuzzing Pipeline with Jenkins Part 3

- [x] Dang, Quang Anh (EXTERN: ITK)
- [x] Baumung, Wjatscheslav (EXTERN: MHP)

Agenda:

- cifuzz as conan package?
- current status of Pipeline
- where to upload the corpus where we can modify it? Currently, it is not modifiedable. Yaccine said to upload it here: https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-components-snapshot in some subfolder.
  We are currently trying out https://jfrog.devstack.vwgroup.com/ui/native/e3-e3sp-gen-teamcicd/cifuzz_corpus/e3_comp_otam_common/
- PMT Solution ticket:
  E3ASWF-11003[550] - Build components with CIFuzz profile in Triton / System Build CLOSED

Protocol:

- Issue:
  - System Build to with our exact fuzzing build profiles fails => not the fault from our team
- How does injecting profile work?
  - Pipeline search for the profile in the bp1 container
  - If not found => use the hardcoded profile in InsertConanProfilesFuzzing.groovy
- Corpus overwrite. Slavic doesn't know if the Jenkin user has overwrite access for the corpus
- Baumung, Wjatscheslav (EXTERN: MHP) will implement following 2 steps documented here
  - https://devstack.vwgroup.com/bitbucket/projects/E3SWCDEP/repos/e3_comp_otam_common/browse/tests/fuzzing/fuzz_Utilities.cpp?at=refs%2Fheads%2Fcustom%2Fcifuzz_2024_01#15,21
- After the building block, jenkins will have to call the building block to execute the fuzz tests like follows:
  - https://git.hub.vwgroup.com/CARIAD/tools-bb-fuzzing-src/blob/feature/VWOS-15181-fuzzing_building_block/Jenkinsfile#L51-L60

549 https://devstack.vwgroup.com/jira/browse/E3-134268?src=confmacro
550 https://devstack.vwgroup.com/jira/browse/E3ASWF-11003?src=confmacro

TODOs:

- [ ] How to use custom unsupported docker container?
- [ ] We need access to https://jenkins.ioi.qs.imfa.hub.vwgroup.com/job/SANDBOXES/job/AutoCops/, https://jenkins.ioi.qs.imfa.hub.vwgroup.com/job/SANDBOXES/job/AutoCops/job/CI%20Fuzz/job/CI%20Fuzz%20Pipeline%20Project/, build, configure rights. Contact: Smaoui, Yacine (TP-S2) - CICD pipeline, jenkins expert
- [ ] Discuss when the CIFUZZ pipeline should be triggered
- [x] Smaoui, Yacine (TZ-CC) Jenkins Team needs a module that builds the first target Summarize the step here on this page and send it to the round
- [x] POC page draft version is not accessible for Jenkins Team
- [ ] Dang, Quang Anh (EXTERN: ITK) Update building block to the next version which expects to be inside the docker container already

Helpful Links

Proof Of Concept Fuzzing All in One documentation page:
CI-Fuzz Proof of Concept (see page 136)

Example for Jenkinsfile:

https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3_aux_kpi_lake/browse/Jenkinsfile

Jenkins Shared library:

https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_pipeline_generic_shared_lib_premium/browse/src/de/vwg/e3/ioi/premium/nodes/Nodes.groovy

Contact: Smaoui, Yacine (TP-S2) - CICD triton pipeline, jenkins expert

Contact: Baumung, Wjatscheslav (EXTERN: MHP) -

18.9 2024-08-13 Sync Fuzzing Pipeline with Jenkins Part 4 - System Build & Building Block

- [x] Dang, Quang Anh (EXTERN: ITK)
- [x] Baumung, Wjatscheslav (EXTERN: MHP)
- [x] Willers, Martin (T5-PG)

Agenda:

- System build fails
- Build dependencies for Clang 15

- Jenkin after build => Expects to be in a docker container. Can we provide the name of the container/image and start/create the container to run?
  Or should we rewrite the script completely?

Protocol:

- Wjatsches Build Error is because of missing profile
- Trying to build conan package with clang15
- ASAN_OPTION=... setzen

TODOs:

- [ ] How to use custom unsupported docker container?
- [ ] We need access to https://jenkins.ioi.qs.imfa.hub.vwgroup.com/job/SANDBOXES/job/AutoCops/, https://jenkins.ioi.qs.imfa.hub.vwgroup.com/job/SANDBOXES/job/AutoCops/job/CI%20Fuzz/job/CI%20Fuzz%20Pipeline%20Project/, build, configure rights. Contact: Smaoui, Yacine (TP-S2) - CICD pipeline, jenkins expert
- [ ] Discuss when the CIFUZZ pipeline should be triggered
- [x] Smaoui, Yacine (TZ-CC) Jenkins Team needs a module that builds the first target
  Summarize the step here on this page and send it to the round
- [x] POC page draft version is not accessible for Jenkins Team
- [ ] Dang, Quang Anh (EXTERN: ITK) Update building block to the next version which expects to be inside the docker container already

Helpful Links

Proof Of Concept Fuzzing All in One documentation page:
CI-Fuzz Proof of Concept (see page 136)

Example for Jenkinsfile:

https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3_aux_kpi_lake/browse/Jenkinsfile

Jenkins Shared library:

https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_pipeline_generic_shared_lib_premium/browse/src/de/vwg/e3/ioi/premium/nodes/Nodes.groovy

Contact: Smaoui, Yacine (TP-S2) - CICD triton pipeline, jenkins expert

Contact: Baumung, Wjatscheslav (EXTERN: MHP) -

18.10 2024-08-30 Sync regarding BB-reporting

18.10.1 Participants

Unknown User (e7hqlnm)

Muckenschnabl, David Jose (EXTERN: ITK)

18.10.2 Agenda

- iscuss implementation of reporting-BB

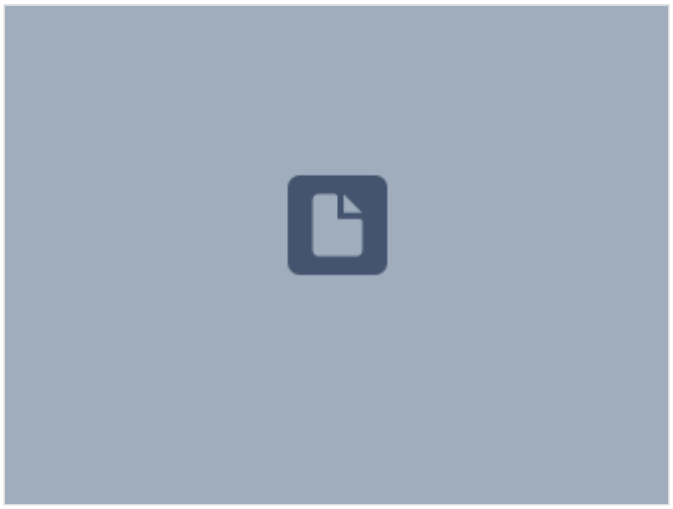
18.10.3 Protocol

- The official overview page which will be automatically updated regularly
  E3 1.2 - Fuzzing List[551]
- Feature requests / Feedback for the page that we would like to implement:
  - [ ] Add the team name to the page
  - [ ] Add tabs similiar to ICP-14 Unit Test Sanitizer Report[552] for develop branch, CL3, CL6, CL8 etc.
  - [ ] Separate the page into different ARTs e.g. ICP and BSrv
  - [ ] Add the repository name like a clickable name such as e3_comp_otam_common
  - [ ] Status of the fuzz tests
    - We generate a fuzzing report with following information:
      - List of fuzz targets Unknown User (e7hqlnm)
        - This comes from the doxygen report
      - Coverage information Muckenschnabl, David Jose (EXTERN: ITK)
      - Running duration and #inputs
      - Findings Unknown User (e7hqlnm)
      - Status of run Muckenschnabl, David Jose (EXTERN: ITK)
        - TODO: Not started nor created yet, meaning no report for the target compared to the e3fuzz.xml
        - ERROR: Anything related to build or run error
        - SUCCESSFUL: Fuzz tests ran successfully and no finding/bug was identified
        - FINDINGS: Bugs identified

551 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/3041828882/E3+1.2+-+Fuzzing+List
552 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1237654038/ICP-14+Unit+Test+Sanitizer+Report

- [ ] Jira bug reporting upon bug found
  - Datendeuter will generate a bug ticket based on the information we provide in the report in TARBALL
  - List of Jira Bug fields that need to be filled can be found here: Fuzzing Pipeline incl. JIRA bug reporting with Jenkins (see page 361)
- [ ] (low prio) Add relative paths to the function to fuzz

18.10.4 Links

- Sneak-peek page:
  - [!SNEAKPEEK!] Fuzzing Function Overview Team Update Core (OTA-Master)[553]
  - https://devstack.vwgroup.com/confluence/display/E3SPVMAN/%5B%21DEV%21%5D+Fuzzing+List
- Jira Tickets
  - VWOS-12019[554] - [Template] General hardening via CodeMarking WON'T FIX
  - VWOS-12020[555] - Define means to identify fuzzable functions DONE
  - 
  - E3-134268[556] - [Data lake KPI] Create report for fuzzing and fuzzed compoents DONE

18.10.5 TODO

- [ ] Security team will start writing the fuzzing report to upload as part of the TARBALL in artefactory
- [ ] Datendeuter will start with the feature requests list above

553 https://devstack.vwgroup.com/confluence/spaces/E3SPVMAN/pages/2660667242/SNEAKPEEK+Fuzzing+Function+Overview+Team+Update+Core+OTA-Master
554 https://devstack.vwgroup.com/jira/browse/VWOS-12019?src=confmacro
555 https://devstack.vwgroup.com/jira/browse/VWOS-12020?src=confmacro
556 https://devstack.vwgroup.com/jira/browse/E3-134268?src=confmacro

18.11 2024-09-05 Sync Fuzzing Pipeline with Jenkins Part 5 - Pipeline and Building Block Status

- [x] Dang, Quang Anh (EXTERN: ITK)
- [ ] Baumung, Wjatscheslav (EXTERN: MHP)
- [x] Willers, Martin (T5-PG)
- [x] Unknown User (e7hqlnm)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [x] Smaoui, Yacine (TZ-CC)

Agenda:

- (CICD Team) Status update on Triton Pipeline implementation
- (Security Team) Status update on BB implementation
- (Both) Clarification of open questions:
  - Where to store cifuzz conan package?
    => e3-e3sp-tools-snapshot: https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-tools-snapshot
  - Where to store corpus and how to deal with overwrite corpus? Which access right does the official jenkin user have?
    Yacine said:

    Where to store corpus?

    ```
    https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-components-snapshot
    Something like <that-repo>/<fuzzing-corpus-subfolder-for-all>/<here some structure to differentiate between the components>
    ```

    => Martin will create a ticket to open a new repo for fuzzing corpus
  - Consesus on result of fuzzing format to give to Datendeuter and where to put this fuzzing_result.json file in the tarball?
    fuzzing_result.json[557]
  - We need e3fuzz.xml for our 2nd Building Block to evaluate the fuzzing result => How to best get this in the pipeline, since a parallel one would be generate doxygen xml files anyway?
    TODO: Martin?

557 https://devstack.vwgroup.com/confluence/download/attachments/3791160210/fuzzing_result.json?api=v2&modificationDate=1725458456897&version=1

- Lack of access for the Example Jenkin Pipeline to test the building block: https://jenkins.ioi.imfa.hub.vwgroup.com/job/BUILDING_BLOCKS/job/tools-bb-fuzzing-src/
  - The technical user can't even clone a repo
- Ask CICD team to add option to allow team to execute fuzzing on other branches as well on the CICD pipeline
  TODO: Wjatschev Baumung?
- llvm-16 or higher to avoid segmentation fault
  TODO: Julius?

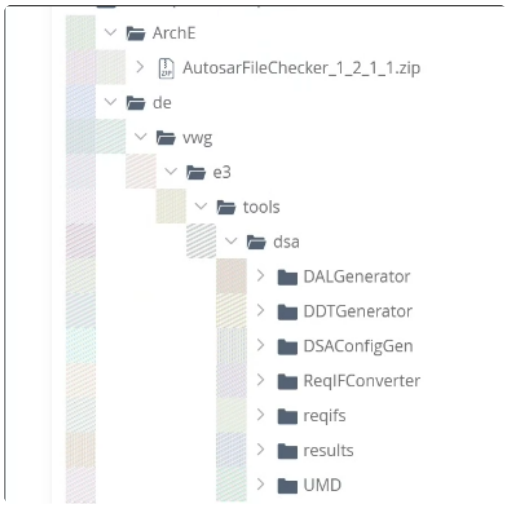
Protocol:

- 2 weeks until end of PI. After that, the developer who is implementing it won't be there any more
- Build container: **master/e3sdk-build-bp1:4.5.0**
- Pipeline can build remote repair
  - normal build?
  - coverage?
  - Pipeline must build all the dependencies with fuzzing profile

```
{
    "fileName": "fuzzing-x86_64",
    "fileContent": """
        [env]
        CC=/usr/bin/clang-15
        CXX=/usr/bin/clang++-15
        LD=/usr/bin/ld
        AR=/usr/bin/ar

        # The cmake_find_package generator does not properly handle compiler.libcxx, so we have to add
        it manually:
        CXXFLAGS="-stdlib=libc++"

        [settings]
        # Host settings
        os_build=Linux
        arch_build=x86_64

        # Target settings
        os=Linux
        os.target=local
        arch=x86_64

        compiler=clang
        compiler.version=15
        compiler.libcxx=libc++

        build_type=Debug
    """
```

- Test jenkin pipeline: https://jenkins.ioi.imfa.hub.vwgroup.com/job/BUILDING_BLOCKS/job/tools-bb-fuzzing-src/
  We have to adjust the credentials. The current technical user doesn't have access to Bitbucket
  - Test jenkin is in a docker container already => adjust the Jenkinfile to use the local user
  - TODO: ticket für support required . Willers, Martin (T5-PG) will create a request ticket
  - For now, we can use `https://jfrog.hub.vwgroup.com/artifactory/vwos-tools-generic-playground/fuzzing_corpus_storage_test`

for testing

- Where to store corpus?
  - Fuzzing should has its own remote
- Regarding clang-18 and llvm-18, we have to create a new branch and pull request
  TODO: Dang, Quang Anh (EXTERN: ITK) Muckenschnabl, David Jose (EXTERN: ITK) Unknown User (e7hqlnm)
- Conan package for cifuzz tooling:
  - Martin: Do we have any tool in conan package form?
    In VW.os currently, no conan package in that form is used
  - Martin: e3-sp-tools could be suitable
  - To create the conan package
  - e3-e3sp-tools-snapshot: https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-tools-snapshot
  - 

- Doxygen generation
  - doxygen <path to doxy config>
    e.g. doxygen build/doc/

TODOs:

- [ ] How to use custom unsupported docker container?
- [ ] Discuss when the CIFUZZ pipeline should be triggered
- [ ] Dang, Quang Anh (EXTERN: ITK) Update building block to the next version which expects to be inside the docker container already

Helpful Links

Proof Of Concept Fuzzing All in One documentation page:
CI-Fuzz Proof of Concept (see page 393)

Example for Jenkinsfile:

https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3_aux_kpi_lake/browse/Jenkinsfile

Jenkins Shared library:

https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_pipeline_generic_shared_lib_premium/browse/src/de/vwg/e3/ioi/premium/nodes/Nodes.groovy

Contact: Smaoui, Yacine (TP-S2) - CICD triton pipeline, jenkins expert

Contact: Baumung, Wjatscheslav (EXTERN: MHP) -

18.12 2024-09-05 Sync Fuzzing Pipeline with Jenkins Part 6 - Pipeline and Building Block Status

- [x] Dang, Quang Anh (EXTERN: ITK)
- [x] Baumung, Wjatscheslav (EXTERN: MHP)
- [x] Willers, Martin (T5-PG)
- [x] Unknown User (e7hqlnm)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [x] Smaoui, Yacine (TZ-CC)
- [ ] Stefan Slosarek

Agenda:

- (CICD Team) Status update on Triton Pipeline implementation
- (Security Team) Status update on BB implementation
- (Both) Clarification of open questions:
  - Where to store cifuzz conan package?
    => e3-e3sp-tools-snapshot: https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-tools-snapshot
  - Where to store corpus and how to deal with overwrite corpus? Which access right does the official jenkin user have?
    Yacine said:

Where to store corpus?
`https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-components-snapshot` `Something like <that-repo>/<fuzzing-corpus-subfolder-for-all>/<here some structure to differentiate between the components>`

=> Martin will create a ticket to open a new repo for fuzzing corpus

- Consesus on result of fuzzing format to give to Datendeuter and where to put this fuzzing_result.json file in the tarball?
  fuzzing_result.json[558]
- We need e3fuzz.xml for our 2nd Building Block to evaluate the fuzzing result => How to best get this in the pipeline, since a parallel one would be generate doxygen xml files anyway?
  TODO: Martin?
- Lack of access for the Example Jenkin Pipeline to test the building block: https://jenkins.ioi.imfa.hub.vwgroup.com/job/BUILDING_BLOCKS/job/tools-bb-fuzzing-src/
  - The technical user can't even clone a repo
- Ask CICD team to add option to allow team to execute fuzzing on other branches as well on the CICD pipeline
  TODO: Wjatschev Baumung?
- llvm-16 or higher to avoid segmentation fault
  TODO: Julius?

Protocol:

11 Sep 2024

- Baumung, Wjatscheslav (EXTERN: MHP) Build with fuzzing fails on the pipeline
  - Baumung, Wjatscheslav (EXTERN: MHP) will focus on building dependencies with fuzzing profile first
- Willers, Martin (T5-PG) requested repo for the corpus. This is ongoing.
- Smaoui, Yacine (TZ-CC) Techuser has the right to overwrite in snapshot e3-e3sp-tools-snapshot: https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-tools-snapshot
  - Issue with the python version in the docker container. Python3.10 is currently required.
  - Idea: Reduce the version required by the building block to 3.8
  - TODO: Unknown User (e7hqlnm) will reduce the version of python to 3.8
- Unknown User (e7hqlnm) and Dang, Quang Anh (EXTERN: ITK) will try fix the building block by today

03 Sep 2024

- 2 weeks until end of PI. After that, the developer who is implementing it won't be there any more
- Build container: **master/e3sdk-build-bp1:4.5.0**
- Pipeline can build remote repair
  - normal build?

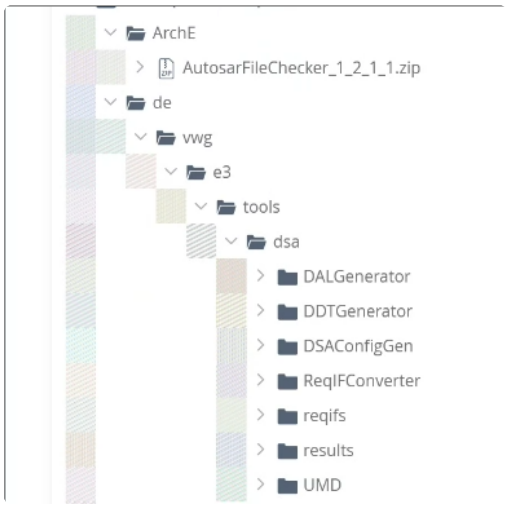
558 https://devstack.vwgroup.com/confluence/download/attachments/3828070660/fuzzing_result.json?api=v2&modificationDate=1726052551578&version=1

  - coverage?
  - Pipeline must build all the dependencies with fuzzing profile

```
{
    "fileName": "fuzzing-x86_64",
    "fileContent": """
        [env]
        CC=/usr/bin/clang-15
        CXX=/usr/bin/clang++-15
        LD=/usr/bin/ld
        AR=/usr/bin/ar

        # The cmake_find_package generator does not properly handle compiler.libcxx, so we have to add
        it manually:
        CXXFLAGS="-stdlib=libc++"

        [settings]
        # Host settings
        os_build=Linux
        arch_build=x86_64

        # Target settings
        os=Linux
        os.target=local
        arch=x86_64

        compiler=clang
        compiler.version=15
        compiler.libcxx=libc++

        build_type=Debug
    """
```

- Test jenkin pipeline: https://jenkins.ioi.imfa.hub.vwgroup.com/job/BUILDING_BLOCKS/job/tools-bb-fuzzing-src/
  We have to adjust the credentials. The current technical user doesn't have access to Bitbucket
  - Test jenkin is in a docker container already => adjust the Jenkinfile to use the local user
  - TODO: ticket für support required . Willers, Martin (T5-PG) will create a request ticket
  - For now, we can use `https://jfrog.hub.vwgroup.com/artifactory/vwos-tools-generic-playground/fuzzing_corpus_storage_test`

    for testing
- Where to store corpus?
  - Fuzzing should has its own remote
- Regarding clang-18 and llvm-18, we have to create a new branch and pull request
  TODO: Dang, Quang Anh (EXTERN: ITK) Muckenschnabl, David Jose (EXTERN: ITK) Unknown User (e7hqlnm)
- Conan package for cifuzz tooling:
  - Martin: Do we have any tool in conan package form?
    In VW.os currently, no conan package in that form is used
  - Martin: e3-sp-tools could be suitable
  - To create the conan package
  - e3-e3sp-tools-snapshot: https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-tools-snapshot

- 

- Doxygen generation
  - doxygen <path to doxy config>
    e.g. doxygen build/doc/

TODOs:

- [ ] Baumung, Wjatscheslav (EXTERN: MHP) will finish the building with fuzzing profile and build the remote repair repo

Helpful Links

Proof Of Concept Fuzzing All in One documentation page:
CI-Fuzz Proof of Concept (see page 393)

Example for Jenkinsfile:

https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3_aux_kpi_lake/browse/Jenkinsfile

Jenkins Shared library:

https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_pipeline_generic_shared_lib_premium/browse/src/de/vwg/e3/ioi/premium/nodes/Nodes.groovy

Contact: Smaoui, Yacine (TP-S2) - CICD triton pipeline, jenkins expert

Contact: Baumung, Wjatscheslav (EXTERN: MHP) -

18.13 2024-09-16 Sync Fuzzing Pipeline with Jenkins Part 7 - Pipeline and Building Block Status

- [ ] Dang, Quang Anh (EXTERN: ITK)

- [x] Baumung, Wjatscheslav (EXTERN: MHP)
- [x] Willers, Martin (T5-PG)
- [x] Unknown User (e7hqlnm)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [x] Smaoui, Yacine (TZ-CC)
- [x] Stefan Slosarek

Agenda:

- (CICD Team) Status update on Triton Pipeline implementation
- (Security Team) Status update on BB implementation
- (Both) Clarification of open questions:
  - Where to store cifuzz conan package?
    => e3-e3sp-tools-snapshot: https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-tools-snapshot
  - Where to store corpus and how to deal with overwrite corpus? Which access right does the official jenkin user have?
    Yacine said:

    Where to store corpus?

    ```
    https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-components-snapshot
    Something like <that-repo>/<fuzzing-corpus-subfolder-for-all>/<here some structure to differentiate between the components>
    ```

    => Martin will create a ticket to open a new repo for fuzzing corpus
  - Consesus on result of fuzzing format to give to Datendeuter and where to put this fuzzing_result.json file in the tarball?
    fuzzing_result.json[559]
  - We need e3fuzz.xml for our 2nd Building Block to evaluate the fuzzing result => How to best get this in the pipeline, since a parallel one would be generate doxygen xml files anyway?
    TODO: Martin?
  - Lack of access for the Example Jenkin Pipeline to test the building block: https://jenkins.ioi.imfa.hub.vwgroup.com/job/BUILDING_BLOCKS/job/tools-bb-fuzzing-src/
    - The technical user can't even clone a repo

559 https://devstack.vwgroup.com/confluence/download/attachments/3852785891/fuzzing_result.json?api=v2&modificationDate=1726472678822&version=1

    - Ask CICD team to add option to allow team to execute fuzzing on other branches as well on the CICD pipeline
      TODO: Wjatschev Baumung?
    - llvm-16 or higher to avoid segmentation fault
      TODO: Julius?

News:

- New profile using corrected compiler flags and mostly works. Issue with IPC common lib. Culprit found Bable tools
  - Team Comet has to change config of Bable tools
  - Muckenschnabl, David Jose (EXTERN: ITK) tell Bernd and Daniel to escalate ticket
- Yacine has managed to modify the commonpart where we can use pip for the Building Block
- Radu has a feature branch where the building block is working better
  - BB is missing the binary of cifuzz
  - Radu is working on making it work on his local machine for testing the pipeline
  - Unknown User (e7hqlnm) will contact Baumung, Wjatscheslav (EXTERN: MHP) when he has
- Artifactory space for the corpus
  - https://jfrog.hub.vwgroup.com/ui/repos/tree/General/vwos-tools-generic-internal
    - we have to define the internal structure of this.
    - Unknown User (e7hqlnm) will test if the technichal user can use this jfrog correctly
      - if something does not work contact Yacine
- Building block python 3.10
  - Smaoui, Yacine (TZ-CC) will ask Afrizal
  - The pip version in bp1 is too old, Yacine is working on fixing this

Protocol:

11 Sep 2024

- Baumung, Wjatscheslav (EXTERN: MHP) Build with fuzzing fails on the pipeline
  - Baumung, Wjatscheslav (EXTERN: MHP) will focus on building dependencies with fuzzing profile first
- Willers, Martin (T5-PG) requested repo for the corpus. This is ongoing.
- Smaoui, Yacine (TZ-CC) Techuser has the right to overwrite in snapshot e3-e3sp-tools-snapshot: https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-tools-snapshot
  - Issue with the python version in the docker container. Python3.10 is currently required.
  - Idea: Reduce the version required by the building block to 3.8
  - TODO: Unknown User (e7hqlnm) will reduce the version of python to 3.8

- Unknown User (e7hqlnm) and Dang, Quang Anh (EXTERN: ITK) will try fix the building block by today

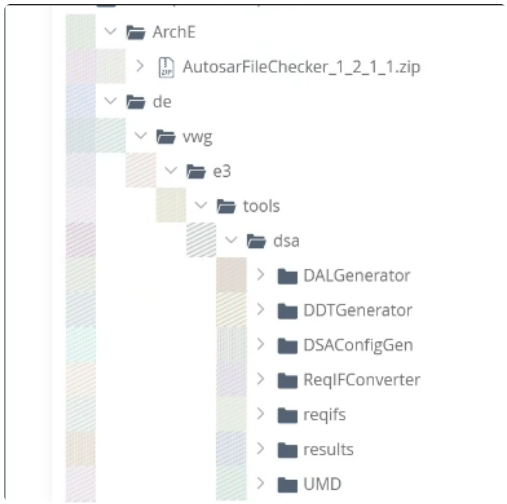
03 Sep 2024

- 2 weeks until end of PI. After that, the developer who is implementing it won't be there any more
- Build container: **master/e3sdk-build-bp1:4.5.0**
- Pipeline can build remote repair
  - normal build?
  - coverage?
  - Pipeline must build all the dependencies with fuzzing profile

```
{
    "fileName": "fuzzing-x86_64",
    "fileContent": """
        [env]
        CC=/usr/bin/clang-15
        CXX=/usr/bin/clang++-15
        LD=/usr/bin/ld
        AR=/usr/bin/ar

        # The cmake_find_package generator does not properly handle compiler.libcxx, so we have to add
        it manually:
        CXXFLAGS="-stdlib=libc++"

        [settings]
        # Host settings
        os_build=Linux
        arch_build=x86_64

        # Target settings
        os=Linux
        os.target=local
        arch=x86_64

        compiler=clang
        compiler.version=15
        compiler.libcxx=libc++

        build_type=Debug
    """
```

- Test jenkin pipeline: https://jenkins.ioi.imfa.hub.vwgroup.com/job/BUILDING_BLOCKS/job/tools-bb-fuzzing-src/
  We have to adjust the credentials. The current technical user doesn't have access to Bitbucket
  - Test jenkin is in a docker container already => adjust the Jenkinfile to use the local user
  - TODO: ticket für support required . Willers, Martin (T5-PG) will create a request ticket
  - For now, we can use `https://jfrog.hub.vwgroup.com/artifactory/vwos-tools-generic-playground/fuzzing_corpus_storage_test`

    for testing
- Where to store corpus?
  - Fuzzing should has its own remote
- Regarding clang-18 and llvm-18, we have to create a new branch and pull request
  TODO: Dang, Quang Anh (EXTERN: ITK) Muckenschnabl, David Jose (EXTERN: ITK) Unknown User (e7hqlnm)

- Conan package for cifuzz tooling:
  - Martin: Do we have any tool in conan package form?
    In VW.os currently, no conan package in that form is used
  - Martin: e3-sp-tools could be suitable
  - To create the conan package
  - e3-e3sp-tools-snapshot: https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-tools-snapshot
  - 

- Doxygen generation
  - doxygen <path to doxy config>
    e.g. doxygen build/doc/

TODOs:

Helpful Links

Proof Of Concept Fuzzing All in One documentation page:
CI-Fuzz Proof of Concept (see page 393)

Example for Jenkinsfile:

https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3_aux_kpi_lake/browse/Jenkinsfile

Jenkins Shared library:

https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_pipeline_generic_shared_lib_premium/browse/src/de/vwg/e3/ioi/premium/nodes/Nodes.groovy

Contact: Smaoui, Yacine (TP-S2) - CICD triton pipeline, jenkins expert

Contact: Baumung, Wjatscheslav (EXTERN: MHP) -

18.14 2024-09-20 Sync Fuzzing Pipeline with Jenkins Part 8 - Pipeline and Building Block Status

- [ ] Dang, Quang Anh (EXTERN: ITK)
- [x] Baumung, Wjatscheslav (EXTERN: MHP)
- [x] Willers, Martin (T5-PG)
- [ ] Unknown User (e7hqlnm)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [x] Smaoui, Yacine (TZ-CC)
- [x] Stefan Slosarek

Agenda:

- (CICD Team) Status update on Triton Pipeline implementation
- (Security Team) Status update on BB implementation
- (Both) Clarification of open questions:
  - Where to store cifuzz conan package?
    => e3-e3sp-tools-snapshot: https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-tools-snapshot
  - Where to store corpus and how to deal with overwrite corpus? Which access right does the official jenkin user have?
    Yacine said:

    Where to store corpus?

    ```
    https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-components-snapshot
    Something like <that-repo>/<fuzzing-corpus-subfolder-for-all>/<here some structure to differentiate between the components>
    ```

    => Martin will create a ticket to open a new repo for fuzzing corpus
  - Consesus on result of fuzzing format to give to Datendeuter and where to put this fuzzing_result.json file in the tarball?
    fuzzing_result.json[560]
  - We need e3fuzz.xml for our 2nd Building Block to evaluate the fuzzing result => How to best get this in the pipeline, since a parallel one would be generate doxygen xml files anyway?
    TODO: Martin?

560 https://devstack.vwgroup.com/confluence/download/attachments/3886682011/fuzzing_result.json?api=v2&modificationDate=1726822843042&version=1

- Lack of access for the Example Jenkin Pipeline to test the building block: https://jenkins.ioi.imfa.hub.vwgroup.com/job/BUILDING_BLOCKS/job/tools-bb-fuzzing-src/
  - The technical user can't even clone a repo
- Ask CICD team to add option to allow team to execute fuzzing on other branches as well on the CICD pipeline
  TODO: Wjatschev Baumung?
- llvm-16 or higher to avoid segmentation fault
  TODO: Julius?

News:

- TeamComet's ticket will be fixed next PI
- A workaround was found, clang15 build was achieved
- Slavik will try the building block out on monday

Protocol:

11 Sep 2024

- Baumung, Wjatscheslav (EXTERN: MHP) Build with fuzzing fails on the pipeline
  - Baumung, Wjatscheslav (EXTERN: MHP) will focus on building dependencies with fuzzing profile first
- Willers, Martin (T5-PG) requested repo for the corpus. This is ongoing.
- Smaoui, Yacine (TZ-CC) Techuser has the right to overwrite in snapshot e3-e3sp-tools-snapshot: https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-tools-snapshot
  - Issue with the python version in the docker container. Python3.10 is currently required.
  - Idea: Reduce the version required by the building block to 3.8
  - TODO: Unknown User (e7hqlnm) will reduce the version of python to 3.8
- Unknown User (e7hqlnm) and Dang, Quang Anh (EXTERN: ITK) will try fix the building block by today

03 Sep 2024

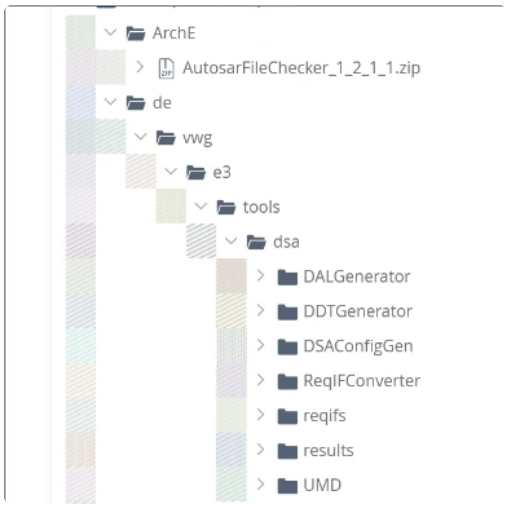
- 2 weeks until end of PI. After that, the developer who is implementing it won't be there any more
- Build container: **master/e3sdk-build-bp1:4.5.0**
- Pipeline can build remote repair
  - normal build?
  - coverage?

- Pipeline must build all the dependencies with fuzzing profile

```
{
    "fileName": "fuzzing-x86_64",
    "fileContent": """
        [env]
        CC=/usr/bin/clang-15
        CXX=/usr/bin/clang++-15
        LD=/usr/bin/ld
        AR=/usr/bin/ar

        # The cmake_find_package generator does not properly handle compiler.libcxx, so we have to add
        it manually:
        CXXFLAGS="-stdlib=libc++"

        [settings]
        # Host settings
        os_build=Linux
        arch_build=x86_64

        # Target settings
        os=Linux
        os.target=local
        arch=x86_64

        compiler=clang
        compiler.version=15
        compiler.libcxx=libc++

        build_type=Debug
    """
```

- Test jenkin pipeline: https://jenkins.ioi.imfa.hub.vwgroup.com/job/BUILDING_BLOCKS/job/tools-bb-fuzzing-src/
  We have to adjust the credentials. The current technical user doesn't have access to Bitbucket
  - Test jenkin is in a docker container already => adjust the Jenkinfile to use the local user
  - TODO: ticket für support required . Willers, Martin (T5-PG) will create a request ticket
  - For now, we can use `https://jfrog.hub.vwgroup.com/artifactory/vwos-tools-generic-playground/fuzzing_corpus_storage_test`

    for testing
- Where to store corpus?
  - Fuzzing should has its own remote
- Regarding clang-18 and llvm-18, we have to create a new branch and pull request
  TODO: Dang, Quang Anh (EXTERN: ITK) Muckenschnabl, David Jose (EXTERN: ITK) Unknown User (e7hqlnm)
- Conan package for cifuzz tooling:
  - Martin: Do we have any tool in conan package form?
    In VW.os currently, no conan package in that form is used
  - Martin: e3-sp-tools could be suitable
  - To create the conan package
  - e3-e3sp-tools-snapshot: https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-tools-snapshot

- 

- Doxygen generation
  - doxygen <path to doxy config>
    e.g. doxygen build/doc/

TODOs:

Helpful Links

Proof Of Concept Fuzzing All in One documentation page:
CI-Fuzz Proof of Concept (see page 393)

Example for Jenkinsfile:

https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3_aux_kpi_lake/browse/Jenkinsfile

Jenkins Shared library:

https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_pipeline_generic_shared_lib_premium/browse/src/de/vwg/e3/ioi/premium/nodes/Nodes.groovy

Contact: Smaoui, Yacine (TP-S2) - CICD triton pipeline, jenkins expert

Contact: Baumung, Wjatscheslav (EXTERN: MHP) -

18.15 2024-09-25 Sync Fuzzing Pipeline with Jenkins Part 9 - Pipeline and Building Block Status

- [ ] Dang, Quang Anh (EXTERN: ITK)
- [x] Baumung, Wjatscheslav (EXTERN: MHP)
- [x] Willers, Martin (T5-PG)

- [x] Unknown User (e7hqlnm)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [x] Smaoui, Yacine (TZ-CC)
- [x] Stefan Slosarek

18.15.1 Agenda:

- @Slavik: continue with BB integration
- @David: sketch an example JSON output file for consumption by the Data Lakers
- @Radu: solve the race conditions when downloading/uploading the corpus

18.15.2 Decisions so far:

- @Martin's suggestion so far is to run fuzzing only on master by default
- TODO: discuss this with QA
- The output of the "master" fuzzing runs should be presented in the 100% report in the usual way.
- TODO: exact output yet TBD
- TODO: how to link that with the @e3fuzzing-generated pages? https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=3061303501
- As teams have requested to run fuzzing on feature branches as well, that should be made possible as an optional quality gate.
- The output of the feature-branch fuzzing runs should be shown in the job result page.
- Let's run the entire fuzzing step for a fixed 10 minutes for now.
  This will likely change in later revisions, maybe based on coverage data (e.g. run until coverage data has increased by some amount), maybe based on corpus data (e.g. run while corpus still sees "interesting" changes), maybe based on component size/complexity (e.g. consider LoC and other source code metrics), etc.
  There should always also be an absolute timeout, as we must prevent endless fuzzing.
- The second fuzzing BB collects and reports coverage data, and for that we need the Doxygen XML output (where the @e3fuzzing data is available).
  Hence, the BB needs to run Doxygen by itself again.
- The "Doxyfile" config file is being patched by the pipeline with probably at least these changes:
  -GENERATE_HTML = YES
  -GENERATE_XML = NO
  +GENERATE_HTML = NO
  +GENERATE_HTML = YES
- We should look at other settings to patch that could make the doxygen run finish more quickly.

Helpful Links

Proof Of Concept Fuzzing All in One documentation page:
CI-Fuzz Proof of Concept (see page 393)

Example for Jenkinsfile:

https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3_aux_kpi_lake/browse/Jenkinsfile

Jenkins Shared library:

https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_pipeline_generic_shared_lib_premium/browse/src/de/vwg/e3/ioi/premium/nodes/Nodes.groovy

Contact: Smaoui, Yacine (TP-S2) - CICD triton pipeline, jenkins expert

Contact: Baumung, Wjatscheslav (EXTERN: MHP) -

18.16 2024-10-02 Sync Fuzzing Pipeline with Jenkins Part 10 - Pipeline and Building Block Status

- [ ] Dang, Quang Anh (EXTERN: ITK)
- [x] Baumung, Wjatscheslav (EXTERN: MHP)
- [x] Willers, Martin (T5-PG)
- [x] Unknown User (e7hqlnm)
- [x] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Smaoui, Yacine (TZ-CC)
- [x] Stefan Slosarek

18.16.1 Agenda

- @Slavik: build all with Clang profile à release next week

- @Slavik: continue with BB integration à still on feature branch for now

- @David: sketch an example JSON output file for consumption by the Data Lakers à send as RFC to Maik/DataLakers

- @Radu: solve the race conditions when downloading/uploading the corpus

  à i.e. some data loss is ok for now (e.g. latest additions to the corpus are lost), but data corruption is NOT ok

    (e.g. upload/download yields invalid corpus)

- @Radu: release first working version of the 2. BB (e.g. simple output of results to console)

- @Martin: discuss fuzzing workmode with QA

18.16.2 Decisions so far

- Inter-stage signalling postponed
  - how to stop fuzzing run prematurely? à an optimization only, postpone for now
  - how to trigger stop by other pipeline stages? à an optimization only, postpone for now
- @Martin's suggestion so far is to run fuzzing only on master by default

TODO: discuss this with QA

- The output of the "master" fuzzing runs should be presented in the 100% report in the usual way.

TODO: exact output yet TBD

TODO: how to link that with the @e3fuzzing-generated pages? https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=3061303501

- As teams have requested to run fuzzing on feature branches as well,

that should be made possible as an optional quality gate.

- The output of the feature-branch fuzzing runs should be shown in the job result page.
- Let's run the entire fuzzing step for a fixed 10 minutes for now.

This will likely change in later revisions, maybe based on coverage data (e.g. run until coverage

data has increased by some amount), maybe based on corpus data (e.g. run while corpus

still sees "interesting" changes), maybe based on component size/complexity (e.g. consider LoC

and other source code metrics), etc.

There should always also be an absolute timeout, as we must prevent endless fuzzing.

- The second fuzzing BB collects and reports coverage data, and for that we need the

Doxygen XML output (where the @e3fuzzing data is available).

Hence, the BB needs to run Doxygen by itself again.

- The "Doxyfile" config file is being patched by the pipeline with probably at least these changes:

-GENERATE_HTML = YES

-GENERATE_XML = NO

+GENERATE_HTML = NO

+GENERATE_XML = YES

We should look at other settings to patch that could make the doxygen run finish more quickly,

e.g. omit the generation of UML diagrams etc.

18.16.3 TODO

- [x] Muckenschnabl, David Jose (EXTERN: ITK) send a mail to Maik Engelhart with json example

Helpful Links

Proof Of Concept Fuzzing All in One documentation page:
CI-Fuzz Proof of Concept (see page 393)

Example for Jenkinsfile:

https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3_aux_kpi_lake/browse/Jenkinsfile

Jenkins Shared library:

https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_pipeline_generic_shared_lib_premium/browse/src/de/vwg/e3/ioi/premium/nodes/Nodes.groovy

Contact: Smaoui, Yacine (TP-S2) - CICD triton pipeline, jenkins expert

Contact: Baumung, Wjatscheslav (EXTERN: MHP) -

18.17 2024-10-16 Sync Fuzzing Pipeline with Jenkins Part 11 - Pipeline and Building Block Status

- [x] Dang, Quang Anh (EXTERN: ITK)
- [x] Baumung, Wjatscheslav (EXTERN: MHP)
- [x] Willers, Martin (T5-PG)
- [x] Unknown User (e7hqlnm)
- [ ] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Smaoui, Yacine (TZ-CC)
- [x] Stefan Slosarek
- [x] Burkhard, Dietterle

18.17.1 Agenda

- @Slavik: status of Clang profile build
- @Slavik: status of BB integration
- @Quang & Radu: status of both BB

- @Quang: the ci_coverage_calculator-0.3.0-py3-none-any.whl tool for coverage calculation. Where to include them?
  - In docker container?
  - In Conan package
  - Install during build by CICD pipeline
- @Martin: discuss fuzzing workmode with QA
- @Martin: Decision on running fuzzing on which branch? Suggestion is to run it on develop branch (cluster6) and feature branch
- Future features:
  - Coverage evaluation before running the fuzz target. Skipping the targets with > 80 % code coverage
  - @Martin: Stopping fuzz run prematurely or trigger stop by other pipeline stages.
    TODO: discuss with QA
  - @Martin: Output of fuzz run to be presented in 100% report.
    TODO: how to link that with the @e3fuzzing-generated pages? https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=3061303501
    TODO: discuss once the output is set by the 2nd BB and visualized by DataLakers
  - @Martin: Teams request to be able to run fuzz tests on feature branches. That should be made possible as an optional quality gate.
    The output of the feature-branch fuzzing runs should be shown in the job result page.
  - Discuss nightly run / weekend fuzz run in the future
  - Doxygen generation before calling 2nd BB. Status and how to speed this up with modified settings e.g. omit the generation of UML diagrams. Takes 20s at most. Currently using this settings:

    -GENERATE_HTML = YES

    -GENERATE_XML = NO

    +GENERATE_HTML = NO

    +GENERATE_XML = YES

18.17.2 Meeting minutes

- **@Slavic:** No change regarding pipeline building block
- Meeting with DataLakers to discuss Result JSON will take place 16 Oct 2024 with Metrics Board from DataLakers
- Unknown User (e7hqlnm) already fixed the race condition for uploading/downloading the corpus in BB-1
- BB-1 has undergone some changes for bug fix
  https://git.hub.vwgroup.com/CARIAD/tools-bb-fuzzing-src/pull/6

- BB-2 requires ci_coverage_calculator-0.3.0-py3-none-any.whl tool to calculate coverage to be installed on the pipeline
  Building block is meant to be shared between different pipelines. External dependencies are not good
  - Slosarek, Stefan: Not to include this into docker. Now, store it into the artefactory for now.
- For now the CICD pipeline only works for Remote Repair for now
  - Problem with team comet (babel tool) to be fixed. Babeltool currently has issue with CLANG-15 profile
    System Build 1727250996415[561]
  - TODO: Build dependencies from bottom up for triton build
- Stopping fuzz run prematurely or trigger stop by other pipeline stages.
  - Triton Pipeline in it architecture can not deactivate the build itself and only run fuzz run
- Currently, system build with fuzzing is run manually
  - Idea for the future: Daily, Weekly, etc.
  - System build status: https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_e3sp_system_build_configs/browse/cron.yaml
- System Build vs Triton Build:
  - System Build is triggered periodically at a specific time or manually
  - Triton Pipeline is triggered via push/PR etc.
- Doxygen Generation currently uses following settings:

  GENERATE_HTML = YES

  -GENERATE_XML = NO

  +GENERATE_HTML = NO

  +GENERATE_XML = YES
- BB1: https://git.hub.vwgroup.com/CARIAD/tools-bb-fuzzing-report-src

  BB2: https://git.hub.vwgroup.com/CARIAD/tools-bb-fuzzing-report-src/tree/feature/implement-basic-reporting

18.17.3 TODO

- [x] Muckenschnabl, David Jose (EXTERN: ITK) send a mail to Maik Engelhart with json example

Helpful Links

Proof Of Concept Fuzzing All in One documentation page:
CI-Fuzz Proof of Concept (see page 393)

Example for Jenkinsfile:

561 https://devstack.vwgroup.com/confluence/spaces/VWOSCUS/pages/3909956924/System+Build+1727250996415

https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3_aux_kpi_lake/browse/Jenkinsfile

Jenkins Shared library:

https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_pipeline_generic_shared_lib_premium/browse/src/de/vwg/e3/ioi/premium/nodes/Nodes.groovy

Contact: Smaoui, Yacine (TP-S2) - CICD triton pipeline, jenkins expert

Contact: Baumung, Wjatscheslav (EXTERN: MHP) -

18.18 2024-10-22 Sync regarding visualization of fuzzing result from pipeline

18.18.1 Participants

Unknown User (e7hqlnm)

Engelhardt, Maik (T5-PG)

Bartels, Bjoern (EXTERN: Datendeuter)

Kutscher, Max (EXTERN: PTG)

18.18.2 Agenda

- How to visualize fuzzing result?
- Any further information required from the report.json
- Where to store all the outputs e.g
  - findings log incl. stacktrace

18.18.3 Protocol

From last time:

- Feature requests / Feedback for the page that we would like to implement:
  - [x] Add the team name to the page
  - [x] Add tabs similiar to ICP-14 Unit Test Sanitizer Report[562] for develop branch, CL3, CL6, CL8 etc.

562 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/1237654038/ICP-14+Unit+Test+Sanitizer+Report

- [x] Separate the page into different ARTs e.g. ICP and BSrv
- [x] Add the repository name like a clickable name such as e3_comp_otam_common
- [ ] Status of the fuzz tests
  - We generate a fuzzing report with following information:
    - List of fuzz targets Unknown User (e7hqlnm)
      - This comes from the doxygen report
    - Coverage information Muckenschnabl, David Jose (EXTERN: ITK)
    - Running duration and #inputs
    - Findings Unknown User (e7hqlnm)
    - Status of run Muckenschnabl, David Jose (EXTERN: ITK)
      - TODO: Not started nor created yet, meaning no report for the target compared to the e3fuzz.xml
      - ERROR: Anything related to build or run error
      - SUCCESSFUL: Fuzz tests ran successfully and no finding/bug was identified
      - FINDINGS: Bugs identified
- [ ] Jira bug reporting upon bug found
  - Datendeuter will generate a bug ticket based on the information we provide in the report in TARBALL
  - List of Jira Bug fields that need to be filled can be found here: Fuzzing Pipeline incl. JIRA bug reporting with Jenkins (see page 361)
- [ ] (low prio) Add relative paths to the function to fuzz

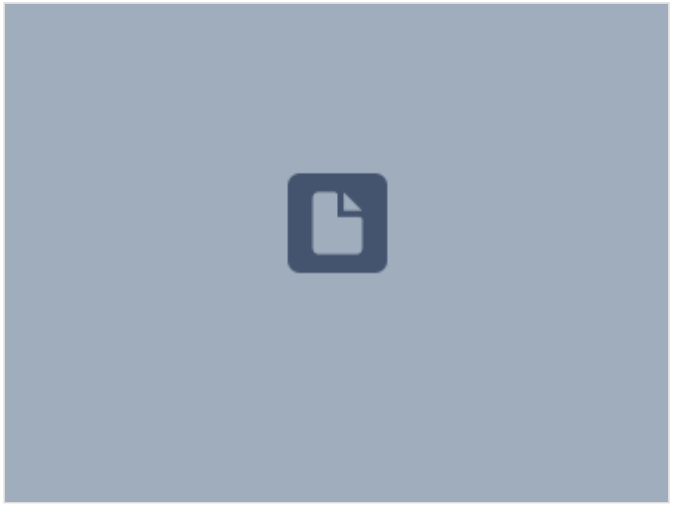
22 Oct 2024

- The official overview page which will be automatically updated regularly
  E3 1.2 - Fuzzing List[563]
- TODO: Directory / Path to store the fuzzing reporting and findings
- How to determine status of a fuzz test?
  - SUCCESSFUL: The function header and signature can be found in the report.json AND findings is empty
  - FINDINGS: The function header and signature can be found
    - [ ] bring the function header incl. parameter signature to the doxygen format to match
  - BUILD ERROR: Datendeuter can extract this information from the build pipeline
  - TODO: Fuzz test for the function is not yet implemented
    => parse report.json, if the function header is not available in "covered"=> then set it to TODO
- Which info is needed for bug reporting?
  - [ ] CVSS score is missing in the report.json

563 https://devstack.vwgroup.com/confluence/spaces/E3SWPAC/pages/3041828882/E3+1.2+-+Fuzzing+List

- TODO:
  - [ ] Put stack trace from finding directly into report.json
  - [ ] Put crashing input from finding firectly into report.json
  - [ ] Put stack trace file and crashing input file into the reporting directory
  - [ ] Bartels, Bjoern (EXTERN: Datendeuter) will create a Jira ticket for this task

18.18.4 Links

- Sneak-peek page:
  - E3 1.2 - Fuzzing List[564]
- Jira Tickets
  - From Security Team
    VWOS-16084[565] - Support DataLakers to finish visualizing fuzzing CICD result DONE
  - From Daten Deuter E3-154410[566] - [Reporting] Reporting of Fuzzing test results DONE
  - (old) PI 24.1 DataLakers Jira Ticket
    E3-134268[567] - [Data lake KPI] Create report for fuzzing and fuzzed compoents DONE
  - 

18.18.5 TODO

- [ ] Security team will start writing the fuzzing report to upload as part of the TARBALL in artefactory
- [ ] Datendeuter will start with the feature requests list above

564 https://devstack.vwgroup.com/confluence/spaces/E3SPVMAN/pages/2660667242/SNEAKPEEK+Fuzzing+Function+Overview+Team+Update+Core+OTA-Master
565 https://devstack.vwgroup.com/jira/browse/VWOS-16084?src=confmacro
566 https://devstack.vwgroup.com/jira/browse/E3-154410?src=confmacro
567 https://devstack.vwgroup.com/jira/browse/E3-134268?src=confmacro

18.19 2024-10-24 Sync Fuzzing Pipeline with Jenkins Part 12 - Pipeline and Building Block Status

- [x] Dang, Quang Anh (EXTERN: ITK)
- [x] Baumung, Wjatscheslav (EXTERN: MHP)
- [x] Willers, Martin (T5-PG)
- [x] Unknown User (e7hqlnm)
- [ ] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Smaoui, Yacine (TZ-CC)
- [ ] Stefan Slosarek
- [x] Burkhard, Dietterle
- [ ] Litke Johann
- [ ] Max Kutscher

18.19.1 Agenda

- @Slavik: status of Clang profile build
- @Slavik: status of BB integration
- @Quang & Radu: status of both BB
- @Quang: the ci_coverage_calculator-0.3.0-py3-none-any.whl tool for coverage calculation. Where to include them?
  - In docker container?
  - In Conan package
  - Install during build by CICD pipeline
- @Martin: discuss fuzzing workmode with QA
- @Martin: Decision on running fuzzing on which branch? Suggestion is to run it on develop branch (cluster6) and feature branch
- Future features:
  - Coverage evaluation before running the fuzz target. Skipping the targets with > 80 % code coverage
  - @Martin: Stopping fuzz run prematurely or trigger stop by other pipeline stages.
    TODO: discuss with QA
  - @Martin: Output of fuzz run to be presented in 100% report.
    TODO: how to link that with the @e3fuzzing-generated pages? https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=3061303501
    TODO: discuss once the output is set by the 2nd BB and visualized by DataLakers
  - @Martin: Teams request to be able to run fuzz tests on feature branches. That should be made possible as an optional quality gate.
    The output of the feature-branch fuzzing runs should be shown in the job result page.

- Discuss nightly run / weekend fuzz run in the future
- Doxygen generation before calling 2nd BB. Status and how to speed this up with modified settings e.g. omit the generation of UML diagrams. Takes 20s at most. Currently using this settings:

  -GENERATE_HTML = YES

  -GENERATE_XML = NO

  +GENERATE_HTML = NO

  +GENERATE_XML = YES

18.19.2 Meeting minutes

24 Oct 2024

Ask about where to put fuzzing result. Datendeuter

- Slavic has progress with the build
- Put the reporting JSON to a directory in build/fuzzing-report
- Building Block 2 - First version is ready
- Installation of the final package of ci_coverage_calculator-0.3.2-py3-none-any.whl
- Team Comet has fixed the issue in the BabelTool. However, they haven't released any package for it yet
  - We need Team Comet to release a new package to fix the build for remote repair
  - Problem:
    - E3-151753[568] - Faulty assumptions in babel_tools DONE
    - Libbabeltool hardcodes libc into their profile => which leads to problem
- Currently, system build is trying to build cluster 6. We should move it to cluster 8
  => Team Comet will have much better chance to merge the fix to cluster 8

16 Oct 2024

- **@Slavic:** No change regarding pipeline building block
- Meeting with DataLakers to discuss Result JSON will take place 16 Oct 2024 with Metrics Board from DataLakers
- Unknown User (e7hqlnm) already fixed the race condition for uploading/downloading the corpus in BB-1
- BB-1 has undergone some changes for bug fix
  https://git.hub.vwgroup.com/CARIAD/tools-bb-fuzzing-src/pull/6

568 https://devstack.vwgroup.com/jira/browse/E3-151753?src=confmacro

- BB-2 requires ci_coverage_calculator-0.3.0-py3-none-any.whl tool to calculate coverage to be installed on the pipeline
  Building block is meant to be shared between different pipelines. External dependencies are not good
  - Slosarek, Stefan: Not to include this into docker. Now, store it into the artefactory for now.
- For now the CICD pipeline only works for Remote Repair for now
  - Problem with team comet (babel tool) to be fixed. Babeltool currently has issue with CLANG-15 profile
    System Build 1727250996415[569]
  - TODO: Build dependencies from bottom up for triton build
- Stopping fuzz run prematurely or trigger stop by other pipeline stages.
  - Triton Pipeline in it architecture can not deactivate the build itself and only run fuzz run
- Currently, system build with fuzzing is run manually
  - Idea for the future: Daily, Weekly, etc.
  - System build status: https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_e3sp_system_build_configs/browse/cron.yaml
- System Build vs Triton Build:
  - System Build is triggered periodically at a specific time or manually
  - Triton Pipeline is triggered via push/PR etc.
- Doxygen Generation currently uses following settings:

  GENERATE_HTML = YES

  -GENERATE_XML = NO

  +GENERATE_HTML = NO

  +GENERATE_XML = YES
- BB1: https://git.hub.vwgroup.com/CARIAD/tools-bb-fuzzing-report-src

  BB2: https://git.hub.vwgroup.com/CARIAD/tools-bb-fuzzing-report-src/tree/feature/implement-basic-reporting

18.19.3 TODO

- [ ] Martin Willers and Slavic => Do the system build for cluster 8 instead so team Comet can patch the issue [x] E3-151753[570] - Faulty assumptions in babel_tools DONE
- [ ] Extend System Build to another team or more teams

Helpful Links

569 https://devstack.vwgroup.com/confluence/spaces/VWOSCUS/pages/3909956924/System+Build+1727250996415

570 https://devstack.vwgroup.com/jira/browse/E3-151753?src=confmacro

Proof Of Concept Fuzzing All in One documentation page:
CI-Fuzz Proof of Concept (see page 393)

Example for Jenkinsfile:

https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3_aux_kpi_lake/browse/Jenkinsfile

Jenkins Shared library:

https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_pipeline_generic_shared_lib_premium/browse/src/de/vwg/e3/ioi/premium/nodes/Nodes.groovy

Contact: Smaoui, Yacine (TP-S2) - CICD triton pipeline, jenkins expert

Contact: Baumung, Wjatscheslav (EXTERN: MHP) -

18.20 2024-11-06 Sync Fuzzing Pipeline with Jenkins Part 13 - Pipeline and Building Block Status

- [x] Dang, Quang Anh (EXTERN: ITK)
- [x] Baumung, Wjatscheslav (EXTERN: MHP)
- [x] Willers, Martin (T5-PG)
- [x] Unknown User (e7hqlnm)
- [ ] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Smaoui, Yacine (TZ-CC)
- [x] Stefan Slosarek
- [x] Burkhard, Dietterle
- [ ] Litke Johann
- [x] Max Kutscher
- [ ] Niklas Henrich (CI)
- [ ] Jacek Trossen (CI)
- [x] Schaller, Bernd (T5-SF)

18.20.1 TODOs

- [ ] TODO: Implement interrupt feature for BB-1 to receive/check for SIGTERM and implement interrupt method
  Talk to Afrizal
- [ ] Code Intelligence will fix the cov-calc tool to not use old lib from pypi

18.20.2 Agenda

- @Slavik: status of Clang profile build
- @Slavik: status of BB integration
- @Quang & Radu: status of both BB
- @Quang: the ci_coverage_calculator-0.3.0-py3-none-any.whl tool for coverage calculation. Where to include them?
  - In docker container?
  - In Conan package
  - Install during build by CICD pipeline
- @Martin: discuss fuzzing workmode with QA
- @Martin: Decision on running fuzzing on which branch? Suggestion is to run it on develop branch (cluster6) and feature branch
- Future features:
  - Coverage evaluation before running the fuzz target. Skipping the targets with > 80 % code coverage
  - @Martin: Stopping fuzz run prematurely or trigger stop by other pipeline stages.
    TODO: discuss with QA
  - @Martin: Output of fuzz run to be presented in 100% report.
    TODO: how to link that with the @e3fuzzing-generated pages? https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=3061303501
    TODO: discuss once the output is set by the 2nd BB and visualized by DataLakers
  - @Martin: Teams request to be able to run fuzz tests on feature branches. That should be made possible as an optional quality gate.
    The output of the feature-branch fuzzing runs should be shown in the job result page.
  - Discuss nightly run / weekend fuzz run in the future
  - Doxygen generation before calling 2nd BB. Status and how to speed this up with modified settings e.g. omit the generation of UML diagrams. Takes 20s at most. Currently using this settings:

    -GENERATE_HTML = YES

    -GENERATE_XML = NO

    +GENERATE_HTML = NO

    +GENERATE_XML = YES

18.20.3 Meeting minutes

06 Nov 2024

- Code Intelligence feature request to stop running fuzz test on the pipeline on demand

    - CICD Team needs an asynchronous way to stop fuzzing process
    - Duration time of the longest stage on Triton Pipeline is not known beforehand
    - TODO: Implement interrupt feature for BB-1
- Option to pass to deactivate unicode output in cifuzz
    - *--plain*
- ci_coverage_calculator is pulling some old package from PyPI => Request CI for using the built-in path lib first
- Team Comet has promised to put the fix to develop branch
- System Build looks good sofar
- No progress on the BB integration sofar
- JSON Report to Data Lakers
    - A first draft example has been handed over to Data Lakers
- Stress test for BB-1 for mutiple parallel builds and upload/download corpus
- @Bi Interrupt() function to the BB interface. Talk to Afrizal about this.

24 Oct 2024

Ask about where to put fuzzing result. Datendeuter

- Slavic has progress with the build
- Put the reporting JSON to a directory in build/fuzzing-report
- Building Block 2 - First version is ready
- Installation of the final package of ci_coverage_calculator-0.3.2-py3-none-any.whl
- Team Comet has fixed the issue in the BabelTool. However, they haven't released any package for it yet
    - We need Team Comet to release a new package to fix the build for remote repair
    - Problem:
        - [x] E3-151753[571] - Faulty assumptions in babel_tools DONE
        - Libbabeltool hardcodes libc into their profile => which leads to problem
- Currently, system build is trying to build cluster 6. We should move it to cluster 8
  => Team Comet will have much better chance to merge the fix to cluster 8

16 Oct 2024

- **@Slavic:** No change regarding pipeline building block
- Meeting with DataLakers to discuss Result JSON will take place 16 Oct 2024 with Metrics Board from DataLakers
- Unknown User (e7hqlnm) already fixed the race condition for uploading/downloading the corpus in BB-1

571 https://devstack.vwgroup.com/jira/browse/E3-151753?src=confmacro

- BB-1 has undergone some changes for bug fix
  https://git.hub.vwgroup.com/CARIAD/tools-bb-fuzzing-src/pull/6
- BB-2 requires ci_coverage_calculator-0.3.0-py3-none-any.whl tool to calculate coverage to be installed on the pipeline
  Building block is meant to be shared between different pipelines. External dependencies are not good
  - Slosarek, Stefan: Not to include this into docker. Now, store it into the artefactory for now.
- For now the CICD pipeline only works for Remote Repair for now
  - Problem with team comet (babel tool) to be fixed. Babeltool currently has issue with CLANG-15 profile
    System Build 1727250996415[572]
  - TODO: Build dependencies from bottom up for triton build
- Stopping fuzz run prematurely or trigger stop by other pipeline stages.
  - Triton Pipeline in it architecture can not deactivate the build itself and only run fuzz run
- Currently, system build with fuzzing is run manually
  - Idea for the future: Daily, Weekly, etc.
  - System build status: https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_e3sp_system_build_configs/browse/cron.yaml
- System Build vs Triton Build:
  - System Build is triggered periodically at a specific time or manually
  - Triton Pipeline is triggered via push/PR etc.
- Doxygen Generation currently uses following settings:

  GENERATE_HTML = YES

  -GENERATE_XML = NO

  +GENERATE_HTML = NO

  +GENERATE_XML = YES
- BB1: https://git.hub.vwgroup.com/CARIAD/tools-bb-fuzzing-report-src

  BB2: https://git.hub.vwgroup.com/CARIAD/tools-bb-fuzzing-report-src/tree/feature/implement-basic-reporting

18.20.4 TODO

- [ ] Martin Willers and Slavic => Do the system build for cluster 8 instead so team Comet can patch the issue [x] E3-151753[573] - Faulty assumptions in babel_tools DONE
- [ ] Extend System Build to another team or more teams

572 https://devstack.vwgroup.com/confluence/spaces/VWOSCUS/pages/3909956924/System+Build+1727250996415

573 https://devstack.vwgroup.com/jira/browse/E3-151753?src=confmacro

Helpful Links

Proof Of Concept Fuzzing All in One documentation page:
CI-Fuzz Proof of Concept (see page 393)

Example for Jenkinsfile:

https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3_aux_kpi_lake/browse/Jenkinsfile

Jenkins Shared library:

https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_pipeline_generic_shared_lib_premium/browse/src/de/vwg/e3/ioi/premium/nodes/Nodes.groovy

Contact: Smaoui, Yacine (TP-S2) - CICD triton pipeline, jenkins expert

Contact: Baumung, Wjatscheslav (EXTERN: MHP) -

18.21 2024-11-20 Sync Fuzzing Pipeline with Jenkins Part 15 - Pipeline and Building Block Status

- [x] Dang, Quang Anh (EXTERN: ITK)
- [x] Baumung, Wjatscheslav (EXTERN: MHP)
- [x] Willers, Martin (T5-PG)
- [ ] Unknown User (e7hqlnm)
- [ ] Muckenschnabl, David Jose (EXTERN: ITK)
- [ ] Smaoui, Yacine (TZ-CC)
- [ ] Stefan Slosarek
- [x] Burkhard, Dietterle
- [x] Litke, Johann (EXTERN: Bertrandt)
- [x] Kutscher, Max (EXTERN: PTG)
- [ ] Niklas Henrich (CI)
- [ ] Jacek Trossen (CI)
- [x] Schaller, Bernd (T5-SF)

18.21.1 TODOs

- [ ] TODO: Implement interrupt feature for BB-1 to receive/check for SIGTERM and implement interrupt method
  Talk to Afrizal
- [ ] Code Intelligence will fix the cov-calc tool to not use old lib from pypi

18.21.2 Agenda

- @Slavik: status of Clang profile build
- @Slavik: status of BB integration
- @Quang & Radu: status of both BB
- @Quang: the ci_coverage_calculator-0.3.0-py3-none-any.whl tool for coverage calculation. Where to include them?
  - In docker container?
  - In Conan package
  - Install during build by CICD pipeline
- @Martin: discuss fuzzing workmode with QA
- @Martin: Decision on running fuzzing on which branch? Suggestion is to run it on develop branch (cluster6) and feature branch
- Future features:
  - Coverage evaluation before running the fuzz target. Skipping the targets with > 80 % code coverage
  - @Martin: Stopping fuzz run prematurely or trigger stop by other pipeline stages.
    TODO: discuss with QA
  - @Martin: Output of fuzz run to be presented in 100% report.
    TODO: how to link that with the @e3fuzzing-generated pages? https://devstack.vwgroup.com/confluence/pages/viewpage.action?pageId=3061303501
    TODO: discuss once the output is set by the 2nd BB and visualized by DataLakers
  - @Martin: Teams request to be able to run fuzz tests on feature branches. That should be made possible as an optional quality gate.
    The output of the feature-branch fuzzing runs should be shown in the job result page.
  - Discuss nightly run / weekend fuzz run in the future
  - Doxygen generation before calling 2nd BB. Status and how to speed this up with modified settings e.g. omit the generation of UML diagrams. Takes 20s at most. Currently using this settings:

    -GENERATE_HTML = YES

    -GENERATE_XML = NO

    +GENERATE_HTML = NO

    +GENERATE_XML = YES

18.21.3 Meeting minutes

27 Nov 2024

- Examine

@Radu: BB code does not properly propagate errors to the pipeline

- eg. "lcov: ERROR: cannot read file" should be an error
- eg. "Exit Code : 9" should be an error
- eg. "ERROR: Error downloading (corpus)"

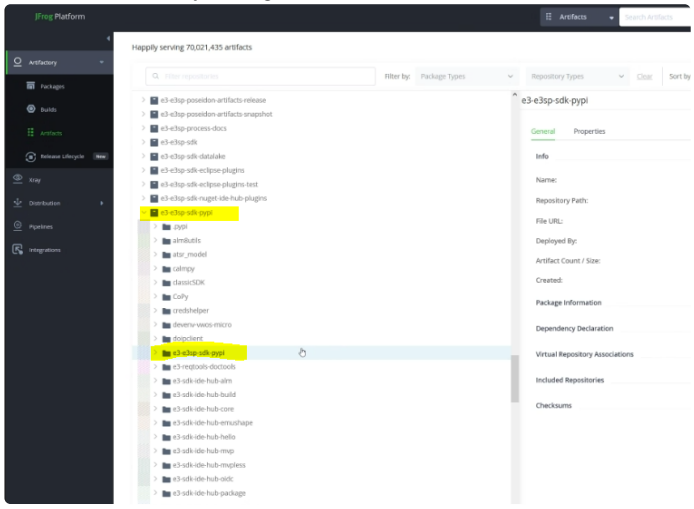
As a first approximation, whenever an executable returns with anything else than exit code 0, the BB should abort,

and the pipeline should fail. For those bad executable that always return 0 even on error, then you need to look

at eg. stdout/stderror output in order to determine the success.

=> This is fixed in the new version 1.0.1 of the Building Block 1

- Remote Repair and Diag Common fuzzing branch cluster 8
  => Slavik is running it
- Team Comet is merged to develop but not to master
- Upgrade Clang-18 not yet started. Hopefully, PMT will take this with them
- How to deal with https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-gen-teamcicd/ci_fuzz/ci_coverage_calculator-0.3.2-py3-none-any.whl?
  - Normally, we have a pypi artefactory to store this
  - Who decides where?
  - Solution; We are putting the whl file here:



- Where to store fuzzing result and which format?
  - Wait for Datalaker to finish their parsing and the format is final
  - @Martin: As long as the format is not final, we should not write the result in files. Only log in terminal
    - [ ] Dang, Quang Anh (EXTERN: ITK) create a flag to turn off file writing.
- New BP1 version:
  - https://devstack.vwgroup.com/bitbucket/projects/E3INT/repos/docker_e3sdk_build_bp1/pull-requests/314/overview
    Missing Tag

- Slavik

20 Nov 2024

- Update projects and branches to test system builds on:
  - Remote Repair: https://devstack.vwgroup.com/bitbucket/projects/E3APP/repos/e3_app_remote_repair/browse?at=refs%2Fheads%2Ffeature%2FE3APP007-1168-establish-cifuzz-testing
  - DiagMaster Common: https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_diagmaster_common/browse?at=refs%2Fheads%2Ffeature%2FVWOS-15920-establish-fuzzing-testing
- Interrupt feature POC:
  - https://git.hub.vwgroup.com/CARIAD/tools-bb-fuzzing-src/compare/main...feature/POC_interrupt_SIGINT
- PR new version commonpart
  - https://devstack.vwgroup.com/bitbucket/projects/E3INT/repos/docker_e3sdk_commonpart_build_bp1/pull-requests/333/overview
- System Build status:
  - Baumung, Wjatscheslav (EXTERN: MHP) execute the new system build with fuzzing branch for Remote Repair
  - Current system build:
    - https://devstack.vwgroup.com/confluence/pages/viewpage.action?spaceKey=TOOLCHAIN&title=System+Build+1731766749777
  - Slavik executing pipelines with fuzzing:
    - https://jenkins.ioi.imfa.hub.vwgroup.com/blue/organizations/jenkins/E3_APPS%2FE3APP_REM-REPAIR%2Fe3_app_remote_repair/detail/feature%2FE3ASWF-11003-conan-packages-fuzzing-module/20/pipeline/1389/
    - https://jenkins.ioi.imfa.hub.vwgroup.com/blue/organizations/jenkins/E3_APPS%2FE3APP_REM-REPAIR%2Fe3_app_remote_repair/detail/feature%2FE3ASWF-11003-conan-packages-fuzzing-module-WITHOUT-patched-SB-requires/5/pipeline
    - https://jenkins.ioi.imfa.hub.vwgroup.com/blue/organizations/jenkins/E3_APPS%2FE3APP_REM-REPAIR%2Fe3_app_remote_repair/detail/feature%2FE3ASWF-11003-conan-packages-fuzzing-module/17/pipeline
- Reporting with Datendeuter
  - Datendeuter is on it, No update in ticket sofar

    ☑ E3-154410[574] - [Reporting] Reporting of Fuzzing test results DONE
- End goal is to run system build for develop AND master
  - CICD team needs a fuzzing branch based on master => Create a feature branch

574 https://devstack.vwgroup.com/jira/browse/E3-154410?src=confmacro

13 Nov 2024

- Asynchronous prematurely stop of BB. Current idea is to use signal module to wait for SIGINT signal
  - This SIGINT is to be sent by other parallel pipeline
  - **how to implement interrupt**

```
def interrupt_handler(signal_num, frame):
print("\nSIGINT received. Exiting gracefully.")
sys.exit(0)

# Register the signal handler
signal.signal(signal.SIGINT, interrupt_handler)

def main():
try:
                …
except KeyboardInterrupt:
                sys.exit(0)
```

- Ticket to track Code Intelligence prorgress:
  - CVSS score of finding not in finding.json. Currently, one has to generate it separately. CI is on this issue and wil integrate this in a future Cifuzz version https://code-intelligence.atlassian.net/servicedesk/customer/portal/5/SD-193
  - ci_coverage_calculator using old path package from PyPI (ongoing) https://code-intelligence.atlassian.net/servicedesk/customer/portal/5/SD-194
  - Mismatching in doxygen header format and ci_coverage_calculator format:
    CI will fix this issue when the ci_coverage_calculator is integrated directly into cifuzz in a future version
    https://code-intelligence.atlassian.net/servicedesk/customer/portal/5/SD-192
- Slavik is trying to build EVERYTHING (every dependency) and it seems to work, however, Martin has to patch the Team Comet issue manually
  - The plan is to switch to cluster 8 system build, since the cluster 6 patch is very strict currently
  - successful build of all dependencies with Clang15 profile: https://devstack.vwgroup.com/confluence/display/TOOLCHAIN/System+Build+1727250996415
    - 2024-11-05: Clang15 build of “develop”: https://devstack.vwgroup.com/confluence/display/TOOLCHAIN/System+Build+1730821937945
    - 2024-11-06: Clang15 build of “develop”: https://devstack.vwgroup.com/confluence/display/TOOLCHAIN/System+Build+1730883202921

    (issues with QuAuq components, which have a vastly outdated “develop”)
    - 2024-11-06: Clang15 build of “master”: https://devstack.vwgroup.com/confluence/pages/viewpage.action?spaceKey=TOOLCHAIN&title=System+Build+1730898435467

- Upgrade Clang18 has a Customer Need ticket:
  PMTC-2755[575] - Update Clang in bp1 to v18 or newer CLOSED
  - Currently nothing happen => Probably December. Not before PI Planning CICD team
- Radu upload/download corpus
  - Radu wants to try running 2-3 pipeline at the same time to test concurence
  - TODO: Upload a hash for corpus.tar.gz to avoid corrupted corpus

18.21.4 TODO

- [ ] Martin Willers and Slavic => Do the system build for cluster 8 instead so team Comet can patch the issue E3-151753[576] - Faulty assumptions in babel_tools DONE
- [ ] Extend System Build to another team or more teams
- [x] Willers, Martin (T5-PG) open customer need tickets for Building Block manager
- [ ] Dang, Quang Anh (EXTERN: ITK) eliminate unicode logging from BB-1 and BB-2
- [ ] Dang, Quang Anh (EXTERN: ITK) eliminate sanitizer_death_callback_during generation of lcov
- [ ] Dang, Quang Anh (EXTERN: ITK) sync with Radu and give the BB-2 to Slavik to include in the pipeline

Helpful Links

Proof Of Concept Fuzzing All in One documentation page:
CI-Fuzz Proof of Concept (see page 393)

Example for Jenkinsfile:

https://devstack.vwgroup.com/bitbucket/projects/E3SPAUX/repos/e3_aux_kpi_lake/browse/Jenkinsfile

Jenkins Shared library:

https://devstack.vwgroup.com/bitbucket/projects/E3SDK/repos/e3_pipeline_generic_shared_lib_premium/browse/src/de/vwg/e3/ioi/premium/nodes/Nodes.groovy

Contact: Smaoui, Yacine (TP-S2) - CICD triton pipeline, jenkins expert

Contact: Baumung, Wjatscheslav (EXTERN: MHP) -

575 https://devstack.vwgroup.com/jira/browse/PMTC-2755?src=confmacro
576 https://devstack.vwgroup.com/jira/browse/E3-151753?src=confmacro

19 Fuzzing Presentation to Management Slides

Version	Date	Updated by	Change log
1.0	17 Mar 2023	Dang, Quang Anh (EXTERN: ITK)	First uploaded
1.1	13 Jun 2023	Dang, Quang Anh (EXTERN: ITK)	Updated the status of the teams and graph of risk driven fuzzing process and release criteria
1,2	30 Aug 2023	Dang, Quang Anh (EXTERN: ITK)	Update onboarded teams and timeline and useful links in the end
1.3	23 Oct 2023	Muckenschnabl, David Jose (EXTERN: ITK)	Update date, fix typos, update result tables
2.0	31 Oct 2023	Dang, Quang Anh (EXTERN: ITK)	(Draft) Major update after management feedback
2.1	24 Nov 2023	Dang, Quang Anh (EXTERN: ITK)	Update implemented feedback from Bernd Schaller
3.0	03 Apr 2024	Dang, Quang Anh (EXTERN: ITK)	Added Doxygen and CICD Concept. Implemented Daniel Neeb feedback
3.1	09 Apr 2024	Dang, Quang Anh (EXTERN: ITK)	Implemented Bernd feedback
3.2	12 Apr 2024	Dang, Quang Anh (EXTERN: ITK)	Added google perspective regarding memory safety (Andreas Feedback)
3.3	23 Oct 2024	Dang, Quang Anh (EXTERN: ITK)	Updated changes from CICD Pipeline and Pilot with Remote Repair

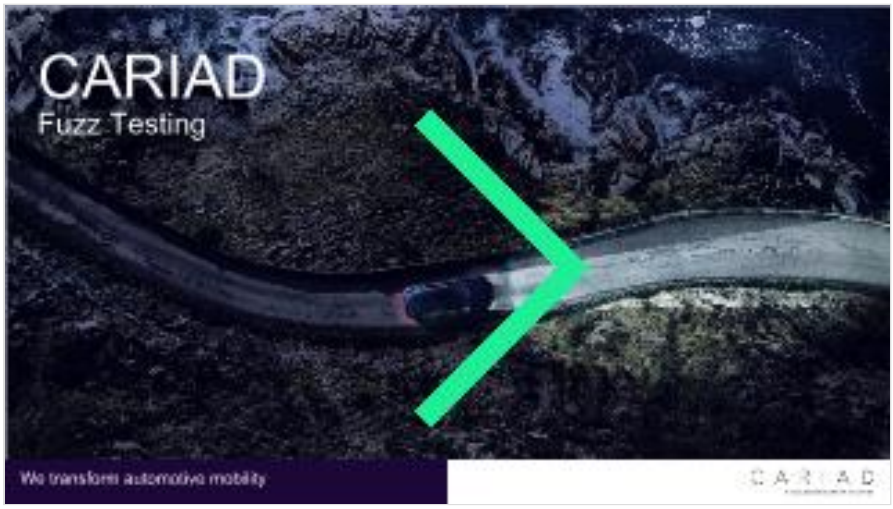
Version	Date	Updated by	Change log
4.0	13 Nov 2024	Dang, Quang Anh (EXTERN: ITK)	Implemented Daniel's feedback on pilot with Remote Repair

Protocol:

10 Mar 2023 Fuzzing Review with VW-OS Core PL Board

PDF:

PPT:



20 Q&A and Troubleshooting CIFuzz

Search here:

Search

20.1 ICP ART

20.1.1 LUM Local Update Manager

Where to find BP1_VERSION ?

Search for build_container version

Example:

In e3_comp_lum[577] we followed these steps to figure out the BP1_VERSION

1. Read **build.sh** there we saw that **e3sdk_scripts/docker_wrapper.sh** is used for building.
2. Read **docker_wrapper.sh**. The get_build_container function is used to get the version from the **e3sdk_metadata** file.
3. Read **e3sdk_metadata** file and saw the build container is version 4.2.0 in the case of e3swp which was the **$TARGET** in the **docker_wrapper.sh**.

20.1.2 Adaptive Adaptive Stack (EB)

How to fix ERROR: Failed conan config install: [Errno 13] Permission denied: '/opt/e3sdk/conan_settings/profiles/fuzzing-x86_64'

Set the permission of the fuzzing-x86_64 file to 664:

```
$ chmod 664 fuzzing-x86_64
```

Editing of CMakelist.txt file for cifuzz use case is too complicated. Can we make this more simple "one line" somehow?

Currently, this is a topic to be worked on. No alternative solution sofar.

20.1.3 DiagMaster Serviceorientierte Diagnose

Error: TARGET_NAME is not defined in conanfile

577 https://devstack.vwgroup.com/bitbucket/projects/E3COMP/repos/e3_comp_lum/browse

As CIFuzz builds the targets internally it does not call conan. Because of this the TARGET_NAME and CONAN_PKG_VERSION variables are never set.

To go around this we set the variables inside our block, before they are checked

Block modification

```
if(DEFINED ENV{CIFUZZ})
    ...

    # Workaround: conanfile.py is not executed by CIFuzz thus we will hardcode the
variables here.
    set(TARGET_NAME "diagupdate")
    set(CONAN_PKG_VERSION "10.3.16")
endif()

...

if(NOT TARGET_NAME)
...

if(NOT CONAN_PKG_VERSION)
...
```

20.2 BSrv

20.2.1 CodeBuster Function on demand (2.0)

When runining/building a fuzz target code under the submodule folder is not found

When cloning a repository with submodules, the submodules have to also be cloned.

Clone Sub-repositories

```
git submodule update --init --recursive
```

What if there is an assumption/requirement that the data received by a function is to be TRUSTED?

- TODO

When to write this fuzztest? Parallel to the unit tests or in the end?

Fuzz Test is to be incorporated into the Test Master Plan 2.0. If the component is sec relevant.

What is the goal of the fuzzing tools? Will it run on the CICD pipeline?

The goal is to have it run on the CIFuzz server independently.

This will be triggered by the CICD Pipeline

Whats the status for SRA and SRS?

N/A

What about 3rd party component and standard code? Does it make sense to fuzz them?

We should fuzz your own implementation. It highly depends on the code of the teams.

Can you give examples of criteria for functions to be security relevant?

- Outside/untrusted input
- Parsers
- Cryptographic related functions
- Functions that check for keys

20.2.2 LostPilot

We develop two independetn component, one in Mid Stack and one in Micro Stack, soes micro Stack have to be considered while fuzzing ?

Yes Micro stack will be also considered in fuzzing.

Will the building also work for 2.0 as it does not have a build.sh?

Yes it should work as we also have our own script.

Can we create a personalized way to manage input structure nicely?

Yes, CIFuzz lets the user define it's own mutator for the inputs. This can be complex and make the testing slowdown. But is possible.

Can we use CIFuzz to determine if a new implementation works the same as an old one ? Can we compare implementations ? Is there a way to manage input nicely?

Parallel testing can be done.

CIFuzz provides an object called FuzzedDataProvider, this can be used to input data with more ease.

The server runs a fresh run every time, how can one make sure this will stay deterministic ?

Input of bugs and new found branches are saved in the corpus. The target stores these and will always execute them in a new run.
Are runs deterministic or totally randomized? Are runs saved?

A run mutates the input, thus it is kind of random but not totally. I a bug is found and the corpus is not modified the run will get to the same bug eminently. This might change in the case of a state machine. When fuzzing a state machine care has to be taken to start with the wished state.
Is the debug and fix target step of the fuzzing workflowautomatic?

No, this step is "hand work" in that a bug report is automatically generated but the fix is dome by a developer that has to debug find the problem.
Do we fuzz on component level or function level ?

We fuzz on function level

Is Fuzzing coverage driven ?

Yes the project is compiled with instrumentation, to track coverage.

20.3 General Troubleshooting

Before going into specific use cases, there are some steps to perform which in general solves most of the building/fuzzing issue:

1. Do a clean build by removing the cache of the old build.

Removing cache of the old build

```
rm -r .cifuzz-build
rm -r build
rm -r build-cov
```

2. Remove conan cache via

Removing cache of the old build

```
rm -r ~/conan/data
```

3. Check permission of the files inside the docker container. Since the files in the current directory will be mounted into the docker container, sometimes the file in the current directory. Make sure to set the permission of the files to 1000:1000

20.3.1 Docker Related

How to solve '#3 ERROR: failed to authorize: failed to fetch anonymous token: unexpected status: 401 Unauthorized'

First and foremost, make sure you have access to the resource by looking for it on https://jfrog.devstack.vwgroup.com/[578]. If you can't find it, it is likely that you don't have access for it. In that case, request access for it https://devstack.vwgroup.com/jira/servicedesk/customer/portal/6. Fill in your team and your PO.

578 https://jfrog.devstack.vwgroup.com/ui/repos/tree/General/e3-e3sp-deb-repo

Add conan user, login to docker remote repo and request access to the repository

```
# First check conan user. Note that this command is only available for conan 1.X

$ conan user
Current user of remote 'e3-release' set to: 'XXXXXXX' [Authenticated]
Current user of remote 'e3-snapshot' set to: 'None' (anonymous)
Current user of remote 'e3-app-release' set to: 'None' (anonymous)
Current user of remote 'e3-app-snapshot' set to: 'None' (anonymous)
Current user of remote 'e3-vwos-production' set to: 'None' (anonymous)

#If it is anything like above, you have to specify user name for all the remotes
conan user XXXXXXX -r ****

# If it doesn't work, try to login to the docker repo for example via. Put in your
respective repository of course
docker login e3-e3sp-docker-3rdparty.docker.devstack.vwgroup.com
docker login docker-public-virtual.docker.devstack.vwgroup.com
docker login e3-e3sp-dockerdev.docker.devstack.vwgroup.com
```

20.3.2 Questions regarding building and error fixing

When building docker checkout <ci-fuzz version> does not succed

Add **--no-cache** flag to docker build command.
Ubuntu 18.04 comes with cmake 3.10, but your template CMakeLists.txt requires cmake 3.16. What shall I do?

Update cmake to 3.16 or higher.
My component's minimum required cmake version is 3.10 and this clashes with the minimum required cmake version of 3.16 of the fuzzing makefiles

Please update your cmake dependencies to 3.16
"cifuzz.sh run" shows compilation errors. How can I improve verbosity to see whats going wrong?

First rerun ./cifuzz.sh with "--verbose" parameter.
If that does not reveal enough information, increase the CMake verbosity by adding
" `set(CMAKE_VERBOSE_MAKEFILE ON)` " to your CMakelist.txt
"cifuzz.sh run" shows linking errors. What could be the source of the problem?

Error such as

```
clang: error: linker command failed with exit code 1 (use -v to see invocation)
```

could have different causes such as:

- The new clang compiler is to new for the linker => In this case, you can tell clang to use an older DWARF format or use a more current linker.

The build target runs, but coverage is not written. What's wrong?

Your function under test may opt out very early in the game.
Enable stdout via "--verbose" parameter. If that does not help launch your target with "`NO_CIFUZZ=1 gdb <path_to_executable>`"

What type of functions should be fuzzed?

In general, functions which receive tainted data/inputs are good candidates for fuzzing

What are the acceptance criteria for fuzzing? How long should a fuzz test run until a SUT is considered "passed"?

Currently, the requirements for fuzzing are yet to be specified. However, there are some best practices e.g

- Code coverage alone is not good enough metric for fuzzing, since there is often dead code or code irrelevant for fuzzings.
- Identify which parts of the application are critical and focus on getting a high code coverage in these parts. This can be done by either filtering the code coverage for specific files or by just checking if all functions from a specific interface are covered/called by the fuzz test. The architecture engineer/test engineer might have some insight.
- Limited resources e.g time, CPUs, memory are to be divided accordingly to severity of the fuzz targets

What version should I add in the "Affects Version" field of a Bug report

The version is derived from the year and the calendar week in the following way: **R[last two cifers of the year][calendar week]** for example R2220 = release from year 2022 week 20.

E3-99887 eb-ara_DM - undefined reference to get_application_id() while building

The issue E3-99887[579] occurred while building OTA Master `e3_comp_otam_common` with CI-Fuzz

CMake variables are not set, as conanfile.py::cmake_configure is not called

Please hard code these variables in the fuzzing section of your CMakeList and stay tuned for the SDK integration of the tool

Partial instrumentation only

Only the component under test is rebuild with instrumentation (fuzzing, sanitizers). Its dependencies are not touched as existing conan build profiles are reused.
A fully instrumented build requires a new profile and must be supported by the SDK.

Sometimes, the cifuzz build fails because certain header file can not be found. What to do?

Look into the different CMakeList.txt files. Sometimes, there would be something like this

```
if ("${BUILD_PLATFORM}" STREQUAL "E3_1_1")
    include(${CMAKE_CURRENT_LIST_DIR}/e3_1_1/CMakeLists.txt)
    message(STATUS "BUILD_PLATFORM SET to 1.1")
elseif ("${BUILD_PLATFORM}" STREQUAL "E3_1_2")
    include(${CMAKE_CURRENT_LIST_DIR}/e3_1_2/CMakeLists.txt)
```

579 https://devstack.vwgroup.com/jira/browse/E3-99887

```
    message(STATUS "BUILD_PLATFORM SET to 1.2")
endif()
```

Since we build with CIFUZZ, some VARIABLES might not be set. Make sure to manually set it.

When building docker container ERROR: failed to solve: failed to fetch anonymous token: unexpected status: 401 Unauthorized

Docker cannot log-in into the SDK server we pull from.

Run:

```
sudo docker login e3-e3sp-dockerdev.docker.devstack.vwgroup.com
```

and input your Cariad internal ID and W-TAMU password

Conan cannot log into remote: Remote 'remote_x' username: ERROR: Failed requirement 'remote_x' from 'conanfile.py (lib_x)'

Conan cannot login into remote

Open a shell in docker container:

```
$ ./cifuzz.sh /shell
```

Check user status for remotes:

```
$ conan user \n

Current user of remote 'e3-release' set to: 'None' (anonymous)

Current user of remote 'e3-snapshot' set to: 'None' (anonymous)

Current user of remote 'e3-app-release' set to: 'None' (anonymous)

Current user of remote 'e3-app-snapshot' set to: 'None' (anonymous)

Current user of remote 'e3-vwos-production' set to: 'None' (anonymous)
```

If the user is set to None, then you need to set them to your CARIAD user:

```
$ conan user <your global ID> -r e3-release
$ conan user <your global ID> -r e3-snapshot
$ conan user <your global ID> -r e3-app-release
```

```
$ conan user <your global ID> -r e3-app-snapshot
$ conan user <your global ID> -r e3-vwos-production
```

Next steps to solve is to set the enviroment variable to your Cariad internal ID:

```
$ export CONAN_LOGIN_USERNAME=<your global ID>
```

after this conan should prompt for your password next time it tries to pull from a remote.

More infos on conan user : https://docs.conan.io/1/reference/commands/misc/user.html

"error: a function declaration without a prototype is deprecated in all versions of C [-Werror,-Wstrict-prototypes]" while running ./cifuzz.sh run

Add followings in the if(DEFINED ENV{CIFUZZ}) block inside the root CMakeList.txt file

```
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -Wno-error=strict-prototypes")
```

Conan lock is held after running cifuzz.sh script

Solution: delete conan cache inside docker container

delete conan cache

```
./cifuzz.sh /shell          # To open a shell in the docker container
rm -rf ~/.conan/data        # To remove conan cache
```

ERROR building docker image because of cifuzz git checkout exit code 1

Try run docker build with --no-cache option

```
docker build . --tag cifuzz --build-arg BP1_VERSION=x.y.z --no-cache --build-arg
https_proxy --build-arg http_proxy
```

How to suppress error 'unknown warning option' while building with cmake?

Add these lines to CMakeLists.txt

```
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -Wno-error=unknown-warning-option")
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wno-error=unknown-warning-option")
```

How to suppress 'strict-prototypes' error while building with cmake?

Add these lines to CMakeLists.txt

```
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -Wno-error=strict-prototypes")
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wno-error=strict-prototypes")
```

How to fix build error: "Could NOT find Threads (missing: Threads_FOUND)"

This issue is probably due to clang compiler not identifying libpthread automatically. In this case, you need to add following into your root CMakeLists.txt file for hotfix.

Comment out this

```
# TODO: This would fix the Error. Probably only occur with CLANG
# Make Error at /usr/share/cmake-3.16/Modules/FindPackageHandleStandardArgs.cmake:146
(message):
#   Could NOT find Threads (missing: Threads_FOUND)
# Call Stack (most recent call first):
#   /usr/share/cmake-3.16/Modules/FindPackageHandleStandardArgs.cmake:393
(_FPHSA_FAILURE_MESSAGE)
#   /usr/share/cmake-3.16/Modules/FindThreads.cmake:220
(FIND_PACKAGE_HANDLE_STANDARD_ARGS)
#   CMakeLists.txt:65 (find_package)
# find_package(Threads REQUIRED)
set(CMAKE_THREAD_LIBS_INIT "-lpthread")
set(CMAKE_HAVE_THREADS_LIBRARY 1)
set(CMAKE_USE_WIN32_THREADS_INIT 0)
set(CMAKE_USE_PTHREADS_INIT 1)
set(THREADS_PREFER_PTHREAD_FLAG ON)
```

How to fix build error: "Multiple definitions of main"

One build shall only have one instance of function main defined while building for fuzzing. If there is another instance defined somewhere, this should be commented out in the code e.g.,

Comment out this

```
# Main executable
#add_executable(${TARGET_NAME} main.cpp)
```

How do I login to the docker container as root?

You should check your ~/.conan/setting.yml file. Possibly, there are some options missing in there. Add these 2 lines to fix the problem

su

< Password is root

ERROR: Failed requirement '...' from 'conanfile.py'. Unable to find '...' in remote

1. Check your ~/.conan/remotes.json and find the link to the artefactory where the packages are hosted. Usually "https://jfrog.devstack.vwgroup.com/artifactory/"
2. Visit https://jfrog.devstack.vwgroup.com/artifactory/ and look for the package which is missing. Go to:
   Artefacts => 'e3-e3sp-conan-release' => <the path to package as specified in conanfile.py>
   Example: e3-e3sp-conan-release/e3swp3rdparty/e3-poco/5001.10.1-23.1/stable/...
3. If you can't find it there, then you probably don't have access. In that case, please apply for the right access right here: https://devstack.vwgroup.com/jira/servicedesk/customer/portals

How to fix build error: "ERROR: 'setting.os.target' doesn't exist for 'Linux'"

You should check your ~/.conan/setting.yml file. Possibly, there are some options missing in there. Add these 2 lines to fix the problem

```
os:
...
 Linux:
        target: [local]
...
```

How to fix build error: "ERROR: failed to solve: failed to fetch anonymous token: unexpected status: 401 Unauthorized"

Login to the docker remote server via:

```
docker login e3-e3sp-docker-3rdparty.docker.devstack.vwgroup.com
```

During build I got error 'CMake Error at /usr/share/cmake-3.16/Modules/CMakeDetermineSystem.cmake:99 Could not find toolchain file: /opt/e3sdk/toolchain/toolchainfile_wrapper.cmake'

Using the build.sh script from PMT Solution. Your $HOME/.conan directory will be mounted into /home/e3-user/.conan of the docker container. Hence, the conan config inside the docker container will be identical to that of your local machine.

For some reason, the CMAKE Variable CMAKE_TOOLCHAIN_FILE is set inside the /usr/share/cmake-3.16/Modules/CMakeDetermineSystem.cmake file. This is the result of the conan ENV configuration CONAN_CMAKE_TOOLCHAIN_FILE. This can be overwritten or set inside:

~/.conan/conan.cfg

To fix this issue, add following line under [general] in ~/.conan/conan.cfg:

conan.cfg

```
...
[general]
...
cmake_toolchain_file = ''
...
```

This will force the CONAN_CMAKE_TOOLCHAIN_FILE variable to be empty and consequently, also CMAKE_TOOLCHAIN_FILE, which solves the problem.

Cariad user has no pull docker rights

conan.cfg

```
the problem could be that the login needs an apikey instead of a password.
https://jfrog.devstack.vwgroup.com/ui/user_profile
There is a field "API Key"
Can you try
docker login e3-e3sp-dockerdev.docker.devstack.vwgroup.com
UserName: <YourId>
Password: <ApiKey>
```