ABDK CONSULTING

SMART CONTRACT AUDIT

AAVE

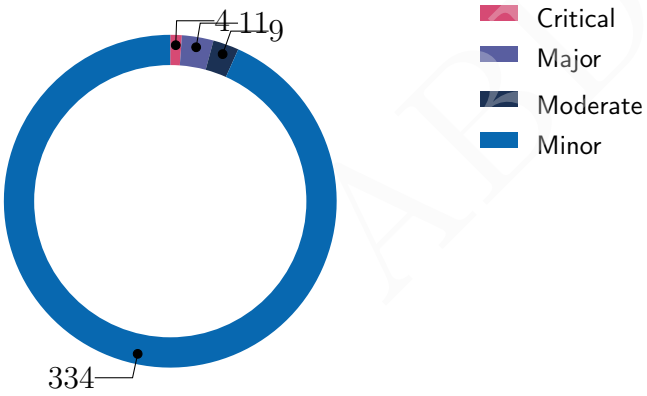
Solidity

abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
26th January 2022

We've been asked to review the 59 files in a GitHub repository. We found 4 critical, 11 major, and a few less important issues.



Findings

ID	Severity	Category	Status
CVF-1	Minor	Procedural	Opened
CVF-2	Minor	Readability	Opened
CVF-3	Minor	Readability	Opened
CVF-4	Minor	Suboptimal	Opened
CVF-5	Minor	Suboptimal	Opened
CVF-6	Minor	Procedural	Opened
CVF-7	Minor	Bad datatype	Opened
CVF-8	Minor	Procedural	Opened
CVF-9	Minor	Suboptimal	Opened
CVF-10	Minor	Suboptimal	Opened
CVF-11	Minor	Overflow/Underflow	Opened
CVF-12	Minor	Suboptimal	Opened
CVF-13	Minor	Suboptimal	Opened
CVF-14	Minor	Suboptimal	Opened
CVF-15	Minor	Readability	Opened
CVF-16	Major	Unclear behavior	Opened
CVF-17	Minor	Suboptimal	Opened
CVF-18	Minor	Procedural	Opened
CVF-19	Minor	Bad datatype	Opened
CVF-20	Minor	Suboptimal	Opened
CVF-21	Minor	Documentation	Opened
CVF-22	Minor	Suboptimal	Opened
CVF-23	Minor	Suboptimal	Opened
CVF-24	Minor	Suboptimal	Opened
CVF-25	Minor	Suboptimal	Opened
CVF-26	Minor	Documentation	Opened
CVF-27	Minor	Suboptimal	Opened

ID	Severity	Category	Status
CVF-28	Minor	Suboptimal	Opened
CVF-29	Minor	Suboptimal	Opened
CVF-30	Minor	Readability	Opened
CVF-31	Minor	Readability	Opened
CVF-32	Minor	Readability	Opened
CVF-33	Minor	Documentation	Opened
CVF-34	Minor	Suboptimal	Opened
CVF-35	Major	Suboptimal	Opened
CVF-36	Minor	Suboptimal	Opened
CVF-37	Major	Procedural	Opened
CVF-38	Minor	Suboptimal	Opened
CVF-39	Critical	Flaw	Opened
CVF-40	Minor	Documentation	Opened
CVF-41	Minor	Unclear behavior	Opened
CVF-42	Moderate	Suboptimal	Opened
CVF-43	Major	Flaw	Opened
CVF-44	Minor	Suboptimal	Opened
CVF-45	Minor	Suboptimal	Opened
CVF-46	Minor	Suboptimal	Opened
CVF-47	Minor	Suboptimal	Opened
CVF-48	Minor	Suboptimal	Opened
CVF-49	Minor	Overflow/Underflow	Opened
CVF-50	Minor	Suboptimal	Opened
CVF-51	Minor	Suboptimal	Opened
CVF-52	Minor	Suboptimal	Opened
CVF-53	Minor	Suboptimal	Opened
CVF-54	Major	Suboptimal	Opened
CVF-55	Minor	Bad datatype	Opened
CVF-56	Minor	Bad datatype	Opened
CVF-57	Major	Suboptimal	Opened

ID	Severity	Category	Status
CVF-58	Minor	Suboptimal	Opened
CVF-59	Minor	Suboptimal	Opened
CVF-60	Minor	Unclear behavior	Opened
CVF-61	Minor	Unclear behavior	Opened
CVF-62	Minor	Suboptimal	Opened
CVF-63	Minor	Suboptimal	Opened
CVF-64	Minor	Bad naming	Opened
CVF-65	Minor	Bad datatype	Opened
CVF-66	Critical	Flaw	Opened
CVF-67	Minor	Suboptimal	Opened
CVF-68	Minor	Bad datatype	Opened
CVF-69	Minor	Readability	Opened
CVF-70	Minor	Bad naming	Opened
CVF-71	Minor	Suboptimal	Opened
CVF-72	Minor	Readability	Opened
CVF-73	Minor	Suboptimal	Opened
CVF-74	Minor	Suboptimal	Opened
CVF-75	Minor	Bad datatype	Opened
CVF-76	Minor	Suboptimal	Opened
CVF-77	Minor	Readability	Opened
CVF-78	Minor	Readability	Opened
CVF-79	Minor	Suboptimal	Opened
CVF-80	Minor	Suboptimal	Opened
CVF-81	Minor	Suboptimal	Opened
CVF-82	Minor	Documentation	Opened
CVF-83	Minor	Suboptimal	Opened
CVF-84	Minor	Suboptimal	Opened
CVF-85	Minor	Overflow/Underflow	Opened
CVF-86	Minor	Bad naming	Opened
CVF-87	Minor	Suboptimal	Opened

ID	Severity	Category	Status
CVF-88	Minor	Suboptimal	Opened
CVF-89	Major	Flaw	Opened
CVF-90	Minor	Bad naming	Opened
CVF-91	Minor	Procedural	Opened
CVF-92	Minor	Documentation	Opened
CVF-93	Minor	Documentation	Opened
CVF-94	Minor	Bad naming	Opened
CVF-95	Minor	Documentation	Opened
CVF-96	Minor	Bad datatype	Opened
CVF-97	Minor	Documentation	Opened
CVF-98	Minor	Suboptimal	Opened
CVF-99	Minor	Bad naming	Opened
CVF-100	Minor	Suboptimal	Opened
CVF-101	Minor	Bad datatype	Opened
CVF-102	Minor	Bad datatype	Opened
CVF-103	Minor	Documentation	Opened
CVF-104	Minor	Documentation	Opened
CVF-105	Minor	Documentation	Opened
CVF-106	Minor	Bad naming	Opened
CVF-107	Minor	Procedural	Opened
CVF-108	Minor	Bad datatype	Opened
CVF-109	Minor	Unclear behavior	Opened
CVF-110	Minor	Documentation	Opened
CVF-111	Minor	Documentation	Opened
CVF-112	Minor	Suboptimal	Opened
CVF-113	Minor	Bad naming	Opened
CVF-114	Minor	Documentation	Opened
CVF-115	Minor	Bad datatype	Opened
CVF-116	Minor	Bad naming	Opened
CVF-117	Minor	Documentation	Opened

ID	Severity	Category	Status
CVF-118	Minor	Suboptimal	Opened
CVF-119	Minor	Unclear behavior	Opened
CVF-120	Minor	Suboptimal	Opened
CVF-121	Minor	Bad naming	Opened
CVF-122	Minor	Bad datatype	Opened
CVF-123	Minor	Suboptimal	Opened
CVF-124	Minor	Bad datatype	Opened
CVF-125	Minor	Suboptimal	Opened
CVF-126	Minor	Suboptimal	Opened
CVF-127	Minor	Bad datatype	Opened
CVF-128	Minor	Suboptimal	Opened
CVF-129	Minor	Bad datatype	Opened
CVF-130	Minor	Documentation	Opened
CVF-131	Minor	Bad datatype	Opened
CVF-132	Minor	Readability	Opened
CVF-133	Minor	Suboptimal	Opened
CVF-134	Minor	Suboptimal	Opened
CVF-135	Minor	Suboptimal	Opened
CVF-136	Minor	Suboptimal	Opened
CVF-137	Minor	Suboptimal	Opened
CVF-138	Minor	Suboptimal	Opened
CVF-139	Minor	Bad datatype	Opened
CVF-140	Minor	Documentation	Opened
CVF-141	Minor	Documentation	Opened
CVF-142	Minor	Bad datatype	Opened
CVF-143	Minor	Documentation	Opened
CVF-144	Minor	Bad datatype	Opened
CVF-145	Minor	Suboptimal	Opened
CVF-146	Minor	Documentation	Opened
CVF-147	Minor	Suboptimal	Opened

ID	Severity	Category	Status
CVF-148	Minor	Suboptimal	Opened
CVF-149	Minor	Suboptimal	Opened
CVF-150	Minor	Suboptimal	Opened
CVF-151	Minor	Suboptimal	Opened
CVF-152	Minor	Suboptimal	Opened
CVF-153	Moderate	Suboptimal	Opened
CVF-154	Minor	Suboptimal	Opened
CVF-155	Minor	Suboptimal	Opened
CVF-156	Minor	Bad datatype	Opened
CVF-157	Minor	Suboptimal	Opened
CVF-158	Minor	Documentation	Opened
CVF-159	Minor	Procedural	Opened
CVF-160	Moderate	Flaw	Opened
CVF-161	Minor	Bad datatype	Opened
CVF-162	Minor	Bad datatype	Opened
CVF-163	Minor	Flaw	Opened
CVF-164	Minor	Suboptimal	Opened
CVF-165	Minor	Flaw	Opened
CVF-166	Minor	Unclear behavior	Opened
CVF-167	Minor	Suboptimal	Opened
CVF-168	Minor	Suboptimal	Opened
CVF-169	Minor	Suboptimal	Opened
CVF-170	Minor	Bad naming	Opened
CVF-171	Minor	Documentation	Opened
CVF-172	Moderate	Flaw	Opened
CVF-173	Minor	Suboptimal	Opened
CVF-174	Minor	Suboptimal	Opened
CVF-175	Minor	Suboptimal	Opened
CVF-176	Minor	Flaw	Opened
CVF-177	Minor	Readability	Opened

ID	Severity	Category	Status
CVF-178	Minor	Suboptimal	Opened
CVF-179	Minor	Procedural	Opened
CVF-180	Minor	Suboptimal	Opened
CVF-181	Minor	Suboptimal	Opened
CVF-182	Minor	Procedural	Opened
CVF-183	Moderate	Unclear behavior	Opened
CVF-184	Minor	Suboptimal	Opened
CVF-185	Minor	Bad datatype	Opened
CVF-186	Major	Suboptimal	Opened
CVF-187	Minor	Documentation	Opened
CVF-188	Minor	Suboptimal	Opened
CVF-189	Minor	Documentation	Opened
CVF-190	Moderate	Suboptimal	Opened
CVF-191	Minor	Documentation	Opened
CVF-192	Minor	Procedural	Opened
CVF-193	Minor	Procedural	Opened
CVF-194	Minor	Suboptimal	Opened
CVF-195	Minor	Bad naming	Opened
CVF-196	Minor	Suboptimal	Opened
CVF-197	Minor	Suboptimal	Opened
CVF-198	Minor	Readability	Opened
CVF-199	Minor	Suboptimal	Opened
CVF-200	Minor	Procedural	Opened
CVF-201	Minor	Bad naming	Opened
CVF-202	Minor	Suboptimal	Opened
CVF-203	Minor	Suboptimal	Opened
CVF-204	Minor	Bad datatype	Opened
CVF-205	Minor	Suboptimal	Opened
CVF-206	Minor	Readability	Opened
CVF-207	Minor	Unclear behavior	Opened

ID	Severity	Category	Status
CVF-208	Major	Procedural	Opened
CVF-209	Minor	Procedural	Opened
CVF-210	Minor	Bad naming	Opened
CVF-211	Minor	Bad datatype	Opened
CVF-212	Minor	Documentation	Opened
CVF-213	Minor	Readability	Opened
CVF-214	Minor	Bad datatype	Opened
CVF-215	Minor	Readability	Opened
CVF-216	Minor	Suboptimal	Opened
CVF-217	Minor	Documentation	Opened
CVF-218	Minor	Overflow/Underflow	Opened
CVF-219	Minor	Suboptimal	Opened
CVF-220	Minor	Overflow/Underflow	Opened
CVF-221	Minor	Readability	Opened
CVF-222	Minor	Suboptimal	Opened
CVF-223	Minor	Bad datatype	Opened
CVF-224	Minor	Bad datatype	Opened
CVF-225	Minor	Bad datatype	Opened
CVF-226	Minor	Suboptimal	Opened
CVF-227	Minor	Suboptimal	Opened
CVF-228	Minor	Suboptimal	Opened
CVF-229	Major	Suboptimal	Opened
CVF-230	Minor	Suboptimal	Opened
CVF-231	Major	Suboptimal	Opened
CVF-232	Minor	Readability	Opened
CVF-233	Minor	Suboptimal	Opened
CVF-234	Minor	Suboptimal	Opened
CVF-235	Minor	Suboptimal	Opened
CVF-236	Minor	Documentation	Opened
CVF-237	Minor	Bad datatype	Opened

ID	Severity	Category	Status
CVF-238	Minor	Documentation	Opened
CVF-239	Minor	Bad naming	Opened
CVF-240	Minor	Suboptimal	Opened
CVF-241	Minor	Suboptimal	Opened
CVF-242	Minor	Bad naming	Opened
CVF-243	Minor	Bad datatype	Opened
CVF-244	Minor	Suboptimal	Opened
CVF-245	Minor	Suboptimal	Opened
CVF-246	Minor	Suboptimal	Opened
CVF-247	Minor	Flaw	Opened
CVF-248	Minor	Bad datatype	Opened
CVF-249	Minor	Bad datatype	Opened
CVF-250	Minor	Bad naming	Opened
CVF-251	Minor	Bad datatype	Opened
CVF-252	Minor	Procedural	Opened
CVF-253	Minor	Bad datatype	Opened
CVF-254	Minor	Readability	Opened
CVF-255	Minor	Suboptimal	Opened
CVF-256	Minor	Suboptimal	Opened
CVF-257	Minor	Bad datatype	Opened
CVF-258	Minor	Procedural	Opened
CVF-259	Minor	Bad datatype	Opened
CVF-260	Minor	Documentation	Opened
CVF-261	Minor	Suboptimal	Opened
CVF-262	Minor	Bad naming	Opened
CVF-263	Minor	Bad datatype	Opened
CVF-264	Minor	Readability	Opened
CVF-265	Minor	Suboptimal	Opened
CVF-266	Moderate	Unclear behavior	Opened
CVF-267	Minor	Procedural	Opened

ID	Severity	Category	Status
CVF-268	Minor	Suboptimal	Opened
CVF-269	Moderate	Unclear behavior	Opened
CVF-270	Moderate	Unclear behavior	Opened
CVF-271	Minor	Suboptimal	Opened
CVF-272	Minor	Readability	Opened
CVF-273	Minor	Suboptimal	Opened
CVF-274	Minor	Overflow/Underflow	Opened
CVF-275	Minor	Suboptimal	Opened
CVF-276	Minor	Suboptimal	Opened
CVF-277	Minor	Documentation	Opened
CVF-278	Minor	Procedural	Opened
CVF-279	Minor	Suboptimal	Opened
CVF-280	Minor	Suboptimal	Opened
CVF-281	Minor	Suboptimal	Opened
CVF-282	Minor	Suboptimal	Opened
CVF-283	Minor	Overflow/Underflow	Opened
CVF-284	Minor	Procedural	Opened
CVF-285	Minor	Readability	Opened
CVF-286	Minor	Readability	Opened
CVF-287	Critical	Flaw	Opened
CVF-288	Minor	Suboptimal	Opened
CVF-289	Minor	Suboptimal	Opened
CVF-290	Minor	Suboptimal	Opened
CVF-291	Minor	Procedural	Opened
CVF-292	Minor	Suboptimal	Opened
CVF-293	Minor	Suboptimal	Opened
CVF-294	Minor	Suboptimal	Opened
CVF-295	Minor	Suboptimal	Opened
CVF-296	Minor	Suboptimal	Opened
CVF-297	Minor	Suboptimal	Opened

ID	Severity	Category	Status
CVF-298	Minor	Suboptimal	Opened
CVF-299	Minor	Documentation	Opened
CVF-300	Critical	Flaw	Opened
CVF-301	Minor	Bad naming	Opened
CVF-302	Minor	Suboptimal	Opened
CVF-303	Minor	Readability	Opened
CVF-304	Minor	Suboptimal	Opened
CVF-305	Minor	Suboptimal	Opened
CVF-306	Minor	Suboptimal	Opened
CVF-307	Minor	Procedural	Opened
CVF-308	Minor	Suboptimal	Opened
CVF-309	Minor	Suboptimal	Opened
CVF-310	Minor	Suboptimal	Opened
CVF-311	Minor	Suboptimal	Opened
CVF-312	Minor	Bad naming	Opened
CVF-313	Minor	Procedural	Opened
CVF-314	Minor	Procedural	Opened
CVF-315	Minor	Bad datatype	Opened
CVF-316	Minor	Documentation	Opened
CVF-317	Minor	Bad datatype	Opened
CVF-318	Minor	Documentation	Opened
CVF-319	Minor	Documentation	Opened
CVF-320	Minor	Documentation	Opened
CVF-321	Minor	Bad naming	Opened
CVF-322	Minor	Bad datatype	Opened
CVF-323	Minor	Documentation	Opened
CVF-324	Minor	Documentation	Opened
CVF-325	Minor	Unclear behavior	Opened
CVF-326	Minor	Bad naming	Opened
CVF-327	Minor	Bad naming	Opened

ID	Severity	Category	Status
CVF-328	Minor	Bad datatype	Opened
CVF-329	Minor	Suboptimal	Opened
CVF-330	Minor	Bad datatype	Opened
CVF-331	Minor	Procedural	Opened
CVF-332	Minor	Bad naming	Opened
CVF-333	Minor	Bad datatype	Opened
CVF-334	Minor	Procedural	Opened
CVF-335	Minor	Documentation	Opened
CVF-336	Minor	Procedural	Opened
CVF-337	Minor	Procedural	Opened
CVF-338	Minor	Bad naming	Opened
CVF-339	Minor	Procedural	Opened
CVF-340	Minor	Bad datatype	Opened
CVF-341	Minor	Suboptimal	Opened
CVF-342	Minor	Procedural	Opened
CVF-343	Minor	Suboptimal	Opened
CVF-344	Minor	Suboptimal	Opened
CVF-345	Minor	Suboptimal	Opened
CVF-346	Minor	Bad datatype	Opened
CVF-347	Minor	Bad datatype	Opened
CVF-348	Minor	Procedural	Opened
CVF-349	Minor	Procedural	Opened
CVF-350	Minor	Bad naming	Opened
CVF-351	Minor	Bad datatype	Opened
CVF-352	Minor	Bad naming	Opened
CVF-353	Minor	Bad datatype	Opened
CVF-354	Minor	Bad datatype	Opened
CVF-355	Minor	Suboptimal	Opened
CVF-356	Minor	Procedural	Opened
CVF-357	Minor	Suboptimal	Opened

ID	Severity	Category	Status
CVF-358	Minor	Bad datatype	Opened

Contents

1 Document properties **24**

2 Introduction **25**
2.1 About ABDK . . . 27
2.2 Disclaimer . . . 27
2.3 Methodology . . . 27

3 Detailed Results **29**
3.1 CVF-1 . . . 29
3.2 CVF-2 . . . 29
3.3 CVF-3 . . . 30
3.4 CVF-4 . . . 30
3.5 CVF-5 . . . 30
3.6 CVF-6 . . . 30
3.7 CVF-7 . . . 31
3.8 CVF-8 . . . 31
3.9 CVF-9 . . . 31
3.10 CVF-10 . . . 31
3.11 CVF-11 . . . 32
3.12 CVF-12 . . . 32
3.13 CVF-13 . . . 32
3.14 CVF-14 . . . 33
3.15 CVF-15 . . . 33
3.16 CVF-16 . . . 34
3.17 CVF-17 . . . 34
3.18 CVF-18 . . . 34
3.19 CVF-19 . . . 35
3.20 CVF-20 . . . 35
3.21 CVF-21 . . . 36
3.22 CVF-22 . . . 36
3.23 CVF-23 . . . 36
3.24 CVF-24 . . . 37
3.25 CVF-25 . . . 37
3.26 CVF-26 . . . 37
3.27 CVF-27 . . . 38
3.28 CVF-28 . . . 38
3.29 CVF-29 . . . 38
3.30 CVF-30 . . . 39
3.31 CVF-31 . . . 39
3.32 CVF-32 . . . 39
3.33 CVF-33 . . . 40
3.34 CVF-34 . . . 41
3.35 CVF-35 . . . 41
3.36 CVF-36 . . . 41
3.37 CVF-37 . . . 42

3.38 CVF-38 . . . 42
3.39 CVF-39 . . . 42
3.40 CVF-40 . . . 43
3.41 CVF-41 . . . 43
3.42 CVF-42 . . . 44
3.43 CVF-43 . . . 44
3.44 CVF-44 . . . 45
3.45 CVF-45 . . . 45
3.46 CVF-46 . . . 45
3.47 CVF-47 . . . 46
3.48 CVF-48 . . . 46
3.49 CVF-49 . . . 46
3.50 CVF-50 . . . 47
3.51 CVF-51 . . . 47
3.52 CVF-52 . . . 47
3.53 CVF-53 . . . 48
3.54 CVF-54 . . . 48
3.55 CVF-55 . . . 48
3.56 CVF-56 . . . 49
3.57 CVF-57 . . . 49
3.58 CVF-58 . . . 49
3.59 CVF-59 . . . 50
3.60 CVF-60 . . . 50
3.61 CVF-61 . . . 51
3.62 CVF-62 . . . 51
3.63 CVF-63 . . . 51
3.64 CVF-64 . . . 52
3.65 CVF-65 . . . 52
3.66 CVF-66 . . . 52
3.67 CVF-67 . . . 53
3.68 CVF-68 . . . 53
3.69 CVF-69 . . . 54
3.70 CVF-70 . . . 54
3.71 CVF-71 . . . 54
3.72 CVF-72 . . . 55
3.73 CVF-73 . . . 55
3.74 CVF-74 . . . 55
3.75 CVF-75 . . . 56
3.76 CVF-76 . . . 56
3.77 CVF-77 . . . 56
3.78 CVF-78 . . . 57
3.79 CVF-79 . . . 57
3.80 CVF-80 . . . 58
3.81 CVF-81 . . . 58
3.82 CVF-82 . . . 58
3.83 CVF-83 . . . 59

3.84 CVF-84 . . . 59
3.85 CVF-85 . . . 60
3.86 CVF-86 . . . 60
3.87 CVF-87 . . . 60
3.88 CVF-88 . . . 61
3.89 CVF-89 . . . 61
3.90 CVF-90 . . . 61
3.91 CVF-91 . . . 62
3.92 CVF-92 . . . 62
3.93 CVF-93 . . . 62
3.94 CVF-94 . . . 63
3.95 CVF-95 . . . 63
3.96 CVF-96 . . . 64
3.97 CVF-97 . . . 64
3.98 CVF-98 . . . 64
3.99 CVF-99 . . . 65
3.100 CVF-100 . . . 65
3.101 CVF-101 . . . 65
3.102 CVF-102 . . . 66
3.103 CVF-103 . . . 66
3.104 CVF-104 . . . 66
3.105 CVF-105 . . . 67
3.106 CVF-106 . . . 67
3.107 CVF-107 . . . 67
3.108 CVF-108 . . . 67
3.109 CVF-109 . . . 68
3.110 CVF-110 . . . 68
3.111 CVF-111 . . . 69
3.112 CVF-112 . . . 70
3.113 CVF-113 . . . 71
3.114 CVF-114 . . . 71
3.115 CVF-115 . . . 72
3.116 CVF-116 . . . 72
3.117 CVF-117 . . . 73
3.118 CVF-118 . . . 73
3.119 CVF-119 . . . 73
3.120 CVF-120 . . . 74
3.121 CVF-121 . . . 74
3.122 CVF-122 . . . 74
3.123 CVF-123 . . . 75
3.124 CVF-124 . . . 75
3.125 CVF-125 . . . 75
3.126 CVF-126 . . . 76
3.127 CVF-127 . . . 76
3.128 CVF-128 . . . 76
3.129 CVF-129 . . . 77

3.130CVF-130 . . . 77
3.131CVF-131 . . . 77
3.132CVF-132 . . . 78
3.133CVF-133 . . . 78
3.134CVF-134 . . . 78
3.135CVF-135 . . . 79
3.136CVF-136 . . . 79
3.137CVF-137 . . . 79
3.138CVF-138 . . . 80
3.139CVF-139 . . . 80
3.140CVF-140 . . . 80
3.141CVF-141 . . . 81
3.142CVF-142 . . . 81
3.143CVF-143 . . . 81
3.144CVF-144 . . . 81
3.145CVF-145 . . . 82
3.146CVF-146 . . . 82
3.147CVF-147 . . . 83
3.148CVF-148 . . . 83
3.149CVF-149 . . . 84
3.150CVF-150 . . . 84
3.151CVF-151 . . . 84
3.152CVF-152 . . . 85
3.153CVF-153 . . . 85
3.154CVF-154 . . . 85
3.155CVF-155 . . . 86
3.156CVF-156 . . . 86
3.157CVF-157 . . . 86
3.158CVF-158 . . . 87
3.159CVF-159 . . . 87
3.160CVF-160 . . . 87
3.161CVF-161 . . . 88
3.162CVF-162 . . . 88
3.163CVF-163 . . . 88
3.164CVF-164 . . . 89
3.165CVF-165 . . . 89
3.166CVF-166 . . . 90
3.167CVF-167 . . . 90
3.168CVF-168 . . . 91
3.169CVF-169 . . . 91
3.170CVF-170 . . . 91
3.171CVF-171 . . . 92
3.172CVF-172 . . . 92
3.173CVF-173 . . . 92
3.174CVF-174 . . . 93
3.175CVF-175 . . . 93

3.176CVF-176 . . . 94
3.177CVF-177 . . . 94
3.178CVF-178 . . . 95
3.179CVF-179 . . . 96
3.180CVF-180 . . . 97
3.181CVF-181 . . . 98
3.182CVF-182 . . . 98
3.183CVF-183 . . . 98
3.184CVF-184 . . . 99
3.185CVF-185 . . . 99
3.186CVF-186 . . . 100
3.187CVF-187 . . . 100
3.188CVF-188 . . . 101
3.189CVF-189 . . . 101
3.190CVF-190 . . . 101
3.191CVF-191 . . . 102
3.192CVF-192 . . . 102
3.193CVF-193 . . . 103
3.194CVF-194 . . . 103
3.195CVF-195 . . . 104
3.196CVF-196 . . . 104
3.197CVF-197 . . . 105
3.198CVF-198 . . . 105
3.199CVF-199 . . . 105
3.200CVF-200 . . . 106
3.201CVF-201 . . . 106
3.202CVF-202 . . . 106
3.203CVF-203 . . . 107
3.204CVF-204 . . . 107
3.205CVF-205 . . . 107
3.206CVF-206 . . . 108
3.207CVF-207 . . . 108
3.208CVF-208 . . . 108
3.209CVF-209 . . . 109
3.210CVF-210 . . . 109
3.211CVF-211 . . . 109
3.212CVF-212 . . . 110
3.213CVF-213 . . . 110
3.214CVF-214 . . . 110
3.215CVF-215 . . . 111
3.216CVF-216 . . . 112
3.217CVF-217 . . . 112
3.218CVF-218 . . . 113
3.219CVF-219 . . . 113
3.220CVF-220 . . . 114
3.221CVF-221 . . . 114

3.222CVF-222 . . . 114
3.223CVF-223 . . . 115
3.224CVF-224 . . . 115
3.225CVF-225 . . . 115
3.226CVF-226 . . . 116
3.227CVF-227 . . . 116
3.228CVF-228 . . . 116
3.229CVF-229 . . . 117
3.230CVF-230 . . . 117
3.231CVF-231 . . . 118
3.232CVF-232 . . . 118
3.233CVF-233 . . . 119
3.234CVF-234 . . . 119
3.235CVF-235 . . . 120
3.236CVF-236 . . . 120
3.237CVF-237 . . . 121
3.238CVF-238 . . . 122
3.239CVF-239 . . . 122
3.240CVF-240 . . . 122
3.241CVF-241 . . . 123
3.242CVF-242 . . . 123
3.243CVF-243 . . . 124
3.244CVF-244 . . . 124
3.245CVF-245 . . . 125
3.246CVF-246 . . . 125
3.247CVF-247 . . . 125
3.248CVF-248 . . . 126
3.249CVF-249 . . . 126
3.250CVF-250 . . . 126
3.251CVF-251 . . . 127
3.252CVF-252 . . . 127
3.253CVF-253 . . . 127
3.254CVF-254 . . . 128
3.255CVF-255 . . . 128
3.256CVF-256 . . . 128
3.257CVF-257 . . . 129
3.258CVF-258 . . . 129
3.259CVF-259 . . . 129
3.260CVF-260 . . . 130
3.261CVF-261 . . . 130
3.262CVF-262 . . . 130
3.263CVF-263 . . . 131
3.264CVF-264 . . . 131
3.265CVF-265 . . . 132
3.266CVF-266 . . . 132
3.267CVF-267 . . . 133

3.268CVF-268 . . . 133
3.269CVF-269 . . . 133
3.270CVF-270 . . . 134
3.271CVF-271 . . . 134
3.272CVF-272 . . . 134
3.273CVF-273 . . . 135
3.274CVF-274 . . . 135
3.275CVF-275 . . . 136
3.276CVF-276 . . . 136
3.277CVF-277 . . . 136
3.278CVF-278 . . . 137
3.279CVF-279 . . . 137
3.280CVF-280 . . . 138
3.281CVF-281 . . . 138
3.282CVF-282 . . . 139
3.283CVF-283 . . . 139
3.284CVF-284 . . . 139
3.285CVF-285 . . . 140
3.286CVF-286 . . . 140
3.287CVF-287 . . . 141
3.288CVF-288 . . . 141
3.289CVF-289 . . . 142
3.290CVF-290 . . . 142
3.291CVF-291 . . . 142
3.292CVF-292 . . . 143
3.293CVF-293 . . . 143
3.294CVF-294 . . . 144
3.295CVF-295 . . . 144
3.296CVF-296 . . . 145
3.297CVF-297 . . . 145
3.298CVF-298 . . . 145
3.299CVF-299 . . . 146
3.300CVF-300 . . . 146
3.301CVF-301 . . . 146
3.302CVF-302 . . . 147
3.303CVF-303 . . . 147
3.304CVF-304 . . . 147
3.305CVF-305 . . . 148
3.306CVF-306 . . . 148
3.307CVF-307 . . . 149
3.308CVF-308 . . . 150
3.309CVF-309 . . . 151
3.310CVF-310 . . . 151
3.311CVF-311 . . . 151
3.312CVF-312 . . . 152
3.313CVF-313 . . . 152

3.314CVF-314 . . . 152
3.315CVF-315 . . . 153
3.316CVF-316 . . . 153
3.317CVF-317 . . . 153
3.318CVF-318 . . . 154
3.319CVF-319 . . . 154
3.320CVF-320 . . . 154
3.321CVF-321 . . . 155
3.322CVF-322 . . . 156
3.323CVF-323 . . . 157
3.324CVF-324 . . . 158
3.325CVF-325 . . . 158
3.326CVF-326 . . . 159
3.327CVF-327 . . . 159
3.328CVF-328 . . . 160
3.329CVF-329 . . . 160
3.330CVF-330 . . . 161
3.331CVF-331 . . . 162
3.332CVF-332 . . . 162
3.333CVF-333 . . . 163
3.334CVF-334 . . . 164
3.335CVF-335 . . . 164
3.336CVF-336 . . . 165
3.337CVF-337 . . . 166
3.338CVF-338 . . . 166
3.339CVF-339 . . . 167
3.340CVF-340 . . . 168
3.341CVF-341 . . . 169
3.342CVF-342 . . . 169
3.343CVF-343 . . . 170
3.344CVF-344 . . . 170
3.345CVF-345 . . . 171
3.346CVF-346 . . . 171
3.347CVF-347 . . . 172
3.348CVF-348 . . . 172
3.349CVF-349 . . . 172
3.350CVF-350 . . . 173
3.351CVF-351 . . . 173
3.352CVF-352 . . . 173
3.353CVF-353 . . . 174
3.354CVF-354 . . . 174
3.355CVF-355 . . . 174
3.356CVF-356 . . . 175
3.357CVF-357 . . . 175
3.358CVF-358 . . . 175

1 Document properties

Version

Version	Date	Author	Description
0.1	January 26, 2022	D. Khovratovich	Initial Draft
0.2	January 26, 2022	D. Khovratovich	Minor revision
1.0	January 26, 2022	D. Khovratovich	Release

Contact

D. Khovratovich

khovratovich@gmail.com

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.
We have reviewed the next files:

- flashloan/interfaces/IFlashLoanReceiver.sol
- flashloan/interfaces/IFlashLoanReceiver.sol
- flashloan/interfaces/IFlashLoanSimpleReceiver.sol
- interfaces/IAaveIncentivesController.sol
- interfaces/IACLManager.sol
- interfaces/IAToken.sol
- interfaces/IChainlinkAggregator.sol
- interfaces/ICreditDelegationToken.sol
- interfaces/IDelegationToken.sol
- interfaces/IERC20WithPermit.sol
- interfaces/IInitializableAToken.sol
- interfaces/IInitializableDebtToken.sol
- interfaces/IPool.sol
- interfaces/IPoolAddressesProvider.sol
- interfaces/IPoolAddressesProviderRegistry.sol
- interfaces/IPoolConfigurator.sol
- interfaces/IPoolDataProvider.sol
- interfaces/IPriceOracle.sol
- interfaces/IPriceOracleGetter.sol
- interfaces/IPriceOracleSentinel.sol
- interfaces/IReserveInterestRateStrategy.sol
- interfaces/IScaledBalanceToken.sol
- interfaces/ISequencerOracle.sol
- interfaces/IStableDebtToken.sol

- interfaces/IVariableDebtToken.sol
- protocol/configuration/ACLManager.sol
- protocol/configuration/PoolAddressesProvider.sol
- protocol/configuration/PoolAddressesProviderRegistry.sol
- protocol/configuration/PriceOracleSentinel.sol
- protocol/libraries/aave-upgradeability/BaseImmutableAdminUpgradeabilityProxy.sol
- protocol/libraries/aave-upgradeability/InitializableImmutableAdminUpgradeabilityProxy.sol
- protocol/libraries/aave-upgradeability/VersionedInitializable.sol
- protocol/libraries/configuration/ReserveConfiguration.sol
- protocol/libraries/configuration/UserConfiguration.sol
- protocol/libraries/helpers/Errors.sol
- protocol/libraries/helpers/Helpers.sol
- protocol/libraries/logic/BorrowLogic.sol
- protocol/libraries/logic/BridgeLogic.sol
- protocol/libraries/logic/ConfiguratorLogic.sol
- protocol/libraries/logic/EModeLogic.sol
- protocol/libraries/logic/FlashLoanLogic.sol
- protocol/libraries/logic/GenericLogic.sol
- protocol/libraries/logic/LiquidationLogic.sol
- protocol/libraries/logic/ReserveLogic.sol
- protocol/libraries/logic/SupplyLogic.sol
- protocol/libraries/logic/ValidationLogic.sol
- protocol/libraries/math/MathUtils.sol
- protocol/libraries/math/PercentageMath.sol
- protocol/libraries/math/WadRayMath.sol
- protocol/libraries/types/ConfiguratorInputTypes.sol
- protocol/libraries/types/DataTypes.sol
- protocol/pool/DefaultReserveInterestRateStrategy.sol

- protocol/pool/Pool.sol
- protocol/pool/PoolConfigurator.sol
- protocol/pool/PoolStorage.sol
- protocol/tokenization/base/DebtTokenBase.sol
- protocol/tokenization/AToken.sol
- protocol/tokenization/DelegationAwareAToken.sol
- protocol/tokenization/IncentivizedERC20.sol
- protocol/tokenization/StableDebtToken.sol
- protocol/tokenization/VariableDebtToken.sol

2.1 About ABDK

ABDK Consulting, established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function. The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

2.2 Disclaimer

Note that the performed audit represents current best practices and smart contract standards which are relevant at the date of publication. After fixing the indicated issues the smart contracts should be re-audited.

2.3 Methodology

The methodology is not a strict formal procedure, but rather a collection of methods and tactics that combined differently and tuned for every particular project, depending on the project structure and and used technologies, as well as on what the client is expecting from the audit. In current audit we use:

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows code best practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places and that their visibility scopes and access levels are relevant. At this phase we understand overall system architecture and how different parts of the code are related to each other.

- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and is done properly. At this phase we understand user roles and permissions, as well as what assets the system ought to protect.

- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check that code actually does what it is supposed to do, that algorithms are optimal and correct, and that proper data types are used. We also check that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

3 Detailed Results

3.1 CVF-1

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** ValidationLogic.sol

Recommendation Should be "^0.8.0". Also relevant for the next files: Pool.sol, PoolAddressesProvider.sol, MathUtils.sol, PoolAddressesProviderRegistry.sol, PriceOracleSentinel.sol, ACLManager.sol, ConfiguratorInputTypes.sol, ReserveLogic.sol, PercentageMath.sol, Helpers.sol, VersionedInitializable.sol, IVariableDebtToken.sol, EModeLogic.sol, IStableDebtToken.sol, ISequencerOracle.sol, DebtTokenBase.sol, IPriceOracleSentinel.sol, IPriceOracle.sol, IPoolAddressesProviderRegistry.sol, IERC20WithPermit.sol, ICreditDelegationToken.sol, IScaledBalanceToken.sol, IChainlinkAggregator.sol, IACLManager.sol, IAaveIncentivesController.sol, IDelegationToken.sol, IFlashLoanSimpleReceiver.sol, IFlashLoanReceiver.sol, VariableDebtToken.sol, DelegationAwareAToken.sol, AToken.sol,IncentivizedERC20.sol, PoolConfigurator.sol, PoolStorage.sol, IReserveInterestRateStrategy.sol, DefaultReserveInterestRateStrategy.sol, SupplyLogic.sol, LiquidationLogic.sol, FlashLoanLogic.sol, DataTypes.sol, ConfiguratorLogic.sol, BridgeLogic.sol, BorrowLogic.sol, WadRayMath.sol, GenericLogic.sol, Errors.sol, UserConfiguration.sol, ReserveConfiguration.sol, InitializableImmutableAdminUpgradeabilityProxy.sol, BaseImmutableAdminUpgradeabilityProxy.sol, IPriceOracleGetter.sol, IPoolDataProvider.sol, IPoolConfigurator.sol, IPoolAddressesProvider.sol, IPool.sol, IInitializableDebtToken.sol, IInitializableAToken.sol.

Listing 1:

```
2 pragma solidity 0.8.7;
```

3.2 CVF-2

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** ValidationLogic.sol

Recommendation "0.95e27" would be more readable.

Listing 2:

```
39 uint256 public constant REBALANCE_UP_USAGE_RATIO_THRESHOLD =
   ↪ 0.95 * 1e27; //usage ratio of 95%
```

3.3 CVF-3

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** ValidationLogic.sol

Recommendation "0.95e18" would be more readable.

Listing 3:

```
40 uint256 public constant
     ↪ MINIMUM_HEALTH_FACTOR_LIQUIDATION_THRESHOLD = 0.95 * 1e18;
```

3.4 CVF-4

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ValidationLogic.sol

Description This should be executed only when 'supplyCap' is not zero.

Listing 4:

```
61 uint256 reserveDecimals = reserveCache.reserveConfiguration.
     ↪ getDecimals();
```

3.5 CVF-5

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ValidationLogic.sol

Recommendation This check should be done at the very beginning of the function.

Listing 5:

```
64 require(amount != 0, Errors.VL_INVALID_AMOUNT);
```

3.6 CVF-6

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** ValidationLogic.sol

Recommendation These checks should be done right after the "getFlags()" call.

Listing 6:

```
65 require(isActive, Errors.VL_NO_ACTIVE_RESERVE);
   require(!isPaused, Errors.VL_RESERVE_PAUSED);
   require(!isFrozen, Errors.VL_RESERVE_FROZEN);
```

3.7 CVF-7

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** ValidationLogic.sol

Recommendation The type of this field could be more specific.

Listing 7:

```
112 address eModePriceSource;
```

3.8 CVF-8

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** ValidationLogic.sol

Recommendation This code should be executed after require statements, just before the "vars.reserveDecimals" field is used.

Listing 8:

```
134 vars.reserveDecimals = params.reserveCache.reserveConfiguration.
    ↪ getDecimals();
```

3.9 CVF-9

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ValidationLogic.sol

Recommendation This check should be done at the very beginning of the functions.

Listing 9:

```
147 require(params.amount != 0, Errors.VL_INVALID_AMOUNT);
```

3.10 CVF-10

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ValidationLogic.sol

Recommendation This code should be executed only when "vars.borrowCap" is not zero.

Listing 10:

```
165 unchecked {
      vars.assetUnit = 10**vars.reserveDecimals;
    }
```

3.11 CVF-11

- **Severity** Minor
- **Status** Opened
- **Category** Overflow/Underflow
- **Source** ValidationLogic.sol

Description Overflow is possible here.

Listing 11:

```
166 vars.assetUnit = 10**vars.reserveDecimals;
```

3.12 CVF-12

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ValidationLogic.sol

Description The expression "10**vars.reserveDecimals" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 12:

```
166 vars.assetUnit = 10**vars.reserveDecimals;

248 vars.amountInBaseCurrency /= 10**vars.reserveDecimals;
```

3.13 CVF-13

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ValidationLogic.sol

Recommendation This divider should be precomputed.

Listing 13:

```
198 10 **
      (params.reserveCache.reserveConfiguration.getDecimals() -
200     ReserveConfiguration.DEBT_CEILING_DECIMALS)
```

3.14 CVF-14

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ValidationLogic.sol

Description This will revert in case the reserve decimals is less that the debt selling decimals.
Recommendation Consider multiplying in such a case.

Listing 14:

```
199 (params.reserveCache.reserveConfiguration.getDecimals() -
200   ReserveConfiguration.DEBT_CEILING_DECIMALS)
```

3.15 CVF-15

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** ValidationLogic.sol

Recommendation Consider using struct literal syntax with named fields rather than with positional field, to improve readability.

Listing 15:

```
226 DataTypes.CalculateUserAccountDataParams(
      params.userConfig,
      params.reservesCount,
      params.userAddress,
230   params.oracle,
      params.userEModeCategory
    )

555   DataTypes.CalculateUserAccountDataParams(
        userConfig,
        reservesCount,
        user,
        oracle,
560     userEModeCategory
      )
```

3.16 CVF-16

- **Severity** Major
- **Status** Opened
- **Category** Unclear behavior
- **Source** ValidationLogic.sol

Description This limit could be easily bypassed by splitting a borrow into several parts.
Recommendation Consider removing this limit.

Listing 16:

```
282 //calculate the max available loan size in stable rate mode as a
    ↪ percentage of the
//available liquidity
uint256 maxLoanSizeStable = vars.availableLiquidity.percentMul(
    ↪ params.maxStableLoanPercent);
```

3.17 CVF-17

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ValidationLogic.sol

Recommendation This check should be done at the very beginning of the function.

Listing 17:

```
314 require(amountSent > 0, Errors.VL_INVALID_AMOUNT);
```

3.18 CVF-18

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** ValidationLogic.sol

Recommendation This check depends on "msg.sender", i.e. on the transaction context, and thus should probably be moved to the calling code to keep this function agnostic to the transaction context.

Listing 18:

```
338 require(
    amountSent != type(uint256).max || msg.sender == onBehalfOf,
340 Errors.VL_NO_EXPLICIT_AMOUNT_TO_REPAY_ON_BEHALF
);
```

3.19 CVF-19

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** ValidationLogic.sol

Recommendation The types of these arguments could be more specific.

Listing 19:

```
405 address reserveAddress,

408 address aTokenAddress

460 address[] memory assets,

541 mapping(address => DataTypes.ReserveData) storage reservesData,
    mapping(uint256 => address) storage reserves,

548 address oracle

591 mapping(address => DataTypes.ReserveData) storage reservesData,
    mapping(uint256 => address) storage reserves,

595 address asset,

598 address oracle,

652 mapping(address => DataTypes.ReserveData) storage reservesData,
    mapping(uint256 => address) storage reserves,
```

3.20 CVF-20

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ValidationLogic.sol

Description This should be executed only when "totalDebt" is not zero.

Listing 20:

```
418 uint256 availableLiquidity = IERC20(reserveAddress).balanceOf(
      ↪ aTokenAddress).wadToRay();
```

3.21 CVF-21

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** ValidationLogic.sol

Recommendation Using constant values in comments is discouraged, as the constant value could be changed in future versions, making the comment inaccurate.

Listing 21:

```
421 //if the liquidity rate is below REBALANCE_UP_THRESHOLD of the
    ↪ max variable APR at 95% usage,
```

3.22 CVF-22

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ValidationLogic.sol

Recommendation This part of the condition should be checked right after the "usageRatio" value is calculated.

Listing 22:

```
430 usageRatio >= REBALANCE_UP_USAGE_RATIO_THRESHOLD &&
```

3.23 CVF-23

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ValidationLogic.sol

Recommendation This check should be done at the very beginning of the function.

Listing 23:

```
450 require(userBalance > 0, Errors.
    ↪ VL_UNDERLYING_BALANCE_NOT_GREATER_THAN_0);
```

3.24 CVF-24

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ValidationLogic.sol

Recommendation It would be more efficient to pass a single array of structs with two fields, rather than two parallel arrays. Such approach would also make the length check unnecessary.

Listing 24:

```
460 address[] memory assets,
    uint256[] memory amounts,
```

3.25 CVF-25

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ValidationLogic.sol

Recommendation The first parts of these checks should be done right after the collateral reserve flags were obtained.

Listing 25:

```
460 address[] memory assets,
    uint256[] memory amounts,

511   vars.collateralReserveActive && vars.principalReserveActive,

515   !vars.collateralReservePaused && !vars.principalReservePaused,
```

3.26 CVF-26

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** ValidationLogic.sol

Recommendation Consider giving descriptive names to the returned values.

Listing 26:

```
549 ) internal view returns (uint256, bool) {
```

3.27 CVF-27

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ValidationLogic.sol

Description This field is not used.
Recommendation Consider removing it.

Listing 27:

```
573 uint256 healthFactor;
```

3.28 CVF-28

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ValidationLogic.sol

Recommendation The "vars.healthFactor" value is not used, no need to assign it.

Listing 28:

```
603 (vars.healthFactor, vars.hasZeroLtvCollateral) =
    ↪ validateHealthFactor(
```

3.29 CVF-29

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ValidationLogic.sol

Description This code should only be executed when "vars.assetLtv" is not zero.

Listing 29:

```
603 (vars.healthFactor, vars.hasZeroLtvCollateral) =
    ↪ validateHealthFactor(
      reservesData,
      reserves,
      eModeCategories,
      userConfig,
      from,
      userEModeCategory,
610   reservesCount,
      oracle
    );
```

3.30 CVF-30

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** ValidationLogic.sol

Recommendation "require (x == 0); require (y == 0); require (z == 0); could be optimized as: require (x | y | z == 0);"

Listing 30:

```
632 require(
      IERC20(reserve.stableDebtTokenAddress).totalSupply() == 0,
      Errors.RL_STABLE_DEBT_NOT_ZERO
    );
    require(
      IERC20(reserve.variableDebtTokenAddress).totalSupply() == 0,
      Errors.RL_VARIABLE_DEBT_SUPPLY_NOT_ZERO
    );
640 require(IERC20(reserve.aTokenAddress).totalSupply() == 0, Errors
      ↪ .RL_ATOKEN_SUPPLY_NOT_ZERO);
```

3.31 CVF-31

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** Pool.sol

Recommendation 0.25e4 would be more readable.

Listing 31:

```
91 _maxStableRateBorrowSizePercent = 2500;
```

3.32 CVF-32

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** Pool.sol

Recommendation 0.0009e4 would be more readable.

Listing 32:

```
92 _flashLoanPremiumTotal = 9;
```

3.33 CVF-33

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** Pool.sol

Recommendation Consider using the struct literal syntax with named fields rather than with positional fields, for readability.

Listing 33:

```
134   DataTypes.ExecuteSupplyParams(asset, amount, onBehalfOf,
        ↪ referralCode)

162   DataTypes.ExecuteSupplyParams(asset, amount, onBehalfOf,
        ↪ referralCode)

178     DataTypes.ExecuteWithdrawParams(
          asset,
180       amount,
          to,
          _reservesCount,
          _addressesProvider.getPriceOracle(),
          _usersEModeCategory[msg.sender]
        )

202   DataTypes.ExecuteBorrowParams(
        asset,
        msg.sender,
        onBehalfOf,
        amount,
        interestRateMode,
        referralCode,
        true,
210     _maxStableRateBorrowSizePercent,
        _reservesCount,
        _addressesProvider.getPriceOracle(),
        _usersEModeCategory[msg.sender],
        _addressesProvider.getPriceOracleSentinel()
      )

232     DataTypes.ExecuteRepayParams(asset, amount, rateMode,
          ↪ onBehalfOf, false)

259   DataTypes.ExecuteRepayParams memory params = DataTypes.
        ↪ ExecuteRepayParams(
```

(... 290, 332, 356, 390, 462, 588, 690)

3.34 CVF-34

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** Pool.sol

Description As signature elements v,r,s are never used separately, it makes sense to unite them into a struct.

Listing 34:

```
145 uint8 permitV,
    bytes32 permitR,
    bytes32 permitS
```

3.35 CVF-35

- **Severity** Major
- **Status** Opened
- **Category** Suboptimal
- **Source** Pool.sol

Recommendation It is possible to reduce the length of an existing array without copying the data: assembly { mstore (reserves, sub (reserveListCount, droppedReservesCount)) }

Listing 35:

```
535 address[] memory undroppedReserves = new address[](
      ↪ reserveListCount - droppedReservesCount);
    for (uint256 i = 0; i < reserveListCount - droppedReservesCount;
      ↪ i++) {
      undroppedReserves[i] = reserves[i];
    }
```

3.36 CVF-36

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** Pool.sol

Description These function should revert in case the asset is not listed within the protocol.

Listing 36:

```
622 function dropReserve(address asset) external override
      ↪ onlyPoolConfigurator {

639 function setConfiguration(address asset, uint256 configuration)
```

3.37 CVF-37

- **Severity** Major
- **Status** Opened
- **Category** Procedural
- **Source** Pool.sol

Description There are no range checks for the arguments.
Recommendation Consider adding appropriate checks.

Listing 37:

```
648 function updateBridgeProtocolFee(uint256 protocolFee) external
      ↪ override onlyPoolConfigurator {

654   uint256 flashLoanPremiumTotal,
      uint256 flashLoanPremiumToProtocol
```

3.38 CVF-38

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** Pool.sol

Description Using the "uint8" type for a loop counter is actually more expensive than using "uint256".

Listing 38:

```
711 for (uint8 i = 0; i <= reservesCount; i++) {
```

3.39 CVF-39

- **Severity** Critical
- **Status** Opened
- **Category** Flaw
- **Source** Pool.sol

Description This loop is supposed to find the first fee slot in the reserves list and put a reserve into it, but actually it doesn't stop after finding the first free slot and puts the reserve into all the free slots.
Recommendation Consider adding "break;" at the end of the conditional statement body.

Listing 39:

```
711 for (uint8 i = 0; i <= reservesCount; i++) {
      if (_reservesList[i] == address(0)) {
        _reserves[asset].id = i;
        _reservesList[i] = asset;
        _reservesCount = reservesCount + 1;
      }
    }
```

3.40 CVF-40

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** PoolAddressesProvider.sol

Description The semantics of keys and values of this mapping in unclear.
Recommendation Consider documenting.

Listing 40:

```
17 mapping(bytes32 => address) private _addresses;
```

3.41 CVF-41

- **Severity** Minor
- **Status** Opened
- **Category** Unclear behavior
- **Source** PoolAddressesProvider.sol

Description This function silently returens zero address on an invalid ID.
Recommendation Consider reverting in such a case.

Listing 41:

```
58 function getAddress(bytes32 id) public view override returns (
   ↪ address) {
```

3.42 CVF-42

- **Severity** Moderate
- **Status** Opened
- **Category** Suboptimal
- **Source** PoolAddressesProvider.sol

Description These functions modify the same mapping while logging different events. It is thus possible to log AddressSet event while setting Price Oracle or ACL Manager, which is unknown to users as these constants are private.
Recommendation Consider checking that special ids can't be set generically.

Listing 42:

```
88  function setPriceOracle(address priceOracle) external override
      ↪ onlyOwner {

99  function setACLManager(address aclManager) external override
      ↪ onlyOwner {

110 function setACLAdmin(address aclAdmin) external override
      ↪ onlyOwner {

116 function setPriceOracleSentinel(address oracleSentinel) external
      ↪  override onlyOwner {

127 function setPoolDataProvider(address dataProvider) external
      ↪ override onlyOwner {
```

3.43 CVF-43

- **Severity** Major
- **Status** Opened
- **Category** Flaw
- **Source** PoolAddressesProvider.sol

Description This code could call a non-proxy contract as a proxy.
Recommendation Consider storing an "isProxy" flag along with an address to know whether this address is actually a proxy or now. The flag and the address could be stored in a single storage slot to save gas.

Listing 43:

```
147 address payable proxyAddress = payable(_addresses[id]);

149 InitializableImmutableAdminUpgradeabilityProxy proxy =
      ↪ InitializableImmutableAdminUpgradeabilityProxy(
150   proxyAddress
    );
```

3.44 CVF-44

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** PoolAddressesProvider.sol

Description Calling the same function on both, initial deployment and upgrade, is error prone, as there is no simple way for the implementation to know whether to initialize an empty storage or upgrade an already populated storage.
Recommendation Consider calling different functions on initial deployment and upgrade.

Listing 44:

```
152 bytes memory params = abi.encodeWithSignature('initialize(
    ↪ address)', address(this));

157   proxy.initialize(newAddress, params);

160   proxy.upgradeToAndCall(newAddress, params);
```

3.45 CVF-45

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** PoolAddressesProvider.sol

Description Passing the caller's address as a call argument seems redundant, as the callee may always obtain this address as "msg.sender".

Listing 45:

```
152 bytes memory params = abi.encodeWithSignature('initialize(
    ↪ address)', address(this));

155   proxy = new InitializableImmutableAdminUpgradeabilityProxy(
      ↪ address(this));
```

3.46 CVF-46

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** PoolAddressesProvider.sol

Description This event is emitted even if nothing actually changed.

Listing 46:

```
166 emit MarketIdSet(marketId);
```

3.47 CVF-47

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** MathUtils.sol

Description These functions depend on the current time.
Recommendation Passing the interval length (i.e. the current time minus the last update time) as an argument instead of just the last update time, would make these functions pure.

Listing 47:

```
23 function calculateLinearInterest(uint256 rate, uint40
     ↪ lastUpdateTimestamp)
```

3.48 CVF-48

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** MathUtils.sol

Description The conversion to the "uint256" type is redundant.

Listing 48:

```
29 uint256 result = rate * (block.timestamp - uint256(
     ↪ lastUpdateTimestamp));

56 uint256 exp = currentTimestamp - uint256(lastUpdateTimestamp);
```

3.49 CVF-49

- **Severity** Minor
- **Status** Opened
- **Category** Overflow/Underflow
- **Source** MathUtils.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while some intermediary calculations overflow.
Recommendation See the following article for details about how this problem could be addressed: https://medium.com/coinmonks/math-in-solidity-part-3-percents-and-proportions-4db014e080b1

Listing 49:

```
29 uint256 result = rate * (block.timestamp - uint256(
     ↪ lastUpdateTimestamp));
30 unchecked {
     result = result / SECONDS_PER_YEAR;
   }
```

3.50 CVF-50

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** MathUtils.sol

Description Exponentiation by squaring is not that expensive. For a time period of about 1 year it would consume less than 3K gas, while the current implementation consumes about 1.5K gas.
Recommendation Consider using exponentiation by squaring as it offers much better precision.

Listing 50:

```
39 * To avoid expensive exponentiation, the calculation is
   ↪ performed using a binomial approximation:
```

3.51 CVF-51

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** MathUtils.sol

Description The function actually doesn't need these two values separately, but only their difference.
Recommendation Consider passing the difference (i.e. the time interval length) as a single argument instead of these two arguments.

Listing 51:

```
52 uint40 lastUpdateTimestamp,
uint256 currentTimestamp
```

3.52 CVF-52

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** MathUtils.sol

Description The rate per second implicitly calculated here compounded for one year will not give the original "rate" value due to compounding interest.
Recommendation Consider passing the rate per second as an argument instead of calculating it inside the function.

Listing 52:

```
71 basePowerTwo = rate.rayMul(rate) / (SECONDS_PER_YEAR *
   ↪ SECONDS_PER_YEAR);
basePowerThree = basePowerTwo.rayMul(rate) / SECONDS_PER_YEAR;
```

3.53 CVF-53

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** MathUtils.sol

Recommendation Right shift would be more efficient.

Listing 53:

```
77 secondTerm /= 2;
```

3.54 CVF-54

- **Severity** Major
- **Status** Opened
- **Category** Suboptimal
- **Source** PoolAddressesProviderRegistry.sol

Description The zero ID is explicitly prohivited in the code, but given in the comment as an example.
Recommendation Consider using different value as an example and explaining that zero ID is prohibited.

Listing 54:

```
12 * @dev Used for indexing purposes of Aave protocol's markets.
     ↪ The id assigned
   *   to a PoolAddressesProvider refers to the market it is
     ↪ connected with, for
   *   example with `0` for the Aave main market and `1` for the
     ↪ next created.

39     require(id != 0, Errors.PAPR_INVALID_ADDRESSES_PROVIDER_ID);
```

3.55 CVF-55

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** PoolAddressesProviderRegistry.sol

Recommendation The key type for this mapping could be more specific.

Listing 55:

```
17 mapping(address => uint256) private _addressesProviders;
```

3.56 CVF-56

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** PoolAddressesProviderRegistry.sol

Recommendation The element type for this list could be more specific.

Listing 56:

```
18 address[] private _addressesProvidersList;
```

3.57 CVF-57

- **Severity** Major
- **Status** Opened
- **Category** Suboptimal
- **Source** PoolAddressesProviderRegistry.sol

Description The size of an array returned by this function is the total number of addresses providers ever registered, not the number of currently active addresses providers, and the slots corresponding to the unregistered addresses providers are filled with zeros. This doesn't scale and could start causing problems when the number of unregistered addresses providers will become large.
Recommendation Consider returning an array without gaps.

Listing 57:

```
21 function getAddressesProvidersList() external view override
   ↪ returns (address[] memory) {
```

3.58 CVF-58

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** PoolAddressesProviderRegistry.sol

Description The variable name is confusing. It is not the maximum length of something, but rather the actual length of the returned array.
Recommendation Consider renaming.

Listing 58:

```
24 uint256 maxLength = addressesProvidersList.length;
```

3.59 CVF-59

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** PoolAddressesProviderRegistry.sol

Description Uniqueness of the IDs of registered addresses providers is not guaranteed. Probably not an issue.

Listing 59:

```
41 _addressesProviders[provider] = id;
```

3.60 CVF-60

- **Severity** Minor
- **Status** Opened
- **Category** Unclear behavior
- **Source** PoolAddressesProviderRegistry.sol

Description This function doesn't delete the address provider from the "_addressesProvidersList" array.
Recommendation Consider doing this. In order to do this efficiently, the contract will need to maintain a mapping from a registered addresses provider address to its index in the "_addressesProvidersList". Then the removal procedure will looks like this: require (_addressesProviders[provider] > 0); delete _addressesProviders[provider]; uint index = _addressesProvidersIndexes[provider]; delete _addressesProvidersIndexes[provider]; uint lastIndex = _addressesProvidersList.length - 1; // No underflow possible here if (index < lastIndex) { address lastProvider = _addressesProvidersList[lastIndex]; _addressesProvidersList[index] = lastProvider;; _addressesProvidersIndexes[lastProvider] = index; } _addressesProvidersList.pop();

Listing 60:

```
47 function unregisterAddressesProvider(address provider) external
   ↪ override onlyOwner {
```

3.61 CVF-61

- **Severity** Minor
- **Status** Opened
- **Category** Unclear behavior
- **Source** PoolAddressesProviderRegistry.sol

Description This function silently returns zero ID for a non-registered addresses provider.
Recommendation Consider reverting in such a case.

Listing 61:

```
54 function getAddressesProviderIdByAddress(address
   ↪ addressesProvider)
```

3.62 CVF-62

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** PoolAddressesProviderRegistry.sol

Description Linear search is expensive.
Recommendation Consider doing it only in case "_addressesProviders [provider]" is zero. Also, linear search will not be needed in case the contract will remove unregistered addresses providers from the "_addressesProvidersList" array.

Listing 62:

```
66 for (uint256 i = 0; i < providersCount; i++) {
```

3.63 CVF-63

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** PriceOracleSentinel.sol

Recommendation These variables should be declared as immutable.

Listing 63:

```
15 IPoolAddressesProvider public _addressesProvider;
ISequencerOracle public _oracle;
uint256 public _gracePeriod;
```

3.64 CVF-64

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** EModeLogic.sol

Recommendation Events are usually named via nouns.

Listing 64:

```
31 event UserEModeSet(address indexed user, uint8 categoryId);
```

3.65 CVF-65

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** EModeLogic.sol

Recommendation The type of these arguments could be more specific.

Listing 65:

```
34 mapping(address => DataTypes.ReserveData) storage reserves,
   mapping(uint256 => address) storage reservesList,

37 mapping(address => uint8) storage usersEModeCategory,
```

3.66 CVF-66

- **Severity** Critical
- **Status** Opened
- **Category** Flaw
- **Source** EModeLogic.sol

Description Health factor validation may depend on eMode category settings such as liquidation threshold, LTV etc. Thus, health factor validation should be performed even if one eMode category ID is changed to another eMode category ID.

Listing 66:

```
53 if (prevCategoryId != 0 && params.categoryId == 0) {
     ValidationLogic.validateHealthFactor(
```

3.67 CVF-67

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ConfiguratorInputTypes.sol

Description This library is redundant as it contains only struct definitions. Solidity allows defining structures at the top level outside any contracts or libraries.
Recommendation Consider removing this library and defining the struct at the top level.

Listing 67:

```
4 library ConfiguratorInputTypes {
```

3.68 CVF-68

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** ConfiguratorInputTypes.sol

Recommendation The types of these fields could be more specific.

Listing 68:

```
6  address aTokenImpl;
   address stableDebtTokenImpl;
   address variableDebtTokenImpl;

11 address underlyingAsset;
   address treasury;
   address incentivesController;

25 address asset;
   address treasury;
   address incentivesController;

35 address asset;
   address incentivesController;
```

3.69 CVF-69

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** ReserveLogic.sol

Recommendation Keeping all "using" statements together at the beginning of the contract would make the code easier to read.

Listing 69:

```
24 using WadRayMath for uint256;
   using PercentageMath for uint256;
   using SafeERC20 for IERC20;

37 using ReserveLogic for DataTypes.ReserveData;
   using ReserveConfiguration for DataTypes.ReserveConfigurationMap
      ↪ ;
```

3.70 CVF-70

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** ReserveLogic.sol

Recommendation Events are usually named via nouns such as "ReserveDataUpdate".

Listing 70:

```
29 event ReserveDataUpdated(
```

3.71 CVF-71

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ReserveLogic.sol

Description The conversion to the "uint40" type is redundant.

Listing 71:

```
55 if (timestamp == uint40(block.timestamp)) {

82 if (timestamp == uint40(block.timestamp)) {
```

3.72 CVF-72

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** ReserveLogic.sol

Description The code below looks like it is always executed, while actually it is executed only when timestamp != block.timestamp.
Recommendation Consider putting the rest of the function into an explicit "else" branch.

Listing 72:

```
58 }

85 }
```

3.73 CVF-73

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ReserveLogic.sol

Description The variable is redundant as it is used only once.
Recommendation Consider removing it and just using an expression instead.

Listing 73:

```
60 uint256 cumulated = MathUtils

87 uint256 cumulated = MathUtils
```

3.74 CVF-74

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ReserveLogic.sol

Description This function basically calculates the following: liquidityIndex * (1 + amount / liquidity)
Recommendation This could be calculated as a single proportion: liquidityIndex * (amount + liquidity) / liquidity Such calculation could be performed in integer numbers, no need for WAD/RAY math.

Listing 74:

```
114 function cumulateToLiquidityIndex(
```

3.75 CVF-75

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** ReserveLogic.sol

Recommendation The types of these arguments could be more specific.

Listing 75:

```
137 address aTokenAddress,
    address stableDebtTokenAddress,
    address variableDebtTokenAddress,
140 address interestRateStrategyAddress
```

3.76 CVF-76

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ReserveLogic.sol

Description This condition could be met even after initialization, as the "init" function doesn't ensure that the "aTokenAddress" argument is not zero.
Recommendation Consider ether adding a require statement to ensure that the "aTokenAddress" argument is not zero, or adding a separate "isInitialized" boolean flag into the "ReserveData" struct to make the initialization check independent from business-level fields.

Listing 76:

```
142 require(reserve.aTokenAddress == address(0), Errors.
    ↪ RL_RESERVE_ALREADY_INITIALIZED);
```

3.77 CVF-77

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** ReserveLogic.sol

Recommendation Consider declaring all the struct before functions to make the code easier to read.

Listing 77:

```
152 struct UpdateInterestRatesLocalVars {

213 struct AccrueToTreasuryLocalVars {
```

3.78 CVF-78

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** ReserveLogic.sol

Recommendation Consider using a struct literal with named fields, rather than with positioned fields. This would make the code easier to read.

Listing 78:

```
186 DataTypes.CalculateInterestRatesParams(
      reserveCache.reserveConfiguration.getUnbackedMintCap() > 0 ?
        ↪ reserve.unbacked : 0,
      liquidityAdded,
      liquidityTaken,
190   reserveCache.nextTotalStableDebt,
      vars.totalVariableDebt,
      reserveCache.nextAvgStableBorrowRate,
      reserveCache.reserveFactor,
      reserveAddress,
      reserveCache.aTokenAddress
    )
```

3.79 CVF-79

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ReserveLogic.sol

Description Event when "vars.amountToMint" is not zero, the value of "var.amountToMint.rayDiv(reserveCache.nextLiquidityIndex)" could be zero. Probably not an issue.

Listing 79:

```
270 if (vars.amountToMint != 0) {

273     Helpers.castUint128((vars.amountToMint.rayDiv(reserveCache.
          ↪ nextLiquidityIndex)));
```

3.80 CVF-80

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ReserveLogic.sol

Description This variable is redundant.
Recommendation Just give a name to the returned value and use it instead.

Listing 80:

```
328 DataTypes.ReserveCache memory reserveCache;
```

3.81 CVF-81

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** PercentageMath.sol

Description No access level specified for these constants, so internal access will be used by default.
Recommendation Consider explicitly specifying an access level.

Listing 81:

```
14 uint256 constant PERCENTAGE_FACTOR = 1e4; //percentage plus two
   ↪ decimals
uint256 constant HALF_PERCENT = PERCENTAGE_FACTOR / 2;
```

3.82 CVF-82

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** PercentageMath.sol

Recommendation This precision is know as "basis point", not percentage.

Listing 82:

```
14 uint256 constant PERCENTAGE_FACTOR = 1e4; //percentage plus two
   ↪ decimals
```

3.83 CVF-83

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** PercentageMath.sol

Description The constant name is misleading. Once one percent is represented as 100, half percent should be represented as 50, while the constant value is 5000.
Recommendation Consider renaming to "HALF_PERCENTAGE_FACTOR".

Listing 83:

```
15 uint256 constant HALF_PERCENT = PERCENTAGE_FACTOR / 2;
```

3.84 CVF-84

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** PercentageMath.sol

Description This check is redundant as Solidity compiler anyway performs overflow checks. Also, this check could detect a phantom overflow, i.e. a situation when the final calculation result would fit into the destination type, but some intermediary calculations overflow. Consider either removing this check or surrounding the actual calculation below with ""unchecked"" block.

Listing 84:

```
29 require(
30   value <= (type(uint256).max - HALF_PERCENT) / percentage,
     Errors.MATH_MULTIPLICATION_OVERFLOW
   );

48 require(
     value <= (type(uint256).max - halfPercentage) /
       ↪ PERCENTAGE_FACTOR,
50   Errors.MATH_MULTIPLICATION_OVERFLOW
   );
```

3.85 CVF-85

- **Severity** Minor
- **Status** Opened
- **Category** Overflow/Underflow
- **Source** PercentageMath.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while some intermediary calculations overflow. See the following article for details about how this problem could be addressed: https://medium.com/coinmonks/math-in-solidity-part-3-percents-and-proportions-4db014e080b1

Listing 85:

```
34 return (value * percentage + HALF_PERCENT) / PERCENTAGE_FACTOR;

53 return (value * PERCENTAGE_FACTOR + halfPercentage) / percentage
   ↪ ;
```

3.86 CVF-86

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** Helpers.sol

Recommendation Consider giving descriptive names to the returned values.

Listing 86:

```
23 returns (uint256, uint256)

41 returns (uint256, uint256)
```

3.87 CVF-87

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** Helpers.sol

Recommendation The conversions to the "IERC20" type wouldn't be necessary in case the "stableDebtTokenAddress" and "variableDebtTokenAddress" fields in the "DataTypes.ReserveData" struct would already have the "IERC20" type.

Listing 87:

```
26 IERC20(reserve.stableDebtTokenAddress).balanceOf(user),
   IERC20(reserve.variableDebtTokenAddress).balanceOf(user)

44 IERC20(reserve.stableDebtTokenAddress).balanceOf(user),
   IERC20(reserve.variableDebtTokenAddress).balanceOf(user)
```

3.88 CVF-88

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** Helpers.sol

Recommendation This function could be simplified and optimized like this: function caseUint128(unt256 input) internal pure returns (uint128 output) { require ((output = uint128 (input)) == input, Errors.HLP_UINT128_OVERFLOW); }

Listing 88:

```
49 function castUint128(uint256 input) internal pure returns (
   ↪ uint128) {
```

3.89 CVF-89

- **Severity** Major
- **Status** Opened
- **Category** Flaw
- **Source** VersionedInitializable.sol

Description This condition allows applying revisions with gaps, i.e. applying the revision 3 on top of the revision 1 bypassing the revision 2. This could be dangerous, as the initialized may not be able to properly upgrade revisions other than the previous one.
Recommendation Consider changing the "revision > lastInitializedRevision" subcondititon to: "revision == lastInitializedRevision + 1".

Listing 89:

```
33 initializing || isConstructor() || revision >
   ↪ lastInitializedRevision,
```

3.90 CVF-90

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** IVariableDebtToken.sol

Recommendation Consider giving descriptive names to the returned values.

Listing 90:

```
38 ) external returns (bool, uint256);
```

3.91 CVF-91

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** IVariableDebtToken.sol

Description In other interfaces, events are usually grouped before functions.
Recommendation Consider using consistent approach for grouping interface members.

Listing 91:

```
46 event Burn(address indexed user, uint256 amount, uint256 index);
```

3.92 CVF-92

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IStableDebtToken.sol

Description The number format an semantics of rate value is unclear.
Recommendation Consider documenting.

Listing 92:

```
31    uint256 newRate,
      uint256 avgStableRate,

50    uint256 avgStableRate,

71    uint256 rate

95  function getAverageStableRate() external view returns (uint256);

102 function getUserStableRate(address user) external view returns (
      ↪ uint256);

115   * @return The average stable rate
```

3.93 CVF-93

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IStableDebtToken.sol

Recommendation Should be "stable" rather than "stale".

Listing 93:

```
65 * @return The average stale borrow rate
```

3.94 CVF-94

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** IStableDebtToken.sol

Recommendation Consider giving descriptive names to the returned values.

Listing 94:

```
75      bool,
        uint256,
        uint256

89  function burn(address user, uint256 amount) external returns (
    ↪ uint256, uint256);

122     uint256,
        uint256,
        uint256,
        uint40

139 function getTotalSupplyAndAvgRate() external view returns (
    ↪ uint256, uint256);
```

3.95 CVF-95

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** ISequencerOracle.sol

Description The semantics of the returned values is unclear.
Recommendation Consider documenting.

Listing 95:

```
10  function latestAnswer() external view returns (bool, uint256);
```

3.96 CVF-96

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IPriceOracle.sol

Recommendation The type of the "asset" arguments could be more specific.

Listing 96:

```
15 function getAssetPrice(address asset) external view returns (
     ↪ uint256);

22 function setAssetPrice(address asset, uint256 price) external;
```

3.97 CVF-97

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IPriceOracle.sol

Description The price format and semantics is unclear.
Recommendation Consider documenting.

Listing 97:

```
15 function getAssetPrice(address asset) external view returns (
     ↪ uint256);

22 function setAssetPrice(address asset, uint256 price) external;
```

3.98 CVF-98

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IPriceOracle.sol

Description The same function is declared in the "IPriceOracleGetter" interface.
Recommendation Consider inheriting from there instead of declaring here again.

Listing 98:

```
15 function getAssetPrice(address asset) external view returns (
     ↪ uint256);
```

3.99 CVF-99

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** IPoolAddressesProviderRegistry.sol

Recommendation Events are usually named via nouns, such as "NewAddressProvider" and "AddressProviderRemoval".

Listing 99:

```
10 event AddressesProviderRegistered(address indexed newAddress);
   event AddressesProviderUnregistered(address indexed newAddress);
```

3.100 CVF-100

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IPoolAddressesProviderRegistry.sol

Description The parameter name "newAddress" looks inappropriate for an address unregistration event.
Recommendation Consider renaming the parameter.

Listing 100:

```
11 event AddressesProviderUnregistered(address indexed newAddress);
```

3.101 CVF-101

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IPoolAddressesProviderRegistry.sol

Recommendation The types of the "newAddress" parameters could be more specific.

Listing 101:

```
10 event AddressesProviderRegistered(address indexed newAddress);
   event AddressesProviderUnregistered(address indexed newAddress);
```

3.102 CVF-102

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IPoolAddressesProviderRegistry.sol

Recommendation The types of arguments and return values referring to address providers could be more specific.

Listing 102:

```
17 function getAddressesProvidersList() external view returns (
     ↪ address[] memory);

24 function getAddressesProviderIdByAddress(address
     ↪ addressesProvider)

34 function registerAddressesProvider(address provider, uint256 id)
     ↪  external;

40 function unregisterAddressesProvider(address provider) external;
```

3.103 CVF-103

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IPoolAddressesProviderRegistry.sol

Recommendation Should be "of a" instead of "on a".

Listing 103:

```
20 * @notice Returns the id on a registered PoolAddressesProvider
```

3.104 CVF-104

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IERC20WithPermit.sol

Description There is no documentation comment for this interface.
Recommendation Consider adding such comment.

Listing 104:

```
6 interface IERC20WithPermit is IERC20 {
```

3.105 CVF-105

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IERC20WithPermit.sol

Description There is no documentation comment for this function.
Recommendation Consider adding such comment.

Listing 105:

```
7 function permit(
```

3.106 CVF-106

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** ICreditDelegationToken.sol

Recommendation Events are usually named via nouns, such as "BorrowAllowanceDelegation".

Listing 106:

```
17 event BorrowAllowanceDelegated(
```

3.107 CVF-107

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** ICreditDelegationToken.sol

Recommendation This parameter should be indexed.

Listing 107:

```
20 address asset,
```

3.108 CVF-108

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** ICreditDelegationToken.sol

Recommendation The type for this parameter could be more specific.

Listing 108:

```
20 address asset,
```

3.109 CVF-109

- **Severity** Minor
- **Status** Opened
- **Category** Unclear behavior
- **Source** IScaledBalanceToken.sol

Description The number format of the index is unclear. Is it integer or fractional with some predefined denominator?
Recommendation Consider explaining.

Listing 109:

```
35 * @return The last index interest was accrued to the user's
     ↪ balance
```

3.110 CVF-110

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IChainlinkAggregator.sol

Description There is no documentation comment for this interface.
Recommendation Consider adding such comment.

Listing 110:

```
4 interface IChainlinkAggregator {
```

3.111 CVF-111

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IChainlinkAggregator.sol

Description There are no documentation comments for these functions.
Recommendation Consider adding such comments.

Listing 111:

```
5  function latestAnswer() external view returns (int256);

7  function latestTimestamp() external view returns (uint256);

9  function latestRound() external view returns (uint256);

11 function getAnswer(uint256 roundId) external view returns (
       ↪ int256);

13 function getTimestamp(uint256 roundId) external view returns (
       ↪ uint256);

15 event AnswerUpdated(int256 indexed current, uint256 indexed
       ↪ roundId, uint256 timestamp);
   event NewRound(uint256 indexed roundId, address indexed
       ↪ startedBy);
```

3.112 CVF-112

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IACLManager.sol

Description Having separate set of functions for each role could probably make the code a bit easier to read, but also makes the bytecode bigger and makes the contract less flexible, as introducing a new role would require changing the API.
Recommendation Consider defining three generics functions: "assignRole", "revokeRole", and "hasRole", accepting the role ID as an argument.

Listing 112:

```
34  function addPoolAdmin(address admin) external;

40  function removePoolAdmin(address admin) external;

47  function isPoolAdmin(address admin) external view returns (bool)
      ↪ ;

53  function addEmergencyAdmin(address admin) external;

59  function removeEmergencyAdmin(address admin) external;

66  function isEmergencyAdmin(address admin) external view returns (
      ↪ bool);

72  function addRiskAdmin(address admin) external;

78  function removeRiskAdmin(address admin) external;

85  function isRiskAdmin(address admin) external view returns (bool)
      ↪ ;

91  function addFlashBorrower(address borrower) external;

97  function removeFlashBorrower(address borrower) external;

104 function isFlashBorrower(address borrower) external view returns
      ↪  (bool);

110 function addBridge(address bridge) external;

116 function removeBridge(address bridge) external;

123 function isBridge(address bridge) external view returns (bool);

129 function addAssetListingAdmin(address admin) external;
```

(... 135, 142)

3.113 CVF-113

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** IAaveIncentivesController.sol

Recommendation Events are usually named via nouns, such as "RewardAccrual", "RewardClaim", and "ClaimerSetup".

Listing 113:

```
15 event RewardsAccrued(address indexed user, uint256 amount);

17 event RewardsClaimed(address indexed user, address indexed to,
     ↪ uint256 amount);

26 event RewardsClaimed(

38 event ClaimerSet(address indexed user, address indexed claimer);
```

3.114 CVF-114

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IAaveIncentivesController.sol

Description Despite the comment, this event is most probably emitted only during "claimRewardsOnBehalf" calls, and "claimRewards" calls emit another event with the same name.
Recommendation Consider fixing the documentation comment for this event and adding a documentation comment for the other "RewardsClaimed" event.

Listing 114:

```
19 /**
20  * @notice Emitted during `claimRewards` and `
      ↪ claimRewardsOnBehalf`
    * @param user The address that accrued rewards
    * @param to The address that will be receiving the rewards
    * @param claimer The address that performed the claim
    * @param amount The amount of rewards
    */
   event RewardsClaimed(
```

3.115 CVF-115

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IAaveIncentivesController.sol

Recommendation Consder using a more specific type for assets.

Listing 115:

```
47  function getAssetData(address asset)

75  function configureAssets(address[] calldata assets, uint256[]
      ↪ calldata emissionsPerSecond)

85    address asset,

96  function getRewardsBalance(address[] calldata assets, address
      ↪ user)

109   address[] calldata assets,

124   address[] calldata assets,

143 function getUserAssetData(address user, address asset) external
      ↪ view returns (uint256);
```

3.116 CVF-116

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** IAaveIncentivesController.sol

Recommendation Consider giving descriptive names to the returned values to improve code readability.

Listing 116:

```
51  uint256,
    uint256,
    uint256
```

3.117 CVF-117

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IAaveIncentivesController.sol

Description The word "whitelist" here is confusing. One could think that there could be several claimers whitelisted for a single user, while this is not the case .
Recommendation Consider changing the word to "Set".

Listing 117:

```
57 * @notice Whitelists an address to claim the rewards on behalf
   ↪ of another address
```

3.118 CVF-118

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IAaveIncentivesController.sol

Recommendation It would be more efficient to pass a single array of structs with two fields instead of two parallel arrays.

Listing 118:

```
75 function configureAssets(address[] calldata assets, uint256[]
   ↪ calldata emissionsPerSecond)
```

3.119 CVF-119

- **Severity** Minor
- **Status** Opened
- **Category** Unclear behavior
- **Source** IAaveIncentivesController.sol

Description It is unclear, whether this function is called before an update, or after it.
Recommendation Consider explaining.

Listing 119:

```
79 * @notice Called by the corresponding asset on any update that
   ↪ affects the rewards distribution
```

3.120 CVF-120

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IAaveIncentivesController.sol

Recommendation Consider making "user" argument to be the first argument to allow syntax like this: user.getRewardsBalance (assets) user.claimRewardsOnBehalf (assets, amount, to).

Listing 120:

```
96  function getRewardsBalance(address[] calldata assets, address
        ↪ user)

123 function claimRewardsOnBehalf(
```

3.121 CVF-121

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** IAaveIncentivesController.sol

Recommendation Consider renaming to "getUserIndex".

Listing 121:

```
143 function getUserAssetData(address user, address asset) external
        ↪ view returns (uint256);
```

3.122 CVF-122

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IAaveIncentivesController.sol

Recommendation The type of the returned value should be more specific.

Listing 122:

```
149 function REWARD_TOKEN() external view returns (address);
```

3.123 CVF-123

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IFlashLoanSimpleReceiver.sol

Description Having a separate interface for simple flash loan receives makes it harder to use the same receiver for both, simple and full-featured flash loans.
Recommendation Consider using the same interface in both cases.

Listing 123:

```
13 interface IFlashLoanSimpleReceiver {
```

3.124 CVF-124

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IFlashLoanSimpleReceiver.sol

Recommendation The type of this argument could be more specific.

Listing 124:

```
26 address asset,
```

3.125 CVF-125

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IFlashLoanSimpleReceiver.sol

Description These functions look redundant.
Recommendation Consider removing them from the interface.

Listing 125:

```
33 function ADDRESSES_PROVIDER() external view returns (
     ↪ IPoolAddressesProvider);

35 function POOL() external view returns (IPool);
```

3.126 CVF-126

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IFlashLoanReceiver.sol

Recommendation Passing a single array of structs with three fields would be more efficient than passing three parallel arrays.

Listing 126:

```
26 address[] calldata assets,
   uint256[] calldata amounts,
   uint256[] calldata premiums,
```

3.127 CVF-127

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IFlashLoanReceiver.sol

Recommendation The type of this argument could be more specific.

Listing 127:

```
26 address[] calldata assets,
```

3.128 CVF-128

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IFlashLoanReceiver.sol

Description These functions look redundant.
Recommendation Consider removing them from the interface.

Listing 128:

```
33 function ADDRESSES_PROVIDER() external view returns (
     ↪ IPoolAddressesProvider);

35 function POOL() external view returns (IPool);
```

3.129 CVF-129

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** VariableDebtToken.sol

Recommendation The type of this variable could be more specific.

Listing 129:

```
27 address internal _underlyingAsset;
```

3.130 CVF-130

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** VariableDebtToken.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

Listing 130:

```
29 constructor(IPool pool) DebtTokenBase(pool) {}
```

3.131 CVF-131

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** VariableDebtToken.sol

Recommendation The type for this argument could be more specific.

Listing 131:

```
33 address underlyingAsset,
```

3.132 CVF-132

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** VariableDebtToken.sol

Recommendation This could be simplified as: uint256 accumulatedInterest = scaledBalance.rayMul(index - _userState[user].additionalData);

Listing 132:

```
101 uint256 accumulatedInterest = scaledBalance.rayMul(index) -
      scaledBalance.rayMul(_userState[user].additionalData);

124 uint256 accumulatedInterest = scaledBalance.rayMul(index) -
      scaledBalance.rayMul(_userState[user].additionalData);
```

3.133 CVF-133

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** VariableDebtToken.sol

Description The expression "amount + accumulatedInterest" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 133:

```
108 emit Transfer(address(0), onBehalfOf, amount +
      ↪ accumulatedInterest);
    emit Mint(user, onBehalfOf, amount + accumulatedInterest, index)
      ↪ ;
```

3.134 CVF-134

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** VariableDebtToken.sol

Description These events duplicate each other .
Recommendation Consider emitting only one event.

Listing 134:

```
108 emit Transfer(address(0), onBehalfOf, amount +
      ↪ accumulatedInterest);
    emit Mint(user, onBehalfOf, amount + accumulatedInterest, index)
      ↪ ;
```

3.135 CVF-135

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** VariableDebtToken.sol

Description The expression "accumulatedInterest - amount" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 135:

```
132 emit Transfer(address(0), user, accumulatedInterest - amount);
    emit Mint(user, user, accumulatedInterest - amount, index);
```

3.136 CVF-136

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** VariableDebtToken.sol

Description These events duplicate each other .
Recommendation Consider emitting only one event.

Listing 136:

```
132 emit Transfer(address(0), user, accumulatedInterest - amount);
    emit Mint(user, user, accumulatedInterest - amount, index);
```

3.137 CVF-137

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** VariableDebtToken.sol

Description The expression "amount - accumulatedInterest" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 137:

```
135 emit Transfer(user, address(0), amount - accumulatedInterest);
    emit Burn(user, amount - accumulatedInterest, index);
```

3.138 CVF-138

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** VariableDebtToken.sol

Description These events duplicate each other .
Recommendation Consider emitting only one event.

Listing 138:

```
135 emit Transfer(user, address(0), amount - accumulatedInterest);
    emit Burn(user, amount - accumulatedInterest, index);
```

3.139 CVF-139

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** VariableDebtToken.sol

Recommendation The return type for this function could be more specific.

Listing 139:

```
175 function UNDERLYING_ASSET_ADDRESS() external view returns (
      ↪ address) {
```

3.140 CVF-140

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** StableDebtToken.sol

Description The semantics of the key and values of this mapping is unclear.
Recommendation Consider documenting.

Listing 140:

```
29 mapping(address => uint40) internal _timestamps;
```

3.141 CVF-141

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** StableDebtToken.sol

Description The semantics of this variable is unclear from its name.
Recommendation Consider documenting.

Listing 141:

```
30 uint40 internal _totalSupplyTimestamp;
```

3.142 CVF-142

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** StableDebtToken.sol

Recommendation The type of this variable could be more specific.

Listing 142:

```
32 address internal _underlyingAsset;
```

3.143 CVF-143

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** StableDebtToken.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

Listing 143:

```
34 constructor(IPool pool) DebtTokenBase(pool) {}
```

3.144 CVF-144

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** StableDebtToken.sol

Recommendation The type of this argument could be more specific.

Listing 144:

```
38 address underlyingAsset,
```

3.145 CVF-145

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** StableDebtToken.sol

Description The chain ID could change after a hard fork.
Recommendation Consider obtaining it anew on every use.

Listing 145:

```
45 uint256 chainId = block.chainid;
```

3.146 CVF-146

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** StableDebtToken.sol

Recommendation Consider giving descriptive names to the returned values, for readability.

Listing 146:

```
129       bool,
130       uint256,
          uint256

184     returns (uint256, uint256)

260       uint256,
          uint256,
          uint256

283       uint256,
          uint256,
          uint256,
          uint40

294   function getTotalSupplyAndAvgRate() external view override
        ↪ returns (uint256, uint256) {
```

3.147 CVF-147

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** StableDebtToken.sol

Description When interest is compounded, the weighted average rate calculated here when applied to the full balance is not equivalent to applying the old rate to the original balance and the new rate to the added balance: Probably not en issue.

Listing 147:

```
149 vars.nextStableRate = (vars.currentStableRate.rayMul(
      ↪ currentBalance.wadToRay()) +
150   vars.amountInRay.rayMul(rate)).rayDiv((currentBalance + amount
        ↪ ).wadToRay());

158 vars.currentAvgStableRate = _avgStableRate = (vars.
      ↪ currentAvgStableRate.rayMul(
    vars.previousSupply.wadToRay()
160 ) + rate.rayMul(vars.amountInRay)).rayDiv(vars.nextSupply.
      ↪ wadToRay());
```

3.148 CVF-148

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** StableDebtToken.sol

Description This event partially duplicates the already emitted "Transfer" event with zero "from" address, however this event's account doesn't include the interest earned.
Recommendation Consider either emitting a single event with full amount, or emitting two events: one for interest earned and another for the amount minted.

Listing 148:

```
165 emit Mint(
      user,
      onBehalfOf,
      amount,
      currentBalance,
170   balanceIncrease,
      vars.nextStableRate,
      vars.currentAvgStableRate,
      vars.nextSupply
    );
```

3.149 CVF-149

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** StableDebtToken.sol

Description The stored total supply value is only a rough estimate of the real sum of all the balances, not only due to rounding errors, but also due to how the average interest rates are calculated. Probably the total supply is not worth calculating at all.

Listing 149:

```
193 // Since the total supply and each single user debt accrue
    ↪ separately,
// there might be accumulation errors so that the last borrower
    ↪ repaying
// mght actually try to repay more than the available debt
    ↪ supply.
// In this case we simply set the total supply and the avg
    ↪ stable rate to 0
```

3.150 CVF-150

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** StableDebtToken.sol

Description The "balanceIncrease - amount" value is available as the "amountToMint" variable here.

Listing 150:

```
228 emit Transfer(address(0), user, balanceIncrease - amount);
```

3.151 CVF-151

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** StableDebtToken.sol

Description These two event duplicate each other.
Recommendation Consider emitting only one event.

Listing 151:

```
228 emit Transfer(address(0), user, balanceIncrease - amount);
    emit Mint(
```

3.152 CVF-152

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** StableDebtToken.sol

Description The "amount - balanceIncrease" value is available as the "amountToBurn" variable here.

Listing 152:

```
242 emit Transfer(address(0), user, amount - balanceIncrease);
```

3.153 CVF-153

- **Severity** Moderate
- **Status** Opened
- **Category** Suboptimal
- **Source** StableDebtToken.sol

Description The first two parameters of the event go in wrong order.

Listing 153:

```
242 emit Transfer(address(0), user, amount - balanceIncrease);
```

3.154 CVF-154

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** StableDebtToken.sol

Description These two event duplicate each other.
Recommendation Consider emitting only one event.

Listing 154:

```
242 emit Transfer(address(0), user, amount - balanceIncrease);
    emit Burn(user, amountToBurn, currentBalance, balanceIncrease,
    ↪ nextAvgStableRate, nextSupply);
```

3.155 CVF-155

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** StableDebtToken.sol

Description The value of "previousPrincipalBalance + balanceIncrease" was already calcualted in the previous like as "balanceOf(user)".
Recommendation Consider reusing.

Listing 155:

```
274 return (previousPrincipalBalance, previousPrincipalBalance +
    ↪ balanceIncrease, balanceIncrease);
```

3.156 CVF-156

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** StableDebtToken.sol

Recommendation The return type of this function could be more specific.

Listing 156:

```
318 function UNDERLYING_ASSET_ADDRESS() external view returns (
    ↪ address) {
```

3.157 CVF-157

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** DelegationAwareAToken.sol

Description These import are not used.
Recommendation Consider removing them.

Listing 157:

```
4 import {Errors} from '../libraries/helpers/Errors.sol';

7 import {IACLManager} from '../../interfaces/IACLManager.sol';
```

3.158 CVF-158

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** DelegationAwareAToken.sol

Recommendation IT is a good practice to put a comment into an empty block to explain why the block is empty.

Listing 158:

```
17 constructor(IPool pool) AToken(pool) {}
```

3.159 CVF-159

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** DelegationAwareAToken.sol

Recommendation This function should log some event.

Listing 159:

```
23 function delegateUnderlyingTo(address delegatee) external
   ↪ onlyPoolAdmin {
```

3.160 CVF-160

- **Severity** Moderate
- **Status** Opened
- **Category** Flaw
- **Source** AToken.sol

Description The domain separator is kept in the storage and is used from there without any additional checks. This means that in case of a hard fork, where the two branches of the original blockchain will have different chain IDs, the smart contract will use the same domain separator in both branched, so the same signature will be valid in both branched. This voids the original idea of why domain separator was initially introduced.
Recommendation Consider calculating the domain separator on every use.

Listing 160:

```
37 bytes32 public DOMAIN_SEPARATOR;
```

3.161 CVF-161

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** AToken.sol

Recommendation The types of these variables could be more specific.

Listing 161:

```
40 address internal _treasury;
   address internal _underlyingAsset;
```

3.162 CVF-162

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** AToken.sol

Recommendation The types of these arguments could be more specific.

Listing 162:

```
61 address treasury,
   address underlyingAsset,
```

3.163 CVF-163

- **Severity** Minor
- **Status** Opened
- **Category** Flaw
- **Source** AToken.sol

Description The chain ID could change after a hard fork.
Recommendation Consider obtaining it anew on every use.

Listing 163:

```
69 uint256 chainId = block.chainid;
```

3.164 CVF-164

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** AToken.sol

Recommendation This could be simplified as: uint256 accumulatedInterest = scaledBalance.rayMul(index - _userState[user].additionalData);

Listing 164:

```
112 uint256 accumulatedInterest = scaledBalance.rayMul(index) -
      scaledBalance.rayMul(_userState[user].additionalData);

141 uint256 accumulatedInterest = scaledBalance.rayMul(index) -
      scaledBalance.rayMul(_userState[user].additionalData);
```

3.165 CVF-165

- **Severity** Minor
- **Status** Opened
- **Category** Flaw
- **Source** AToken.sol

Description Due to rounding errors, the visible change in the total supply and the user's balance after the burning, not necessary equals to the "amount" value, so there could be some inconsistency between visible state change and emitted event. Probably not an issue.

Listing 165:

```
117 _burn(user, Helpers.castUint128(amountScaled));

123 emit Transfer(user, address(0), amount);
```

3.166 CVF-166

- **Severity** Minor
- **Status** Opened
- **Category** Unclear behavior
- **Source** AToken.sol

Description The "Transfer" to zero address event is used to log burning, however here it could actually log minting.
Recommendation Consider emitting a "Transfer" from zero address event when new token are actually minted to the user and a "Transfer" to zero address event when token are actually burned.

Listing 166:

```
123 emit Transfer(user, address(0), amount);
    if (accumulatedInterest > amount) {
      emit Mint(user, accumulatedInterest - amount, index);
    } else {
      emit Burn(user, receiverOfUnderlying, amount -
        ↪ accumulatedInterest, index);
    }
```

3.167 CVF-167

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** AToken.sol

Description These events are confusing, as they include the "amount" value that was already logged.
Recommendation Consider logging like this: emit Mint (user, accumulatedInterest, index); emit Transfer (user, address(0), amount);

Listing 167:

```
125 emit Mint(user, accumulatedInterest - amount, index);

127 emit Burn(user, receiverOfUnderlying, amount -
      ↪ accumulatedInterest, index);
```

3.168 CVF-168

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** AToken.sol

Description This event is confusing, as it includes the "amount" value that was already logged.
Recommendation Consider logging like this: emit Mint (user, accumulatedInterest, index); emit Transfer (address(0), user, amount);

Listing 168:

```
149 emit Mint(user, amount + accumulatedInterest, index);
```

3.169 CVF-169

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** AToken.sol

Description These events basically log the same.
Recommendation Consider leaving only one of them.

Listing 169:

```
168 emit Transfer(address(0), treasury, amount);
    emit Mint(treasury, amount, index);
```

3.170 CVF-170

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source**

Recommendation Consider giving descriptive names to the returned values.

Listing 170:

```
205 returns (uint256, uint256)
```

3.171 CVF-171

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** DebtTokenBase.sol

Description The semantics of the keys for this mapping is unclear.
Recommendation Consider documenting.

Listing 171:

```
21 mapping(address => mapping(address => uint256)) internal
    ↪ _borrowAllowances;
```

3.172 CVF-172

- **Severity** Moderate
- **Status** Opened
- **Category** Flaw
- **Source** DebtTokenBase.sol

Description The domain separator is kept in the storage and is used from there without any additional checks. This means that in case of a hard fork, where the two branches of the original blockchain will have different chain IDs, the smart contract will use the same domain separator in both branched, so the same signature will be valid in both branched. This voids the original idea of why domain separator was initially introduced.
Recommendation Consider calculating the domain separator on every use.

Listing 172:

```
30 bytes32 public DOMAIN_SEPARATOR;
```

3.173 CVF-173

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** DebtTokenBase.sol

Description Including the delegator's address into the signed message is redundant, as the delegator's address is anyway implicitly included into a signature.
Recommendation Consider removing this value from the hash.

Listing 173:

```
82 delegator,
```

3.174 CVF-174

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** DebtTokenBase.sol

Description These functions still occupy bytecode space.
Recommendation Consider removing them completely from the contrac and its base contracts.

Listing 174:

```
125 function transfer(address, uint256) external virtual override
      ↪ returns (bool) {
      revert('TRANSFER_NOT_SUPPORTED');

129 function allowance(address, address) external view virtual
      ↪ override returns (uint256) {
130   revert('ALLOWANCE_NOT_SUPPORTED');

133 function approve(address, uint256) external virtual override
      ↪ returns (bool) {
      revert('APPROVAL_NOT_SUPPORTED');

137 function transferFrom(

142   revert('TRANSFER_NOT_SUPPORTED');

145 function increaseAllowance(address, uint256) external virtual
      ↪ override returns (bool) {
      revert('ALLOWANCE_NOT_SUPPORTED');

149 function decreaseAllowance(address, uint256) external virtual
      ↪ override returns (bool) {
150   revert('ALLOWANCE_NOT_SUPPORTED');
```

3.175 CVF-175

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IncentivizedERC20.sol

Recommendation These variables should be declared as "immutable".

Listing 175:

```
42 string private _name;
   string private _symbol;
   uint8 private _decimals;
```

3.176 CVF-176

- **Severity** Minor
- **Status** Opened
- **Category** Flaw
- **Source** IncentivizedERC20.sol

Description This emits the "Approval" event, while the ERC-20 standard describes the "Approval" event as being emitted by "approve" function.
Recommendation Consider not emitting the "Approval" event from the "transferFrom" function.

Listing 176:

```
132 _approve(sender, _msgSender(), _allowances[sender][_msgSender()]
    ↪ - castAmount);
```

3.177 CVF-177

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** IncentivizedERC20.sol

Description Here underflow protection is used to enforce a business-level constraint. This makes the code harder to read and more error-prone.
Recommendation Consider checking business-level constraints explicitly.

Listing 177:

```
132 _approve(sender, _msgSender(), _allowances[sender][_msgSender()]
    ↪ - castAmount);

169 _userState[sender].balance = oldSenderBalance - amount;

202 _userState[account].balance = oldAccountBalance - amount;
```

3.178 CVF-178

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IncentivizedERC20.sol

Description The name, symbol, and decimals properties for a token are considered immutable by most of the software, including DeFi smart contracts. Changing these properties for an existing token could cause unpredictable consequences.
Recommendation Consider keeping these values immutable.

Listing 178:

```
220 function _setName(string memory newName) internal {

224 function _setSymbol(string memory newSymbol) internal {

228 function _setDecimals(uint8 newDecimals) internal {
```

3.179 CVF-179

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** PoolConfigurator.sol

Recommendation It would be more logical to emit this event from the pool contract.

Listing 179:

```
 86 emit ReserveDropped(asset);

128 emit BorrowingEnabledOnReserve(asset, stableBorrowRateEnabled);

136 emit BorrowingDisabledOnReserve(asset);

178 emit CollateralConfigurationChanged(asset, ltv,
    ↪ liquidationThreshold, liquidationBonus);

187 emit StableRateEnabledOnReserve(asset);

195 emit StableRateDisabledOnReserve(asset);

203 emit ReserveActivated(asset);

213 emit ReserveDeactivated(asset);

221 emit ReserveFrozen(asset);

229 emit ReserveUnfrozen(asset);

240   emit ReservePaused(asset);

242   emit ReserveUnpaused(asset);

255 emit ReserveFactorChanged(asset, reserveFactor);

266 emit DebtCeilingChanged(asset, ceiling);

274 emit BorrowCapChanged(asset, borrowCap);

282 emit SupplyCapChanged(asset, supplyCap);

297 emit LiquidationProtocolFeeChanged(asset, fee);

312 emit EModeCategoryAdded(categoryId, ltv, liquidationThreshold,
    ↪ liquidationBonus, oracle, label);

336 emit EModeAssetCategoryChanged(asset, categoryId);
```

(... 351, 361, 379, 397, 415)

3.180 CVF-180

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** PoolConfigurator.sol

Description This event is emitted event even if nothing actually changed.

Listing 180:

```
128 emit BorrowingEnabledOnReserve(asset, stableBorrowRateEnabled);

136 emit BorrowingDisabledOnReserve(asset);

178 emit CollateralConfigurationChanged(asset, ltv,
      ↪ liquidationThreshold, liquidationBonus);

187 emit StableRateEnabledOnReserve(asset);

195 emit StableRateDisabledOnReserve(asset);

203 emit ReserveActivated(asset);

213 emit ReserveDeactivated(asset);

221 emit ReserveFrozen(asset);

229 emit ReserveUnfrozen(asset);

240   emit ReservePaused(asset);

242   emit ReserveUnpaused(asset);

255 emit ReserveFactorChanged(asset, reserveFactor);

266 emit DebtCeilingChanged(asset, ceiling);

274 emit BorrowCapChanged(asset, borrowCap);

282 emit SupplyCapChanged(asset, supplyCap);

297 emit LiquidationProtocolFeeChanged(asset, fee);

312 emit EModeCategoryAdded(categoryId, ltv, liquidationThreshold,
      ↪ liquidationBonus, oracle, label);

336 emit EModeAssetCategoryChanged(asset, categoryId);

351 emit UnbackedMintCapChanged(asset, unbackedMintCap);
```

(... 361, 379, 397, 415)

3.181 CVF-181

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** PoolConfigurator.sol

Recommendation This check should be done at the very beginning of the functions.

Listing 181:

```
151 require(ltv <= liquidationThreshold, Errors.
    ↪ PC_INVALID_CONFIGURATION);
```

3.182 CVF-182

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** PoolConfigurator.sol

Description There is not range check for the new reserve factor value.
Recommendation Consider adding an appropriate check.

Listing 182:

```
247 function setReserveFactor(address asset, uint256 reserveFactor)
```

3.183 CVF-183

- **Severity** Moderate
- **Status** Opened
- **Category** Unclear behavior
- **Source** PoolConfigurator.sol

Description Should this be? if (ceiling == 0) {

Listing 183:

```
261 if (currentConfig.getDebtCeiling() == 0) {
```

3.184 CVF-184

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** PoolConfigurator.sol

Description The premiums set by these two function are related.
Recommendation Consider merging these functions into one that sets both premiums atomically.

Listing 184:

```
383 function updateFlashloanPremiumTotal(uint256
    ↪ flashloanPremiumTotal)

401 function updateFlashloanPremiumToProtocol(uint256
    ↪ flashloanPremiumToProtocol)
```

3.185 CVF-185

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** PoolStorage.sol

Recommendation The types of these variables could be more specific.

Listing 185:

```
23 mapping(address => DataTypes.ReserveData) internal _reserves;

27 mapping(uint256 => address) internal _reservesList;
```

3.186 CVF-186

- **Severity** Major
- **Status** Opened
- **Category** Suboptimal
- **Source** PoolStorage.sol

Description In many cases, several of these mappings or their values are passed to library functions are storage references. This increases call costs.
Recommendation Consider wrapping the whole protocol storage into a struct that could be passed by references as a whole.

Listing 186:

```
23 mapping(address => DataTypes.ReserveData) internal _reserves;
   mapping(address => DataTypes.UserConfigurationMap) internal
   ↪ _usersConfig;

27 mapping(uint256 => address) internal _reservesList;

39 mapping(uint8 => DataTypes.EModeCategory) _eModeCategories;

41 mapping(address => uint8) _usersEModeCategory;
```

3.187 CVF-187

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** PoolStorage.sol

Description The formats of the values of these variables is unclear.
Recommendation Consider documenting.

Listing 187:

```
31 uint256 internal _maxStableRateBorrowSizePercent;

33 uint256 internal _flashLoanPremiumTotal;

37 uint256 internal _flashLoanPremiumToProtocol;
```

3.188 CVF-188

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** PoolStorage.sol

Description Including the user eMode category into the "DataType.UserConfigurationMap" could make certain scenarios more efficient.

Listing 188:

```
41 mapping(address => uint8) _usersEModeCategory;
```

3.189 CVF-189

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IReserveInterestRateStrategy.sol

Description The format of the returned rates is unclear.
Recommendation Consider documenting.

Listing 189:

```
16 function getBaseVariableBorrowRate() external view returns (
   ↪ uint256);

22 function getMaxVariableBorrowRate() external view returns (
   ↪ uint256);
```

3.190 CVF-190

- **Severity** Moderate
- **Status** Opened
- **Category** Suboptimal
- **Source** IReserveInterestRateStrategy.sol

Description This function is declared as "view" thus it doesn't allow implementing stateful strategies, e.g. strategies that use current protocol state to calculate the momentum of the interest rates, rather than the rates themselves.
Recommendation Consider allowing stateful strategies.

Listing 190:

```
30 view
```

3.191 CVF-191

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IReserveInterestRateStrategy.sol

Description The semantics of the returned values is unclear.
Recommendation Consider given descriptive names to them and/or describing the returned values in the documentation comment.

Listing 191:

```
32  uint256 ,
    uint256 ,
    uint256
```

3.192 CVF-192

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** DefaultReserveInterestRateStrategy.sol

Description There are no range checks for these arguments, while most probably not all the possible values are valid from business point of view.
Recommendation Consider adding appropriate checks.

Listing 192:

```
67  uint256 optimalUtilizationRate ,
    uint256 baseVariableBorrowRate ,
    uint256 variableRateSlope1 ,
70  uint256 variableRateSlope2 ,
    uint256 stableRateSlope1 ,
    uint256 stableRateSlope2 ,
    uint256 baseStableRateOffset ,
    uint256 stableRateExcessOffset ,
    uint256 optimalStableToTotalDebtRatio
```

3.193 CVF-193

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** DefaultReserveInterestRateStrategy.sol

Description The excess utilization rate calculated here may appear below the optimal utilization rate, while the documentation comment above states that it should be above the optimal rate.
Recommendation Consider adding a require statement to prevent incorrect rates to be set.

Listing 193:

```
78 EXCESS_UTILIZATION_RATE = WadRayMath.RAY -
     ↪ optimalUtilizationRate;
```

3.194 CVF-194

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** DefaultReserveInterestRateStrategy.sol

Description These functions wouldn't be necessary in case the corresponding variables would be public.

Listing 194:

```
90  function getVariableRateSlope1() external view returns (uint256)
      ↪ {

94  function getVariableRateSlope2() external view returns (uint256)
      ↪ {

98  function getStableRateSlope1() external view returns (uint256) {

102 function getStableRateSlope2() external view returns (uint256) {

106 function getBaseStableBorrowRate() public view returns (uint256)
      ↪ {

111 function getBaseVariableBorrowRate() external view override
      ↪ returns (uint256) {

116 function getMaxVariableBorrowRate() external view override
      ↪ returns (uint256) {
```

3.195 CVF-195

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** DefaultReserveInterestRateStrategy.sol

Description Consider giving descriptive names to the returned values.
Recommendation Consider giving descriptive names to the returned values.

Listing 195:

```
137 uint256 ,
    uint256 ,
    uint256
```

3.196 CVF-196

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** DefaultReserveInterestRateStrategy.sol

Description The _baseVariableBorrowRate" value is always added to "vars.currentVariableBorrowRate".
Recommendation Consider just initializing the "vars.currentVariableBorrowRate" field with the "_baseVariableBorrwoRate" value.

Listing 196:

```
155 vars.currentVariableBorrowRate = 0;

178   vars.currentVariableBorrowRate =
        _baseVariableBorrowRate +

187   vars.currentVariableBorrowRate =
        _baseVariableBorrowRate +
```

3.197 CVF-197

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** DefaultReserveInterestRateStrategy.sol

Description The former assignment is redundant as the latter always overwrites it.

Listing 197:

```
156 vars.currentStableBorrowRate = 0;

167 vars.currentStableBorrowRate = getBaseStableBorrowRate();
```

3.198 CVF-198

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** DefaultReserveInterestRateStrategy.sol

Recommendation This could be simplified using "+=".

Listing 198:

```
173 vars.currentStableBorrowRate =
      vars.currentStableBorrowRate +

183 vars.currentStableBorrowRate =
      vars.currentStableBorrowRate +
```

3.199 CVF-199

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** DefaultReserveInterestRateStrategy.sol

Description The expression "vars.borrowUtilizationRate.rayDiv(OPTIMAL_UTILIZATION_RATE)" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 199:

```
185 _stableRateSlope1.rayMul(vars.borrowUtilizationRate.rayDiv(
      ↪ OPTIMAL_UTILIZATION_RATE));

189 _variableRateSlope1.rayMul(vars.borrowUtilizationRate).rayDiv(
      ↪ OPTIMAL_UTILIZATION_RATE);
```

3.200 CVF-200

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** SupplyLogic.sol

Description We didn't review this interface.

Listing 200:

```
8 import {IFlashLoanReceiver} from '../../../flashloan/interfaces/
    ↪ IFlashLoanReceiver.sol';
```

3.201 CVF-201

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** SupplyLogic.sol

Description Events are usually named via nouns.
Recommendation Consider renaming.

Listing 201:

```
35 event ReserveUsedAsCollateralEnabled(address indexed reserve,
    ↪ address indexed user);
   event ReserveUsedAsCollateralDisabled(address indexed reserve,
    ↪ address indexed user);
```

3.202 CVF-202

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** SupplyLogic.sol

Recommendation The types of the event parameters that refer to reserves could be more specific.

Listing 202:

```
35 event ReserveUsedAsCollateralEnabled(address indexed reserve,
    ↪ address indexed user);
   event ReserveUsedAsCollateralDisabled(address indexed reserve,
    ↪ address indexed user);
   event Withdraw(address indexed reserve, address indexed user,
    ↪ address indexed to, uint256 amount);

39   address indexed reserve,
```

3.203 CVF-203

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** SupplyLogic.sol

Description Unlike other events in this library, this event doesn't index the "user" parameter.
Recommendation Consider using a consistent indexing strategy.

Listing 203:

```
40 address user,
```

3.204 CVF-204

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** SupplyLogic.sol

Recommendation The types of these argument could be more specific.

Listing 204:

```
47 mapping(address => DataTypes.ReserveData) storage reserves,
   mapping(uint256 => address) storage reservesList,

84 mapping(address => DataTypes.ReserveData) storage reserves,
   mapping(uint256 => address) storage reservesList,

143 mapping(address => DataTypes.ReserveData) storage reserves,
   mapping(uint256 => address) storage reservesList,

188 mapping(address => DataTypes.ReserveData) storage reserves,
   mapping(uint256 => address) storage reservesList,

192 address asset,

195 address priceOracle,
```

3.205 CVF-205

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** SupplyLogic.sol

Recommendation If zero is a valid value for "params.amount", then consider adding "&& params.amount != 0" to this condition.

Listing 205:

```
153 if (params.from != params.to) {
```

3.206 CVF-206

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** SupplyLogic.sol

Recommendation This could be simplified as: if (params.balanceFromBefore == params.amount) {

Listing 206:

```
170 if (params.balanceFromBefore - params.amount == 0) {
```

3.207 CVF-207

- **Severity** Minor
- **Status** Opened
- **Category** Unclear behavior
- **Source** SupplyLogic.sol

Description If zero is actaully a valid value for "params.amount"?

Listing 207:

```
176 if (params.balanceToBefore == 0 && params.amount != 0) {
```

3.208 CVF-208

- **Severity** Major
- **Status** Opened
- **Category** Procedural
- **Source** SupplyLogic.sol

Description It is unclear, whether the condition here guarantees that the reserve is not using as a collateral.
Recommendation Consider adding an explicit check like this: !userConfig.isUsingAsCollateral(reserve.id) && ...

Listing 208:

```
208 !isolationModeActive &&
      (reserveCache.reserveConfiguration.getDebtCeiling() == 0 ||
210     !userConfig.isUsingAsCollateralAny()),
```

3.209 CVF-209

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** SupplyLogic.sol

Recommendation Should be: { else if (userConfig.isUsingAsCollateral(reserve.id)) {

Listing 209:

```
216 } else {
```

3.210 CVF-210

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** LiquidationLogic.sol

Description Events are usually named via nouns.
Recommendation Consider renaming.

Listing 210:

```
37 event ReserveUsedAsCollateralEnabled(address indexed reserve,
     ↪ address indexed user);
   event ReserveUsedAsCollateralDisabled(address indexed reserve,
     ↪ address indexed user);
```

3.211 CVF-211

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** LiquidationLogic.sol

Recommendation The types of the event parameters referring to assets could be more specific.

Listing 211:

```
37 event ReserveUsedAsCollateralEnabled(address indexed reserve,
     ↪ address indexed user);
   event ReserveUsedAsCollateralDisabled(address indexed reserve,
     ↪ address indexed user);

40   address indexed collateralAsset,
     address indexed debtAsset,
```

3.212 CVF-212

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** LiquidationLogic.sol

Description The format of these number is unclear.
Recommendation Consider documenting.

Listing 212:

```
49 uint256 internal constant DEFAULT_LIQUIDATION_CLOSE_FACTOR =
     ↪ 5000;
50 uint256 public constant MAX_LIQUIDATION_CLOSE_FACTOR = 10000;
```

3.213 CVF-213

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** LiquidationLogic.sol

Recommendation 0.95e18 would be more readable.

Listing 213:

```
51 uint256 public constant CLOSE_FACTOR_HF_THRESHOLD = 0.95 * 1e18;
```

3.214 CVF-214

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** LiquidationLogic.sol

Recommendation The types of these arguments could be more specific.

Listing 214:

```
77 mapping(address => DataTypes.ReserveData) storage reserves,
   mapping(address => DataTypes.UserConfigurationMap) storage
     ↪ usersConfig,

302 address collateralAsset,
    address debtAsset,
```

3.215 CVF-215

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** LiquidationLogic.sol

Recommendation Consider using the struct literal syntax with named fields rather than with positional fields, for readability.

Listing 215:

```
101 DataTypes.CalculateUserAccountDataParams(
      userConfig,
      params.reservesCount,
      params.user,
      params.priceOracle,
      params.userEModeCategory
    )

113 DataTypes.ValidateLiquidationCallParams(
      vars.debtReserveCache,
      vars.userStableDebt + vars.userVariableDebt,
      vars.healthFactor,
      params.priceOracleSentinel
    )
```

3.216 CVF-216

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** LiquidationLogic.sol

Description These code chunks differ only in how much variable debt is burned.
Recommendation Consider calculating the amount of variable debt to burn and then burning in a single place.

Listing 216:

```
164 vars.debtReserveCache.nextScaledVariableDebt =
      ↪ IVariableDebtToken(
    vars.debtReserveCache.variableDebtTokenAddress
  ).burn(
      params.user,
      vars.actualDebtToLiquidate,
      vars.debtReserveCache.nextVariableBorrowIndex
170   );

181   vars.debtReserveCache.nextScaledVariableDebt =
        ↪ IVariableDebtToken(
      vars.debtReserveCache.variableDebtTokenAddress
    ).burn(params.user, vars.userVariableDebt, vars.
        ↪ debtReserveCache.nextVariableBorrowIndex);
  }
```

3.217 CVF-217

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** LiquidationLogic.sol

Description Three values are returned while only two are documented.
Recommendation Consider documenting all the returned values and also giving descriptive names to the returned values.

Listing 217:

```
296 * @return The maximum amount that is possible to liquidate given
      ↪ all the liquidation constraints (user balance, close
      ↪ factor)
  * @return The amount to repay with the liquidation

312     uint256,
        uint256,
        uint256
```

3.218 CVF-218

- **Severity** Minor
- **Status** Opened
- **Category** Overflow/Underflow
- **Source** LiquidationLogic.sol

Description Overflow is possible here.

Listing 218:

```
326 vars.collateralAssetUnit = 10**vars.collateralDecimals;
    vars.debtAssetUnit = 10**vars.debtAssetDecimals;
```

3.219 CVF-219

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** LiquidationLogic.sol

Description Theses two values are never used separately but only as one divided by another. **Recommendation** Consider calculating the difference between collateral decimals and debt decimals, as using 10**difference or 10**(-difference) as a multiplier or divider where applicable.

Listing 219:

```
326 vars.collateralAssetUnit = 10**vars.collateralDecimals;
    vars.debtAssetUnit = 10**vars.debtAssetDecimals;
```

3.220 CVF-220

- **Severity** Minor
- **Status** Opened
- **Category** Overflow/Underflow
- **Source** LiquidationLogic.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type but some intermediary calculation overflow.
Recommendation Consider using the "muldiv" function as described here: https://2π.com/21/muldiv/index.html or using some other approach that prevent phantom overflows.

Listing 220:

```
336 ((vars.debtAssetPrice * debtToCover * vars.collateralAssetUnit))
      ↪ /
(vars.collateralPrice * vars.debtAssetUnit);

345 vars.debtAmountNeeded = ((vars.collateralPrice * vars.
      ↪ collateralAmount * vars.debtAssetUnit) /
    (vars.debtAssetPrice * vars.collateralAssetUnit)).percentDiv(
      ↪ liquidationBonus);
```

3.221 CVF-221

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** LiquidationLogic.sol

Recommendation Should be "else return" for readability.

Listing 221:

```
366 return (vars.collateralAmount, vars.debtAmountNeeded, 0);
```

3.222 CVF-222

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** FlashLoanLogic.sol

Description This event should include the interest rate mode. Otherwise it is hard to understand whether the flash loan was repaid instantly or converted to a debt position.

Listing 222:

```
35 event FlashLoan(
```

3.223 CVF-223

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** FlashLoanLogic.sol

Recommendation The type for this parameter could be more specific.

Listing 223:

```
38 address indexed asset,
```

3.224 CVF-224

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** FlashLoanLogic.sol

Recommendation The types for these fields could be more specific.

Listing 224:

```
46 address oracle;
   address oracleSentinel;

49 address currentAsset;
50 address currentATokenAddress;

55 address debtToken;
   address[] aTokenAddresses;
```

3.225 CVF-225

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** FlashLoanLogic.sol

Recommendation The types for these arguments could be more specific.

Listing 225:

```
63 mapping(address => DataTypes.ReserveData) storage reserves,
   mapping(uint256 => address) storage reservesList,
```

3.226 CVF-226

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** FlashLoanLogic.sol

Recommendation This check should be performed before allocating the "aTokenAddresses" and "totalPremium" arrays.

Listing 226:

```
74 ValidationLogic.validateFlashloan(params.assets, params.amounts,
   ↪ reserves);
```

3.227 CVF-227

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** FlashLoanLogic.sol

Description In case "params.isAuthorizedFlashBorrower" is true, this multiplies by zero on all iterations of the loop.
Recommendation Consider optimizing.

Listing 227:

```
83 vars.totalPremiums[vars.i] = params.amounts[vars.i].percentMul(
   ↪ vars.flashloanPremiumTotal);

106 vars.currentPremiumToProtocol = params.amounts[vars.i].
   ↪ percentMul(
   vars.flashloanPremiumToProtocol
);
```

3.228 CVF-228

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** FlashLoanLogic.sol

Description The expression "params.amounts[vars.i]" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 228:

```
83 vars.totalPremiums[vars.i] = params.amounts[vars.i].percentMul(
   ↪ vars.flashloanPremiumTotal);

86   params.amounts[vars.i]
```

3.229 CVF-229

- **Severity** Major
- **Status** Opened
- **Category** Suboptimal
- **Source** FlashLoanLogic.sol

Description The full featured and simple flash loans do call different callback functions, so it would not be possible to use simple flash loan with a callback designed to handle full featured flash loans.
Recommendation Consider calling the same function in both cases, but just passing arrays of size 1.

Listing 229:

```
91  vars.receiver.executeOperation(
      params.assets,
      params.amounts,
      vars.totalPremiums,
      msg.sender,
      params.params
    ),

201 vars.receiver.executeOperation(
      params.asset,
      params.amount,
      vars.totalPremium,
      msg.sender,
      params.params
    ),
```

3.230 CVF-230

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** FlashLoanLogic.sol

Description The expression "vars.totalPremiums[vars.i]" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 230:

```
105 vars.currentAmountPlusPremium = vars.currentAmount + vars.
      ↪ totalPremiums[vars.i];

109 vars.currentPremiumToLP = vars.totalPremiums[vars.i] - vars.
      ↪ currentPremiumToProtocol;
```

3.231 CVF-231

- **Severity** Major
- **Status** Opened
- **Category** Suboptimal
- **Source** FlashLoanLogic.sol

Description So for a particular asset the user has two options, either to repay the flash loan in full or open a debt positions for the full loan. However, in some cases, a user may prefer to repay flash loan partially and open a debt position for the rest.
Recommendation Consider implementing such scenario.

Listing 231:

```
137 } else {
    // If the user chose to not return the funds, the system
       ↪ checks if there is enough collateral and
    // eventually opens a debt position
```

3.232 CVF-232

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** FlashLoanLogic.sol

Recommendation Consider using the struct literal syntax with named fields rather than with positional fields, for readability.

Listing 232:

```
148 DataTypes.ExecuteBorrowParams(
      vars.currentAsset,
150   msg.sender,
      params.onBehalfOf,
      vars.currentAmount,
      params.modes[vars.i],
      params.referralCode,
      false,
      params.maxStableRateBorrowSizePercent,
      params.reservesCount,
      vars.oracle,
      params.userEModeCategory,
160   vars.oracleSentinel
    )
```

3.233 CVF-233

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** FlashLoanLogic.sol

Description Several events are logged for a single flash loan with several assets, and three out of six parameters are the same for theses events.
Recommendation Consider emitting a single event with array parameters, or, in case parameters related to particular assets ought to be indexed, emitting a main event with the parameters related to the whole flash loan, such as receiver, initiator, and referral, and then emitting a series of event with per-reserve parameters, somehow linked to the main event.

Listing 233:

```
164 emit FlashLoan(
      params.receiverAddress,
      msg.sender,
      vars.currentAsset,
      vars.currentAmount,
      vars.totalPremiums[vars.i],
170   params.referralCode
    );
```

3.234 CVF-234

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** DataTypes.sol

Description This library is redundant as it contains only structs and enums.
Recommendation Consider moving the structs and enums definitions to the top level and removing this library.

Listing 234:

```
4 library DataTypes {
```

3.235 CVF-235

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** DataTypes.sol

Description This structure occupies 10 storage slots and wastes 74 bytes due to padding. By reordering some fields, it could be optimized to occupy only 9 slots and waste only 42 bytes: just put the "isolationModeTotalDebt" field after "currentStableBorrowRate", and the "lastUpdateTimestamp" field after "aTokenAddress".

Listing 235:

```
6 struct ReserveData {
```

3.236 CVF-236

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** DataTypes.sol

Description It is "supply" in the comment but "liquidity" in the code.
Recommendation Consider using consistent terminology.

Listing 236:

```
11 //the current supply rate. Expressed in ray
   uint128 currentLiquidityRate;
```

3.237 CVF-237

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** DataTypes.sol

Recommendation The types of these fields could be more specific.

Listing 237:

```
 21 address aTokenAddress;
    address stableDebtTokenAddress;
    address variableDebtTokenAddress;

 25 address interestRateStrategyAddress;

 69 address priceSource;

 95 address aTokenAddress;
    address stableDebtTokenAddress;
    address variableDebtTokenAddress;

105 address collateralAsset;
    address debtAsset;

109 address priceOracle;

111 address priceOracleSentinel;

115 address asset;

122 address asset;

131 address oracle;

133 address priceOracleSentinel;

137 address asset;

145 address asset;

149 address oracle;

155 address oracle;

160 address asset;

167 address oracle;

174 address[] assets;
```

(... 184, 191, 203, 210, 216, 218, 220, 239)

3.238 CVF-238

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** DataTypes.sol

Description The structure of this value is unclear.
Recommendation Consider documenting.

Listing 238:

```
60 uint256 data;
```

3.239 CVF-239

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** DataTypes.sol

Description The field name is confusing, as it specifies not the actual repay amount, but rather the maximum relay amount, and the actual relay amount is the minimum of this value and the current debt amount.
Recommendation Consider renaming to "maxAmount".

Listing 239:

```
138 uint256 amount;
```

3.240 CVF-240

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** DataTypes.sol

Recommendation It would be more efficient to have a single array of structs with three fields, rather than three parallel arrays.

Listing 240:

```
174 address[] assets;
    uint256[] amounts;
    uint256[] modes;
```

3.241 CVF-241

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** DataTypes.sol

Recommendation The type of this field should be "InterestRateMode".

Listing 241:

```
213 uint256 interestRateMode;
```

3.242 CVF-242

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** ConfiguratorLogic.sol

Recommendation Events are usually named via nouns.

Listing 242:

```
22 event ReserveInitialized(

31 event ATokenUpgraded(

38 event StableDebtTokenUpgraded(

45 event VariableDebtTokenUpgraded(
```

3.243 CVF-243

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** ConfiguratorLogic.sol

Recommendation The types of this parameters could be more specific.

Listing 243:

```
23 address indexed asset,
   address indexed aToken,
   address stableDebtToken,
   address variableDebtToken,
   address interestRateStrategyAddress

32 address indexed asset,
   address indexed proxy,
   address indexed implementation

39 address indexed asset,
40 address indexed proxy,
   address indexed implementation

46 address indexed asset,
   address indexed proxy,
   address indexed implementation
```

3.244 CVF-244

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ConfiguratorLogic.sol

Recommendation The type conversion here could probably make the code a bit more readable but does not make much practical sense, as the converted value is passed to the "abi.encodeWithSelector" functions, that treats all contract and interface types as raw addresses.

Listing 244:

```
58 IAaveIncentivesController(input.incentivesController),

71 IAaveIncentivesController(input.incentivesController),

84 IAaveIncentivesController(input.incentivesController),
```

3.245 CVF-245

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ConfiguratorLogic.sol

Description The configuration returned here is guaranteed to be empty.
Recommendation Consider removing this call and just allocating an empty configuration in memory.

Listing 245:

```
100 DataTypes.ReserveConfigurationMap memory currentConfig = pool.
      ↪ getConfiguration(
    input.underlyingAsset
  );
```

3.246 CVF-246

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ConfiguratorLogic.sol

Description These calls are redundant as these flags are guaranteed to be false already.

Listing 246:

```
107 currentConfig.setPaused(false);
    currentConfig.setFrozen(false);
```

3.247 CVF-247

- **Severity** Minor
- **Status** Opened
- **Category** Flaw
- **Source** ConfiguratorLogic.sol

Description Updating the number of decimals for an existing token may have undesired consequences, as most of the software would not be able to handle such change.

Listing 247:

```
133 decimals,

156 decimals,

187 decimals,
```

3.248 CVF-248

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** ConfiguratorLogic.sol

Recommendation The return type for this function should be "InitializableImmutableAdminUpgradeableProxy".

Listing 248:

```
208 returns (address)
```

3.249 CVF-249

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** ConfiguratorLogic.sol

Recommendation The type of this argument should be "InitializableImmutableAdminUpgreadabilityProxy".

Listing 249:

```
220 address proxyAddress,
```

3.250 CVF-250

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** BridgeLogic.sol

Recommendation Events are usually named via nouns.

Listing 250:

```
26 event ReserveUsedAsCollateralEnabled(address indexed reserve,
     ↪ address indexed user);
   event MintUnbacked(

34 event BackUnbacked(address indexed reserve, address indexed
     ↪ backer, uint256 amount, uint256 fee);
```

3.251 CVF-251

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** BridgeLogic.sol

Recommendation The type of "reserve" parameters could be more specific.

Listing 251:

```
26 event ReserveUsedAsCollateralEnabled(address indexed reserve,
     ↪ address indexed user);

28   address indexed reserve,

34 event BackUnbacked(address indexed reserve, address indexed
     ↪ backer, uint256 amount, uint256 fee);
```

3.252 CVF-252

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** BridgeLogic.sol

Description The "user" parameter is indexed in one event but is not indexed in the other.
Recommendation Consider using consistent indexing strategy.

Listing 252:

```
26 event ReserveUsedAsCollateralEnabled(address indexed reserve,
     ↪ address indexed user);

29   address user,
```

3.253 CVF-253

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** BridgeLogic.sol

Recommendation The type of the "asset" arguments could be more specific.

Listing 253:

```
50 address asset,

97 address asset,
```

3.254 CVF-254

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** BridgeLogic.sol

Recommendation This could be simplified as: uint256 unbacked = reserve.unbacked += Helpers.castUint128(amount);

Listing 254:

```
64 uint256 unbacked = reserve.unbacked = reserve.unbacked + Helpers
   ↪ .castUint128(amount);
```

3.255 CVF-255

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** BridgeLogic.sol

Recommendation The first part of this condition should be checked right after the "unbackedMintCap" value is obtained from the reserve cache.

Listing 255:

```
67 unbackedMintCap > 0 && unbacked / (10**reserveDecimals) <
   ↪ unbackedMintCap,
```

3.256 CVF-256

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** BridgeLogic.sol

Recommendation The value "10**reserveDecimals" should be precomputed.

Listing 256:

```
67 unbackedMintCap > 0 && unbacked / (10**reserveDecimals) <
   ↪ unbackedMintCap,
```

3.257 CVF-257

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** BorrowLogic.sol

Recommendation The types of the "reserve" parameters could be more specific.

Listing 257:

```
31   address indexed reserve,

40   address indexed reserve,

46 event RebalanceStableBorrowRate(address indexed reserve, address
   ↪  indexed user);
   event Swap(address indexed reserve, address indexed user,
   ↪ uint256 rateMode);
```

3.258 CVF-258

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** BorrowLogic.sol

Description The "user" parameter is not indexed here, while it is indexed in other similar events.
Recommendation Consider using consistent strategy of indexing event parameters.

Listing 258:

```
32 address user,
```

3.259 CVF-259

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** BorrowLogic.sol

Recommendation The type of this parameter should be "DataTypes.InterestRateMode".

Listing 259:

```
35 uint256 borrowRateMode,
```

3.260 CVF-260

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** BorrowLogic.sol

Description The formal of the borrow rate value is unclear.
Recommendation Consider documenting,

Listing 260:

```
36 uint256 borrowRate,
```

3.261 CVF-261

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** BorrowLogic.sol

Recommendation This event should have the new stable borrow rate as a parameter. Currently the event is almost useless as it tells when the stable rate was rebalanced, but doesn't give a clue how the rate was changed.

Listing 261:

```
46 event RebalanceStableBorrowRate(address indexed reserve, address
   ↪ indexed user);
```

3.262 CVF-262

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** BorrowLogic.sol

Description The event name is confusing. The word "swap" is associated with exchanging one asset for another, not changing the interest rate mode to a loan.
Recommendation Consider renaming into "InterestRateModeChange", or "InterestRateModeFlip", or something like this.

Listing 262:

```
47 event Swap(address indexed reserve, address indexed user,
   ↪ uint256 rateMode);
```

3.263 CVF-263

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** BorrowLogic.sol

Recommendation The types of these arguments could be made more specific.

Listing 263:

```
50  mapping(address => DataTypes.ReserveData) storage reserves,
    mapping(uint256 => address) storage reservesList,

149 mapping(address => DataTypes.ReserveData) storage reserves,
150 mapping(uint256 => address) storage reservesList,

240 address asset,

274 address asset,
```

3.264 CVF-264

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** BorrowLogic.sol

Recommendation Consider using struct literal syntax with named field rather than with positional field, to improve readability.

Listing 264:

```
71  DataTypes.ValidateBorrowParams(
      reserveCache,
      userConfig,
      params.asset,
      params.onBehalfOf,
      params.amount,
      params.interestRateMode,
      params.maxStableRateBorrowSizePercent,
      params.reservesCount,
80    params.oracle,
      params.userEModeCategory,
      params.priceOracleSentinel,
      isolationModeActive,
      isolationModeCollateralAddress,
      isolationModeDebtCeiling
    )
```

3.265 CVF-265

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** BorrowLogic.sol

Description The expression "DataType.InterestRateMode(params.interestRateMode) == DataTypes.InteerstRateMode.STABLE" is calcualted twice.
Recommendation Consider calculating once and reusing.

Listing 265:

```
92  if (DataTypes.InterestRateMode(params.interestRateMode) ==
        ↪ DataTypes.InterestRateMode.STABLE) {

141   DataTypes.InterestRateMode(params.interestRateMode) ==
          ↪ DataTypes.InterestRateMode.STABLE
```

3.266 CVF-266

- **Severity** Moderate
- **Status** Opened
- **Category** Unclear behavior
- **Source** BorrowLogic.sol

Description These "else" branched will be executed not only when interest rate mode is VARIABLE, but also on any interest rate mode other than STABLE, i.e. on NONE interest rate mode.
Recommendation Consider changing to "else if (interestRateMode == InterestRateMode.VARIABLE) { ... } else revert ();

Listing 266:

```
105 } else {

185 } else {

302 } else {
```

3.267 CVF-267

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** BorrowLogic.sol

Description Obtaining user debt before updating reserve state consumes extra gas on applying interest twice.
Recommendation Consider updating reserve state before obtaining the user debt.

Listing 267:

```
156 (uint256 stableDebt, uint256 variableDebt) = Helpers.
    ↪ getUserCurrentDebt(
  params.onBehalfOf,
  reserve
);

162 reserve.updateState(reserveCache);
```

3.268 CVF-268

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** BorrowLogic.sol

Recommendation This value should be precomputed for the reserve.

Listing 268:

```
206 10 **
  (reserveCache.reserveConfiguration.getDecimals() -
    ReserveConfiguration.DEBT_CEILING_DECIMALS)
```

3.269 CVF-269

- **Severity** Moderate
- **Status** Opened
- **Category** Unclear behavior
- **Source** BorrowLogic.sol

Description This will revert in case the reserve decimals number is less than the debt ceiling decimals number.
Recommendation Consider multiplying instead of dividing in such a case.

Listing 269:

```
207 (reserveCache.reserveConfiguration.getDecimals() -
    ReserveConfiguration.DEBT_CEILING_DECIMALS)
```

3.270 CVF-270

- **Severity** Moderate
- **Status** Opened
- **Category** Unclear behavior
- **Source** BorrowLogic.sol

Description Unlike other functions in this contract, this function validates operation before updating the reserve state. Even if for this particular operation the validation logic doesn't depend on any state that could change during an update, it is better not to rely on such knowledge.

Listing 270:

```
249 ValidationLogic.validateRebalanceStableBorrowRate(
250   reserve,
      reserveCache,
      asset,
      stableDebtToken,
      variableDebtToken,
      reserveCache.aTokenAddress
    );

258 reserve.updateState(reserveCache);
```

3.271 CVF-271

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** WadRayMath.sol

Description A public constant in a library looks odd. Also, a similar constant "RAY" is declared as internal.
Recommendation Consider making this constant internal as well.

Listing 271:

```
16 uint256 public constant RAY = 1e27;
```

3.272 CVF-272

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** WadRayMath.sol

Recommendation Consider defining the value for this constant as "RAY / WAD" for readability.

Listing 272:

```
19 uint256 internal constant WAD_RAY_RATIO = 1e9;
```

3.273 CVF-273

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** WadRayMath.sol

Description These functions are redundant as corresponding constants have the same access level and could be used instead.

Listing 273:

```
25 function wad() internal pure returns (uint256) {

32 function halfRay() internal pure returns (uint256) {

39 function halfWad() internal pure returns (uint256) {
```

3.274 CVF-274

- **Severity** Minor
- **Status** Opened
- **Category** Overflow/Underflow
- **Source** WadRayMath.sol

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while some intermediary calculations overflow. See the following article for details about how this problem could be addressed: https://medium.com/coinmonks/math-in-solidity-part-3-percents-and-proportions-4db014e080b1

Listing 274:

```
57 return (a * b + HALF_WAD) / WAD;

73 return (a * WAD + halfB) / b;

91 return (a * b + HALF_RAY) / RAY;

107 return (a * RAY + halfB) / b;
```

3.275 CVF-275

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** WadRayMath.sol

Description "In theory, RAY to WAD conversion should never overflow.
Recommendation Consider implementing the conversion like this: result = a / WAD_RAY_RATIO; if (a % WAD_RAY_RATIO != 0) result += 1;"

Listing 275:

```
120 require(result >= halfRatio, Errors.MATH_ADDITION_OVERFLOW);
```

3.276 CVF-276

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** GenericLogic.sol

Description This field is redundant. It is used only once, right after setting.

Listing 276:

```
44 uint256 normalizedDebt;
```

3.277 CVF-277

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** GenericLogic.sol

Description This argument is not documented.
Recommendation Consider documenting it.

Listing 277:

```
71 mapping(uint8 => DataTypes.EModeCategory) storage
   ↪ eModeCategories,
```

3.278 CVF-278

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** GenericLogic.sol

Recommendation Consider giving names to the returned values.

Listing 278:

```
77  uint256 ,
    uint256 ,
    uint256 ,
80  uint256 ,
    uint256 ,
    bool
```

3.279 CVF-279

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** GenericLogic.sol

Description The expression "eModeCategories[params.userEModeCategory]" is calculated three times.
Recommendation Consider calculating once and reusing.

Listing 279:

```
92  vars.eModePriceSource = eModeCategories[params.userEModeCategory
      ↪ ].priceSource;
    vars.eModeLtv = eModeCategories[params.userEModeCategory].ltv;
    vars.eModeLiqThreshold = eModeCategories[params.
      ↪ userEModeCategory].liquidationThreshold;
```

3.280 CVF-280

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** GenericLogic.sol

Description "This basically iterates over set bits in a bit mask checking one bit per iteration. This approach is inefficient, especially when only a few bits are set.
Recommendation Consider checking several bits per iteration like this: function iterateBits (uint x) public { uint index = 0; while (x != 0) { if (x & 0xFFFF == 0) { x >>= 16; index += 16; } if (x & 0xFF == 0) { x >>= 8; index += 8; } if (x & 0xF == 0) { x >>= 4; index += 4; } if (x & 0x3 == 0) { x >>= 2; index += 2; } if (x & 0x1 != 0) { // Process index of a set bit } x >>= 1; index += 1; } } "

Listing 280:

```
103 while (vars.i < params.reservesCount) {
      if (!params.userConfig.isUsingAsCollateralOrBorrowing(vars.i))
        ↪ {
```

3.281 CVF-281

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** GenericLogic.sol

Recommendation Consider using a "for" loop instead of "while" to remove this code duplication.

Listing 281:

```
106   ++vars.i;

115   ++vars.i;

193 ++vars.i;
```

3.282 CVF-282

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** GenericLogic.sol

Description This checks every listed reserve for whether this reserve is still active for the user. Such approach is suboptimal.
Recommendation Consider maintaining a bit mask of active reserves and applying it to the user configuration bit mask via bitwise AND before iterating through user configuration bits.

Listing 282:

```
111 vars.currentReserveAddress = reserves[vars.i];

113 if (vars.currentReserveAddress == address(0)) {
```

3.283 CVF-283

- **Severity** Minor
- **Status** Opened
- **Category** Overflow/Underflow
- **Source** GenericLogic.sol

Description Overflow is possible here.

Listing 283:

```
132 vars.assetUnit = 10**vars.decimals;
```

3.284 CVF-284

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** GenericLogic.sol

Recommendation This value should be calculated once per asset, not every time the asset is processed.

Listing 284:

```
132 vars.assetUnit = 10**vars.decimals;
```

3.285 CVF-285

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** GenericLogic.sol

Recommendation This assignment could be simplified by using "+=".

Listing 285:

```
149 vars.totalCollateralInBaseCurrency =
150   vars.totalCollateralInBaseCurrency +

165 vars.avgLiquidationThreshold =
      vars.avgLiquidationThreshold +
```

3.286 CVF-286

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** GenericLogic.sol

Recommendation "This code could be simplified as: if (vars.ltv > 0) vars.avgLtv += ...;"

Listing 286:

```
155 vars.avgLtv = vars.ltv > 0
      ? vars.avgLtv +
        vars.userBalanceInBaseCurrency *
        (
          (params.userEModeCategory == 0 || vars.eModeAssetCategory
            ↪ != params.userEModeCategory)
160         ? vars.ltv
            : vars.eModeLtv
        )
      : vars.avgLtv;
```

3.287 CVF-287

- **Severity** Critical
- **Status** Opened
- **Category** Flaw
- **Source** GenericLogic.sol

Description In case the "vars.ltv" value is zero, but vars.eModeLtv value is non-zero, the "vars.eModeLtv" value will not be counted event if params.userEModeCAteory == vars.eModeAssetCategory.

Listing 287:

```
155 vars.avgLtv = vars.ltv > 0
      ? vars.avgLtv +
        vars.userBalanceInBaseCurrency *
        (
          (params.userEModeCategory == 0 || vars.eModeAssetCategory
            ↪ != params.userEModeCategory)
160         ? vars.ltv
            : vars.eModeLtv
        )
      : vars.avgLtv;
```

3.288 CVF-288

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** GenericLogic.sol

Description The expression "params.userEModeCategory == 0 || vars.eModeAssetCategory != params.userEModeCategory" is calcualted twice.
Recommendation Consider calculating once and reusing.

Listing 288:

```
159   (params.userEModeCategory == 0 || vars.eModeAssetCategory !=
        ↪ params.userEModeCategory)

169 (params.userEModeCategory == 0 || vars.eModeAssetCategory !=
      ↪ params.userEModeCategory)
```

3.289 CVF-289

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** GenericLogic.sol

Recommendation Should be "<=".

Listing 289:

```
236 if (availableBorrowsInBaseCurrency < totalDebtInBaseCurrency) {
```

3.290 CVF-290

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** GenericLogic.sol

Recommendation This line could be simplified using "-=" operator.

Listing 290:

```
240 availableBorrowsInBaseCurrency = availableBorrowsInBaseCurrency
    ↪ - totalDebtInBaseCurrency;
```

3.291 CVF-291

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** Errors.sol

Recommendation Consider assigning a range of numeric codes for each group of the error messages, such as 100-199 for validation logic, 200-299 for math libraries, 300-399 for common errors between tokens, etc. This will allow to keep codes form the same group together even when new codes are added.

Listing 291:

```
 9 *  - VL = ValidationLogic
10 *  - MATH = Math libraries
   *  - CT = Common errors between tokens (AToken,
   ↪ VariableDebtToken and StableDebtToken)
   *  - AT = AToken
   *  - SDT = StableDebtToken
   *  - VDT = VariableDebtToken
   *  - P = Pool
   *  - PAPR = PoolAddressesProviderRegistry
   *  - PC = PoolConfiguration
   *  - RL = ReserveLogic
   *  - P = Pausable
```

3.292 CVF-292

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** Errors.sol

Description These error codes are listed out order making the code more error-prone.
Recommendation Consider reordering.

Listing 292:

```
23 string public constant CALLER_NOT_POOL_ADMIN = '33'; // 'The
     ↪ caller must be the pool admin'

64 string public constant PC_INVALID_CONFIGURATION = '75'; // '
     ↪ Invalid risk parameters for the reserve'
   string public constant PC_CALLER_NOT_EMERGENCY_ADMIN = '76'; //
     ↪ 'The caller must be the emergency admin'
```

3.293 CVF-293

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** UserConfiguration.sol

Recommendation This code could be simplified and optimized like the following: uint256 bit = 1 « (reserveIndex « 1); if (borrowing) self.data |= bit; else self.data &= ~bit;

Listing 293:

```
34 self.data =
     (self.data & ~(1 << (reserveIndex * 2))) |
     (uint256(borrowing ? 1 : 0) << (reserveIndex * 2));
```

3.294 CVF-294

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** UserConfiguration.sol

Recommendation Left shift would be more efficient than multiplication.

Listing 294:

```
35  (self.data & ~(1 << (reserveIndex * 2))) |
    (uint256(borrowing ? 1 : 0) << (reserveIndex * 2));

54  (self.data & ~(1 << (reserveIndex * 2 + 1))) |
    (uint256(usingAsCollateral ? 1 : 0) << (reserveIndex * 2 + 1))
      ↪ ;

71  return (self.data >> (reserveIndex * 2)) & 3 != 0;

88  return (self.data >> (reserveIndex * 2)) & 1 != 0;

105 return (self.data >> (reserveIndex * 2 + 1)) & 1 != 0;
```

3.295 CVF-295

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** UserConfiguration.sol

Description The expression "reserveIndex * 2" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 295:

```
35 (self.data & ~(1 << (reserveIndex * 2))) |
   (uint256(borrowing ? 1 : 0) << (reserveIndex * 2));
```

3.296 CVF-296

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** UserConfiguration.sol

Recommendation This code could be simplified and optimized like the following: uint256 bit = 1 « ((reserveIndex « 1) + 1); if (usingAsCollateral) self.data |= bit; else self.data &= ~bit;

Listing 296:

```
53 self.data =
     (self.data & ~(1 << (reserveIndex * 2 + 1))) |
     (uint256(usingAsCollateral ? 1 : 0) << (reserveIndex * 2 + 1))
        ↪ ;
```

3.297 CVF-297

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** UserConfiguration.sol

Description The expression "reserveIndex * 2 + 1" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 297:

```
54 (self.data & ~(1 << (reserveIndex * 2 + 1))) |
   (uint256(usingAsCollateral ? 1 : 0) << (reserveIndex * 2 + 1));
```

3.298 CVF-298

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** UserConfiguration.sol

Recommendation "This line could be optimized as: return (self.data » (reserveIndex * 2)) & 3 != 0; or even as: return (self.data » (reserveIndex « 1)) & 3 != 0;"

Listing 298:

```
105 return (self.data >> (reserveIndex * 2 + 1)) & 1 != 0;
```

3.299 CVF-299

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** UserConfiguration.sol

Recommendation Should be "if and only if a number is ...", otherwise the comment doesn't fully explain the trick.

Listing 299:

```
111 * @dev this uses a simple trick - if a number is a power of two
    ↪ (only one bit set) then n & (n - 1) == 0
```

3.300 CVF-300

- **Severity** Critical
- **Status** Opened
- **Category** Flaw
- **Source** UserConfiguration.sol

Description In case "collateralData" is zero, this will revert rather than return false.
Recommendation Consider wrapping in "unchecked" block.

Listing 300:

```
121 return collateralData & (collateralData - 1) == 0;
```

3.301 CVF-301

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** UserConfiguration.sol

Recommendation Consider giving descriptive names to the returned values.

Listing 301:

```
172 bool,
    address,
    uint256
```

3.302 CVF-302

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** UserConfiguration.sol

Description These lines are needed only because of a critical bug in "isUsingAsCollateralOne" function.
Recommendation Consider removing these lines after fixing the bug.

Listing 302:

```
177 if (!isUsingAsCollateralAny(self)) {
      return (false, address(0), 0);
    }
```

3.303 CVF-303

- **Severity** Minor
- **Status** Opened
- **Category** Readability
- **Source** UserConfiguration.sol

Recommendation Should be "else if" for readability.

Listing 303:

```
180 if (isUsingAsCollateralOne(self)) {
```

3.304 CVF-304

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** UserConfiguration.sol

Recommendation This variable is redundant. Just give a name to the returned value and use it instead.

Listing 304:

```
205 uint256 id;
```

3.305 CVF-305

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** UserConfiguration.sol

Description "Linear search is suboptimal.
Recommendation Consider using binary search like this: if (firstCollateralPosition » 128 != 0) {firstCollateralPosition »= 128; id += 128; } if (firstCollateralPosition » 64 != 0) {firstCollateralPosition »= 64; id += 64; } if (firstCollateralPosition » 32 != 0) {firstCollateralPosition »= 32; id += 32; } if (firstCollateralPosition » 16 != 0) {firstCollateralPosition »= 16; id += 16; } if (firstCollateralPosition » 8 != 0) {firstCollateralPosition »= 8; id += 8; } if (firstCollateralPosition » 4 != 0) {firstCollateralPosition »= 4; id += 4; } if (firstCollateralPosition » 2 != 0) {firstCollateralPosition »= 2; id += 2; }"

Listing 305:

```
207 while ((firstCollateralPosition >>= 2) > 0) {
      id += 2;
    }
```

3.306 CVF-306

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** UserConfiguration.sol

Description The division here wouldn't be necessary in case the "i" variable would be increased by one rather than by two per loop iteration.

Listing 306:

```
210 return id / 2;
```

3.307 CVF-307

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** ReserveConfiguration.sol

Description No access level specified for these constants, so internal access will be used by default.
Recommendation Consider explicitly specifying an access level.

Listing 307:

```
13 uint256 constant LTV_MASK =                       0
     ↪ xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000
     ↪ ; // prettier-ignore
   uint256 constant LIQUIDATION_THRESHOLD_MASK =     0
     ↪ xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFF
     ↪ ; // prettier-ignore
   uint256 constant LIQUIDATION_BONUS_MASK =         0
     ↪ xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFF
     ↪ ; // prettier-ignore
   uint256 constant DECIMALS_MASK =                  0
     ↪ xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00FFFFFFFFFFFF
     ↪ ; // prettier-ignore
   uint256 constant ACTIVE_MASK =                    0
     ↪ xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFF
     ↪ ; // prettier-ignore
   uint256 constant FROZEN_MASK =                    0
     ↪ xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDFFFFFFFFFFFFFF
     ↪ ; // prettier-ignore
   uint256 constant BORROWING_MASK =                 0
     ↪ xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBFFFFFFFFFFFFFF
     ↪ ; // prettier-ignore
```

(... 20, 31, 30, 40, 48)

3.308 CVF-308

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ReserveConfiguration.sol

Description These masks are used as is in setters and in inverted form in getters. However, the values of these constants already look inverted.
Recommendation Consider inverting the constants values, so the constants will be used as in in getters and in inverted for in setters.

Listing 308:

```
13 uint256 constant LTV_MASK =                       0
   ↪ xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000
   ↪ ; // prettier-ignore
   uint256 constant LIQUIDATION_THRESHOLD_MASK =     0
   ↪ xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFF
   ↪ ; // prettier-ignore
   uint256 constant LIQUIDATION_BONUS_MASK =         0
   ↪ xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFF
   ↪ ; // prettier-ignore
   uint256 constant DECIMALS_MASK =                  0
   ↪ xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00FFFFFFFFFF
   ↪ ; // prettier-ignore
   uint256 constant ACTIVE_MASK =                    0
   ↪ xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFF
   ↪ ; // prettier-ignore
   uint256 constant FROZEN_MASK =                    0
   ↪ xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDFFFFFFFFFFFF
   ↪ ; // prettier-ignore
   uint256 constant BORROWING_MASK =                 0
   ↪ xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBFFFFFFFFFFFF
   ↪ ; // prettier-ignore
20 (... 20)
```

3.309 CVF-309

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ReserveConfiguration.sol

Recommendation It would be better to specify these numbers in hexadecimal form.

Listing 309:

```
48  uint256 constant MAX_VALID_LTV = 65535;
    uint256 constant MAX_VALID_LIQUIDATION_THRESHOLD = 65535;
50  uint256 constant MAX_VALID_LIQUIDATION_BONUS = 65535;
    uint256 constant MAX_VALID_DECIMALS = 255;
    uint256 constant MAX_VALID_RESERVE_FACTOR = 65535;
    uint256 constant MAX_VALID_BORROW_CAP = 68719476735;
    uint256 constant MAX_VALID_SUPPLY_CAP = 68719476735;

56  uint256 constant MAX_VALID_EMODE_CATEGORY = 255;
    uint256 constant MAX_VALID_UNBACKED_MINT_CAP = 68719476735;
    uint256 constant MAX_VALID_DEBT_CEILING = 1099511627775;
```

3.310 CVF-310

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ReserveConfiguration.sol

Recommendation This value seems too high. As 10^77 is the maximum power of 10 that fits into 256 bits, consider using 77 as the maximum valid decimals value.

Listing 310:

```
51  uint256 constant MAX_VALID_DECIMALS = 255;
```

3.311 CVF-311

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** ReserveConfiguration.sol

Description Converting a boolean into an integer via the ternary operator is suboptimal. A more efficient way would be to use assembly to just reinterpret a boolean value as an integer.

Listing 311:

```
174 (uint256(active ? 1 : 0) << IS_ACTIVE_START_BIT_POSITION);
```

3.312 CVF-312

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** ReserveConfiguration.sol

Recommendation Consider giving descriptive names to the returned values.

Listing 312:

```
492 bool ,
    bool ,
    bool ,
    bool ,
    bool
```

3.313 CVF-313

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** BaseImmutableAdminUpgradeabilityProxy.sol

Description No access level specified for this variable, so internal access will be used by default.
Recommendation Consider explicitly specifying an access level.

Listing 313:

```
17 address immutable ADMIN;
```

3.314 CVF-314

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** BaseImmutableAdminUpgradeabilityProxy.sol

Description It looks weird that only the admin may know the admin address.
Recommendation Consider either making this function callable by anyone or removing it.

Listing 314:

```
35 function admin() external ifAdmin returns (address) {
```

3.315 CVF-315

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IPriceOracleGetter.sol

Recommendation The type of the returned value could be more specific.

Listing 315:

```
14 function BASE_CURRENCY() external view returns (address);
```

3.316 CVF-316

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IPriceOracleGetter.sol

Description It is unclear how the base unit is to be used.
Recommendation Consider documented.

Listing 316:

```
17 * @notice Returns the base currency unit
   * @return Returns the base currency unit. 1 ether if eth market
     ↪ and 100000000 if usd market
```

3.317 CVF-317

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IPriceOracleGetter.sol

Recommendation The type of the "asset" argument could be more specific.

Listing 317:

```
27 function getAssetPrice(address asset) external view returns (
     ↪ uint256);
```

3.318 CVF-318

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IPriceOracleGetter.sol

Description The price format and semantics is unclear.
Recommendation Consider documenting.

Listing 318:

```
27 function getAssetPrice(address asset) external view returns (
     ↪ uint256);
```

3.319 CVF-319

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IPoolDataProvider.sol

Description The number formats of these returned values are unclear.
Recommendation Consider documenting.

Listing 319:

```
13 * @return liquidityRate The liquidity rate of the reserve
   * @return variableBorrowRate The variable borrow rate of the
     ↪ reserve
   * @return stableBorrowRate The stable borrow rate of the reserve
   * @return averageStableBorrowRate The average stable borrow rate
     ↪  of the reserve
   * @return liquidityIndex The liquidity index of the reserve
   * @return variableBorrowIndex The variable borrow index of the
     ↪ reserve
```

3.320 CVF-320

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IPoolDataProvider.sol

Recommendation The "asset" argument type could be more specific.

Listing 320:

```
21 function getReserveData(address asset)

44 function getATokenTotalSupply(address asset) external view
     ↪ returns (uint256);
```

3.321 CVF-321

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** IPoolConfigurator.sol

Recommendation Events are usually named via nouns, such as "ReserveIntialization".

Listing 321:

```
20  event ReserveInitialized(

33  event BorrowingEnabledOnReserve(address indexed asset, bool
      ↪ stableRateEnabled);

39  event BorrowingDisabledOnReserve(address indexed asset);

48  event CollateralConfigurationChanged(

59  event StableRateEnabledOnReserve(address indexed asset);

65  event StableRateDisabledOnReserve(address indexed asset);

71  event ReserveActivated(address indexed asset);

77  event ReserveDeactivated(address indexed asset);

83  event ReserveFrozen(address indexed asset);

89  event ReserveUnfrozen(address indexed asset);

95  event ReservePaused(address indexed asset);

101 event ReserveUnpaused(address indexed asset);

107 event ReserveDropped(address indexed asset);

114 event ReserveFactorChanged(address indexed asset, uint256 factor
      ↪ );

121 event BorrowCapChanged(address indexed asset, uint256 borrowCap)
      ↪ ;

128 event SupplyCapChanged(address indexed asset, uint256 supplyCap)
      ↪ ;

135 event LiquidationProtocolFeeChanged(address indexed asset,
      ↪ uint256 fee);
```

(... 142, 148, 159, 173, 180, 188, 200, 212, 223, 229, 235, 241, 247, 253)

3.322 CVF-322

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IPoolConfigurator.sol

Recommendation The types of event parameters referring to contracts with known APIs could be more specific.

Listing 322:

```
21    address indexed asset,
      address indexed aToken,
      address stableDebtToken,
      address variableDebtToken,
      address interestRateStrategyAddress

33  event BorrowingEnabledOnReserve(address indexed asset, bool
      ↪ stableRateEnabled);

39  event BorrowingDisabledOnReserve(address indexed asset);

49    address indexed asset,

59  event StableRateEnabledOnReserve(address indexed asset);

65  event StableRateDisabledOnReserve(address indexed asset);

71  event ReserveActivated(address indexed asset);

77  event ReserveDeactivated(address indexed asset);

83  event ReserveFrozen(address indexed asset);

89  event ReserveUnfrozen(address indexed asset);

95  event ReservePaused(address indexed asset);

101 event ReserveUnpaused(address indexed asset);

107 event ReserveDropped(address indexed asset);

114 event ReserveFactorChanged(address indexed asset, uint256 factor
      ↪ );

121 event BorrowCapChanged(address indexed asset, uint256 borrowCap)
      ↪ ;

128 event SupplyCapChanged(address indexed asset, uint256 supplyCap)
      ↪ ;
```

(... 135, 142, 148, 164, 173, 180, 189, 201, 213, 223)

3.323 CVF-323

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IPoolConfigurator.sol

Description It is unclear what are the number formats for the fractional event parameters and corresponding function arguments, such as thresholds, fees, and factors.
Recommendation Consider explaining in the documentation comments.

Listing 323:

```
114 event ReserveFactorChanged(address indexed asset, uint256 factor
    ↪ );

135 event LiquidationProtocolFeeChanged(address indexed asset,
    ↪ uint256 fee);

241 event BridgeProtocolFeeUpdated(uint256 protocolFee);

247 event FlashloanPremiumTotalUpdated(uint256 flashloanPremiumTotal
    ↪ );

253 event FlashloanPremiumToProtocolUpdated(uint256
    ↪ flashloanPremiumToProtocol);

364 function setReserveFactor(address asset, uint256 reserveFactor)
    ↪ external;

400 function setLiquidationProtocolFee(address asset, uint256 fee)
    ↪ external;

445 function updateBridgeProtocolFee(uint256 protocolFee) external;

454 function updateFlashloanPremiumTotal(uint256
    ↪ flashloanPremiumTotal) external;

460 function updateFlashloanPremiumToProtocol(uint256
    ↪ flashloanPremiumToProtocol) external;
```

3.324 CVF-324

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IPoolConfigurator.sol

Recommendation "a the"

Listing 324:

```
244 * @notice Emitted when a the total premium on flashloans is
    ↪ updated

250 * @notice Emitted when a the part of the premium that goes to
    ↪ protocol is updated
```

3.325 CVF-325

- **Severity** Minor
- **Status** Opened
- **Category** Unclear behavior
- **Source** IPoolConfigurator.sol

Description The phrase "A valid value is 10000" doesn't make sense. Should it be "The maximum valid value is 10000"?

Listing 325:

```
301 * @dev all the values are expressed in percentages with two
    ↪ decimals of precision. A valid value is 10000, which means
    ↪  100.00%
```

3.326 CVF-326

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** IPoolAddressesProvider.sol

Recommendation Events are usually named via nouns, such as "MarketId", "PoolUpdate", "PoolConfigurationUpdate" etc.

Listing 326:

```
10 event MarketIdSet(string newMarketId);
   event PoolUpdated(address indexed newAddress);
   event PoolConfiguratorUpdated(address indexed newAddress);
   event PriceOracleUpdated(address indexed newAddress);
   event ACLManagerUpdated(address indexed newAddress);
   event ACLAdminUpdated(address indexed newAddress);
   event PriceOracleSentinelUpdated(address indexed newAddress);
   event ProxyCreated(bytes32 id, address indexed newAddress);
   event BridgeAccessControlUpdated(address indexed newAddress);
   event PoolDataProviderUpdated(address indexed newAddress);
20 event AddressSet(bytes32 id, address indexed newAddress, bool
     ↪ hasProxy);
```

3.327 CVF-327

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** IPoolAddressesProvider.sol

Recommendation The types of the "newAddress" parameters could be more specific.

Listing 327:

```
11 event PoolUpdated(address indexed newAddress);
   event PoolConfiguratorUpdated(address indexed newAddress);
   event PriceOracleUpdated(address indexed newAddress);
   event ACLManagerUpdated(address indexed newAddress);
   event ACLAdminUpdated(address indexed newAddress);
   event PriceOracleSentinelUpdated(address indexed newAddress);
   event ProxyCreated(bytes32 id, address indexed newAddress);
   event BridgeAccessControlUpdated(address indexed newAddress);
   event PoolDataProviderUpdated(address indexed newAddress);
20 event AddressSet(bytes32 id, address indexed newAddress, bool
     ↪ hasProxy);
```

3.328 CVF-328

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IPoolAddressesProvider.sol

Recommendation The "id" parameter should be indexed.

Listing 328:

```
20 event AddressSet(bytes32 id, address indexed newAddress, bool
   ↪ hasProxy);
```

3.329 CVF-329

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IPoolAddressesProvider.sol

Description There is no way to know whether the current address is set as a proxy or not.
Recommendation Consider storing and returning such info as it is important when upgrading an address.

Listing 329:

```
57 function getAddress(bytes32 id) external view returns (address);
```

3.330 CVF-330

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IPoolAddressesProvider.sol

Recommendation The argument and return value types for these functions could be more specific.

Listing 330:

```
57  function getAddress(bytes32 id) external view returns (address);

63  function getPool() external view returns (address);

70  function setPoolImpl(address pool) external;

76  function getPoolConfigurator() external view returns (address);

83  function setPoolConfiguratorImpl(address configurator) external;

85  function getPriceOracle() external view returns (address);

87  function setPriceOracle(address priceOracle) external;

93  function getACLManager() external view returns (address);

99  function setACLManager(address aclManager) external;

105 function getACLAdmin() external view returns (address);

111 function setACLAdmin(address aclAdmin) external;

117 function getPriceOracleSentinel() external view returns (address
      ↪ );

123 function setPriceOracleSentinel(address oracleSentinel) external
      ↪ ;

129 function setPoolDataProvider(address dataProvider) external;

135 function getPoolDataProvider() external view returns (address);
```

3.331 CVF-331

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** IPool.sol

Description The order of event parameters differs from the order of corresponding function arguments.
Recommendation Consider using the same order.

Listing 331:

```
17 * @param onBehalfOf The beneficiary of the supplied assets,
     ↪ receiving the aTokens
   * @param amount The amount of supplied assets
   * @param referral The referral code used

200  uint256 amount,
     address onBehalfOf,
     uint16 referralCode
```

3.332 CVF-332

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** IPool.sol

Recommendation Events are usually named via nouns, such as "UnbackedMint", "Unbacked-Backing", etc.

Listing 332:

```
21  event MintUnbacked(

35  event BackUnbacked(address indexed reserve, address indexed
      ↪ backer, uint256 amount, uint256 fee);

110 event ReserveUsedAsCollateralEnabled(address indexed reserve,
      ↪ address indexed user);

117 event ReserveUsedAsCollateralDisabled(address indexed reserve,
      ↪ address indexed user);

174 event ReserveDataUpdated(

188 event MintedToTreasury(address indexed reserve, uint256
      ↪ amountMinted);

222 event UserEModeSet(address indexed user, uint8 categoryId);
```

3.333 CVF-333

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IPool.sol

Recommendation The types of event parameters refersing to assets could be made more specific.

Listing 333:

```
22    address indexed reserve,

35  event BackUnbacked(address indexed reserve, address indexed
      ↪ backer, uint256 amount, uint256 fee);

46    address indexed reserve,

60  event Withdraw(address indexed reserve, address indexed user,
      ↪ address indexed to, uint256 amount);

74    address indexed reserve,

91    address indexed reserve,

103 event Swap(address indexed reserve, address indexed user,
      ↪ uint256 rateMode);

110 event ReserveUsedAsCollateralEnabled(address indexed reserve,
      ↪ address indexed user);

117 event ReserveUsedAsCollateralDisabled(address indexed reserve,
      ↪ address indexed user);

124 event RebalanceStableBorrowRate(address indexed reserve, address
      ↪  indexed user);

138   address indexed asset,

156   address indexed collateralAsset,
      address indexed debtAsset,

175   address indexed reserve,

188 event MintedToTreasury(address indexed reserve, uint256
      ↪ amountMinted);
```

3.334 CVF-334

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** IPool.sol

Description Unlike most of other events declared in this interface, these events don't index the "user" parameter.
Recommendation Consider indexing it, probably at the cost of not indexing the "referral" parameter.

Listing 334:

```
23 address user,

47 address user,

75 address user,
```

3.335 CVF-335

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** IPool.sol

Description The last event parameter "fee" is not documented, while the non-existing parameter "amount" is documentation instead.
Recommendation Consider fixing either documentation or the event declaration.

Listing 335:

```
33  * @param amount The amount added as backing

35 event BackUnbacked(address indexed reserve, address indexed
    ↪ backer, uint256 amount, uint256 fee);
```

3.336 CVF-336

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** IPool.sol

Description Some event declarations are formatted as multiple lines, while these declarations are formatted as single lines.
Recommendation Consider using consistent formatting across the code.

Listing 336:

```
35  event BackUnbacked(address indexed reserve, address indexed
      ↪ backer, uint256 amount, uint256 fee);

60  event Withdraw(address indexed reserve, address indexed user,
      ↪ address indexed to, uint256 amount);

103 event Swap(address indexed reserve, address indexed user,
      ↪ uint256 rateMode);

110 event ReserveUsedAsCollateralEnabled(address indexed reserve,
      ↪ address indexed user);

117 event ReserveUsedAsCollateralDisabled(address indexed reserve,
      ↪ address indexed user);

124 event RebalanceStableBorrowRate(address indexed reserve, address
      ↪  indexed user);

188 event MintedToTreasury(address indexed reserve, uint256
      ↪ amountMinted);

222 event UserEModeSet(address indexed user, uint8 categoryId);
```

3.337 CVF-337

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** IPool.sol

Description The order of event parameters differs from the order of corresponding function arguments.
Recommendation Consider using the same order.

Listing 337:

```
48  address indexed onBehalfOf,
    uint256 amount,
50  uint16 indexed referralCode

237 uint256 amount,
    address onBehalfOf,
    uint16 referralCode

259 uint256 amount,
260 address onBehalfOf,
    uint16 referralCode,
```

3.338 CVF-338

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** IPool.sol

Description The event parameter and the function argument basically mean the same, but have different names.
Recommendation Consider using consistent naming.

Listing 338:

```
69  * @param borrowRateMode The rate mode: 1 for Stable, 2 for
      ↪ Variable

293 * @param interestRateMode The interest rate mode at which the
      ↪ user wants to borrow: 1 for Stable, 2 for Variable
```

3.339 CVF-339

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** IPool.sol

Description The order of event parameters and the order of corresponding function arguments are different.
Recommendation Consider using the same order.

Listing 339:

```
 76 address indexed onBehalfOf,
    uint256 amount,
    uint256 borrowRateMode,

 80 uint16 indexed referral

302 uint256 amount,
    uint256 interestRateMode,
    uint16 referralCode,
    address onBehalfOf
```

3.340 CVF-340

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IPool.sol

Recommendation The type of function arguments and returned values referring to assets could be more specific.

Listing 340:

```
199   address asset,

212   address asset,

236   address asset,

258   address asset,

280   address asset,

301   address asset,

321   address asset,

344   address asset,

367   address asset,

378 function swapBorrowRateMode(address asset, uint256 rateMode)
      ↪ external;

389 function rebalanceStableBorrowRate(address asset, address user)
      ↪ external;

396 function setUserUseReserveAsCollateral(address asset, bool
      ↪ useAsCollateral) external;

410   address collateralAsset,
      address debtAsset,

436   address[] calldata assets,

458   address asset,

497   address asset,

509 function dropReserve(address asset) external;
```

(... 517, 526, 533, 553, 560, 567, 580, 593, 683, 698)

3.341 CVF-341

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IPool.sol

Description Most events in this interface are declared before the function declarations, but not this one.
Recommendation Consider moving this even declaration to where other events are declared.

Listing 341:

```
222 event UserEModeSet(address indexed user, uint8 categoryId);
```

3.342 CVF-342

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** IPool.sol

Description The order of arguments in the documentation comment and in the function declaration is different.
Recommendation Consider using the same order.

Listing 342:

```
250 * @param deadline The deadline timestamp that the permit is
      ↪ valid
    * @param referralCode Code used to register the integrator
      ↪ originating the operation, for potential rewards.

261   uint16 referralCode,
      uint256 deadline,
```

3.343 CVF-343

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IPool.sol

Recommendation Consider wrapping permit-related arguments into a struct to separate them from other arguments.

Listing 343:

```
263 uint8 permitV,
    bytes32 permitR,
    bytes32 permitS

349 uint8 permitV,
350 bytes32 permitR,
    bytes32 permitS
```

3.344 CVF-344

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IPool.sol

Description Passing a single array of structs with three fields would be more efficient than passing three parallel arrays.

Listing 344:

```
436 address[] calldata assets,
    uint256[] calldata amounts,
    uint256[] calldata modes,
```

3.345 CVF-345

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IPool.sol

Description These functions should emit some events,but the interface doesn't define suitable events.
Recommendation Consider defining appropriate events in the interface.

Listing 345:

```
496 function initReserve(

509 function dropReserve(address asset) external;

517 function setReserveInterestRateStrategyAddress(address asset,
      ↪ address rateStrategyAddress)

526 function setConfiguration(address asset, uint256 configuration)
      ↪ external;

605 function updateBridgeProtocolFee(uint256 bridgeProtocolFee)
      ↪ external;

615 function updateFlashloanPremiums(

627 function configureEModeCategory(uint8 id, DataTypes.
      ↪ EModeCategory memory config) external;

640 function setUserEMode(uint8 categoryId) external;
```

3.346 CVF-346

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IPool.sol

Recommendation The types of these arguments could be more specific.

Listing 346:

```
498 address aTokenAddress,
    address stableDebtAddress,
500 address variableDebtAddress,
    address interestRateStrategyAddress
```

3.347 CVF-347

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IPool.sol

Recommendation The type of the "rateStrategyAddress" argument could be more specific.

Listing 347:

```
517 function setReserveInterestRateStrategyAddress(address asset,
    ↪ address rateStrategyAddress)
```

3.348 CVF-348

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** IPool.sol

Description The argument type in a setter and the return type in the corresponding getter are different, while holding basically the same data.
Recommendation Consider using the same type for the getter and the setter.

Listing 348:

```
526 function setConfiguration(address asset, uint256 configuration)
    ↪ external;

533 function getConfiguration(address asset)

536   returns (DataTypes.ReserveConfigurationMap memory);
```

3.349 CVF-349

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** IPool.sol

Description These two lines are inconsistent with each other.
Recommendation Consider fixing them.

Listing 349:

```
563 * @notice Returns the state and configuration of the reserve

565 * @return The state of the reserve
```

3.350 CVF-350

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** IInitializableDebtToken.sol

Recommendation Events are usually named via nouns, such as "Initialization".

Listing 350:

```
23 event Initialized(
```

3.351 CVF-351

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IInitializableDebtToken.sol

Recommendation The type for these parameters could be more specific, such as "IPool" for the "pool" parameter, "IAaveIncentivesController" for "incentvesController" etc.

Listing 351:

```
24 address indexed underlyingAsset,
   address indexed pool,
   address incentivesController,
```

3.352 CVF-352

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** IInitializableAToken.sol

Recommendation Events are usually named via nouns, such as "Initialization".

Listing 352:

```
24 event Initialized(
```

3.353 CVF-353

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IInitializableAToken.sol

Recommendation The type of these parameters could be more specific, such as "IPool" for the "pool" parameter, "IAaveIncentivesController" for "incentivesController" etc.

Listing 353:

```
25 address indexed underlyingAsset,
   address indexed pool,
   address treasury,
   address incentivesController,
```

3.354 CVF-354

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IInitializableAToken.sol

Recommendation The types of these arguments could be more specific.

Listing 354:

```
46 address treasury,
   address underlyingAsset,
```

3.355 CVF-355

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IAToken.sol

Description There is no separate parameter for the address that receives the minted tokens.
Recommendation Consider adding such parameter.

Listing 355:

```
21 event Mint(address indexed from, uint256 value, uint256 index);
```

3.356 CVF-356

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** IAToken.sol

Description The names are different, while the semantics is the same.
Recommendation Consider using consistent naming across the code.

Listing 356:

```
39 * @param target The address that will receive the underlying

57 * @param receiverOfUnderlying The address that will receive the
   ↪ underlying
```

3.357 CVF-357

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** IAToken.sol

Description EIP-2612 defines three functions, but only one of them is defined here.
Recommendation Consider defining the two other to make the interface compliant with EIP-2612 Also, consider moving the EIP-2612 stuff into a separate interface and inherit from it.

Listing 357:

```
105 * https://github.com/ethereum/EIPs/blob/8
    ↪ a34d644aacf0f9f8f00815307fd7dd5da07655f/EIPS/eip-2612.md
```

3.358 CVF-358

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** IAToken.sol

Recommendation The return type for these functions could be more specific.

Listing 358:

```
128 function UNDERLYING_ASSET_ADDRESS() external view returns (
    ↪ address);

133 function RESERVE_TREASURY_ADDRESS() external view returns (
    ↪ address);
```