

Supplementary technical details of MSTest

I. DESCRIPTION OF TEST CASES

ID	Description	Related Properties
acc	0–3	overflow
	4–7	overflow, compart., ASan
	8–11	overflow, compart., ASan
	12–13	arb. read, ROP, compart.
	14–15	arb. read, ROP, compart., ASLR
	16	arb. read, ROP, compart., CPI, PA
	17–18	arb. read, ROP, compart.
	19	arb. read, ROP, compart.
	20	arb. read, CET
	21	arb. read, ROP, CPI, RELRO
mss	22–29	overflow, compart., ASan, tag
	30–37	overflow, compart., ASan, tag
	38–45	overflow, compart., ASan, TBI, LAM, UAI
	46–53	overflow, compart., ASan, tag
	54–59	overflow, compart., DFI, ASan, tag
	60–65	overflow, compart., DFI, ASan, tag
	66	overflow, stack canary, PA, backward CFI, SHSTK
	67–72	overflow, compart., ASan
	79–82	overflow, compart., ASan
	83–85	overflow, compart., ASan
	86–89	overflow, compart., ASan, PA, CET
	90–93	overflow, compart., ASan, tag
	94–117	overflow, compart., ASan, ASLR
	118–129	overflow, compart., ASan, ASLR
	130–132	overflow, compart., ASan
	133	overflow, compart., ASan
	134–135	overflow, compart., ASan
	136–139	type, compart., ASan, ADI, MTE, CHERI
	140–143	type, compart., ASan, ADI, MTE, CHERI
mts	144–146	zeroing, randomization
	147,149	zeroing, randomization
	148	randomization
	150–152	zeroing, randomization
	153–154,156	zeroing, randomization
	155	randomization
	157–159	zeroing, randomization
cpi	160–163	COOP
	164–166	COOP
	167	JOP, PA
	168	JOP, PA
	169	arb. write, ROP, CPI, RELRO
cfi-b	170–171	ROP, fine-grained CFI
	172	ROP, coarse-grained CFI
	173	ROP, fine-grained CFI
	174	ROP, fine-grained CFI
	175–176	ROP, fine-grained CFI
	177–180	ROP, fine-grained CFI
	181	ROP, fine-grained CFI
	182	ROP, CET, shadow stack
	183–186	DEP, ROP
cfi-f	187	arb. execute, ROP, coarse-grained CFI
	188	ROP, fine-grained CFI
	189	ROP, path-sensitive CFI
	190–192	COOP, VTable layout, read-only
	193	COOP, type
	194–195	COOP, type
	196	COPP, type
	197	COOP, type
	198–200	COOP, type
	201–204	COOP, type, class hierarchy analysis
	205	COOP, type
	206	COOP, revocation
	207	JOP, type
	208–211	JOP, type, fine-grained CFI
cfi-f	212–215	JOP, type
	216–219	DEP, JOP
	220	arb. execute, ROP, coarse-grained CFI
	221–222	coarse-grained ASLR, JOP, ROP
	223–226	DEP, JOP

II. DEPENDENCY OF TEST CASES

Explanation for dependency with examples:

Example 1: —

No dependency is required. The test case is always tested.

Example 2: 3

The test case is tested if test case 3 returns 0 (exploitable).

Example 3: 0 & 8

The test case is tested if both test cases 8 and 0 return 0.

Example 4: 0 | 8

The test case is tested if either test case 8 or 0 returns 0.

Example 5: 0 & 8, 3

The test case is tested in two possible scenarios, 0 & 8 and 3. Scenario 0 & 8 is checked before scenario 3. Depending on the enabling scenario, the test case may use different macro and runtime arguments.

Example 6: 0 & 8, —

The test case is tested in two possible scenarios, 0 & 8 and —, where the latter is a backup scenario. The test case is always tested but may use different macro and runtime arguments in different scenarios.

A. Generic memory access capability (acc)

ID	Name	Dependency
0	check-data-pointer-arithmetic-stack	—
1	check-data-pointer-arithmetic-heap	—
2	check-data-pointer-arithmetic-data	—
3	check-data-pointer-arithmetic-rodata	—
4	check-inter-obj-stack-redzone	—
5	check-inter-obj-heap-redzone	—
6	check-inter-obj-data-redzone	—
7	check-inter-obj-rodata-redzone	—
8	check-intra-obj-stack-redzone	—
9	check-intra-obj-heap-redzone	—
10	check-intra-obj-data-redzone	—
11	check-intra-obj-rodata-redzone	—
12	copy-stackra-to-heap-explicit-arith	187
13	copy-stackra-to-heap-implicit-arith	—
14	check-prog-ASLR	186
15	check-stack-region-ASLR	14
16	read-func	—
17	get-ra-offset-v-p-g0	—
18	get-ra-offset-v-p-g1	—
19	get-frame-size	—
20	check-IBT	186
21	read-GOT	20

B. Memory spatial safety (mss)

ID	Name	Dependency
22	read-by-enclosing-array-index-stack-overflow	8, 0, —
23	read-by-enclosing-array-index-stack-underflow	8, 0, —
24	read-by-enclosing-array-index-heap-overflow	9, 1, —
25	read-by-enclosing-array-index-data-overflow	10, 2, —
26	read-by-enclosing-array-index-rodata-overflow	11, 3, —
27	read-by-enclosing-array-index-heap-underflow	9, 1, —
28	read-by-enclosing-array-index-data-underflow	10, 2, —
29	read-by-enclosing-array-index-rodata-underflow	11, 3, —
30	read-by-enclosing-array-pointer-stack-overflow	46, —, —
31	read-by-enclosing-array-pointer-heap-overflow	47, —, —
32	read-by-enclosing-array-pointer-data-overflow	48, —, —
33	read-by-enclosing-array-pointer-rodata-overflow	49, —, —
34	read-by-enclosing-array-pointer-stack-underflow	50, —, —
35	read-by-enclosing-array-pointer-heap-underflow	51, —, —
36	read-by-enclosing-array-pointer-data-underflow	52, —, —
37	read-by-enclosing-array-pointer-rodata-underflow	53, —, —
38	read-by-bare-array-pointer-stack-overflow	4, 0, —
39	read-by-bare-array-pointer-heap-overflow	5, 1, —
40	read-by-bare-array-pointer-data-overflow	6, 2, —
41	read-by-bare-array-pointer-rodata-overflow	7, 3, —
42	read-by-bare-array-pointer-stack-underflow	4, 0, —
43	read-by-bare-array-pointer-heap-underflow	5, 1, —
44	read-by-bare-array-pointer-data-underflow	6, 2, —
45	read-by-bare-array-pointer-rodata-underflow	7, 3, —
46	read-by-enclosing-array-pointer-large-count-stack-overflow	8 & 0, —
47	read-by-enclosing-array-pointer-large-count-heap-overflow	9 & 1, —
48	read-by-enclosing-array-pointer-large-count-data-overflow	10 & 2, —
49	read-by-enclosing-array-pointer-large-count-rodata-overflow	11 & 3, —
50	read-by-enclosing-array-pointer-large-count-stack-underflow	8 & 0

Following the previous table.

ID	Name	Dependency
51	read-by-enclosing-array-pointer-large-count-heap-underflow	9 & 1, —
52	read-by-enclosing-array-pointer-large-count-data-underflow	& 2, —
53	read-by-enclosing-array-pointer-large-count-rodata-underflow	& 3, —
54	write-by-enclosing-array-index-stack-overflow	8, 0, —
55	write-by-enclosing-array-index-heap-overflow	9, 1, —
56	write-by-enclosing-array-index-data-overflow	10, 2, —
57	write-by-enclosing-array-index-stack-underflow	8, 0, —
58	write-by-enclosing-array-index-heap-underflow	9, 1, —
59	write-by-enclosing-array-index-data-underflow	10, 2, —
60	write-by-enclosing-array-pointer-stack-overflow	73, —, —
61	write-by-enclosing-array-pointer-heap-overflow	74, —, —
62	write-by-enclosing-array-pointer-data-overflow	75, —, —
63	write-by-enclosing-array-pointer-stack-underflow	76, —, —
64	write-by-enclosing-array-pointer-heap-underflow	77, —, —
65	write-by-enclosing-array-pointer-data-underflow	78, —, —
66	write-by-stack-pointer	—
67	write-by-bare-array-pointer-stack-overflow	4, 0, —
68	write-by-bare-array-pointer-heap-overflow	5, 1, —
69	write-by-bare-array-pointer-data-overflow	6, 2, —
70	write-by-bare-array-pointer-stack-underflow	4, 0, —
71	write-by-bare-array-pointer-heap-underflow	5, 1, —
72	write-by-bare-array-pointer-data-underflow	6, 2, —
73	write-by-enclosing-array-pointer-large-count-stack-overflow	8, 0
74	write-by-enclosing-array-pointer-large-count-heap-overflow	9, 1
75	write-by-enclosing-array-pointer-large-count-data-overflow	10, 2
76	write-by-enclosing-array-pointer-large-count-stack-underflow	8, 0
77	write-by-enclosing-array-pointer-large-count-heap-underflow	9, 1
78	write-by-enclosing-array-pointer-large-count-data-underflow	10, 2
79	read-cross-object-ptr-stack	38, 42
80	read-cross-object-ptr-heap	31, 35
81	read-cross-object-ptr-data	32, 36
82	read-cross-object-ptr-rodata	33, 37
83	write-cross-object-ptr-stack-overflow	67
84	write-cross-object-ptr-heap-overflow	61
85	write-cross-object-ptr-data-overflow	62
86	read-cross-frame-index	22
87	read-cross-frame-ptr	79
88	write-cross-frame-index	54 57
89	write-cross-frame-ptr	83
90	read-cross-page-index-stack	22
91	read-cross-page-ptr-stack	79
92	write-cross-page-index-stack	54 & 57
93	write-cross-page-ptr-stack	83
94	read-cross-segment-stack-to-heap-index	90
95	read-cross-segment-stack-to-heap-ptr	91
96	read-cross-segment-stack-to-data-index	90
97	read-cross-segment-stack-to-data-ptr	91
98	read-cross-segment-stack-to-rodata-index	90
99	read-cross-segment-stack-to-rodata-ptr	91
100	read-cross-segment-heap-to-stack-index	24 27
101	read-cross-segment-heap-to-stack-ptr	80
102	read-cross-segment-heap-to-data-index	24 27
103	read-cross-segment-heap-to-data-ptr	80
104	read-cross-segment-heap-to-rodata-index	24 27
105	read-cross-segment-heap-to-rodata-ptr	80
106	read-cross-segment-data-to-stack-index	25 28
107	read-cross-segment-data-to-stack-ptr	81
108	read-cross-segment-data-to-heap-index	25 28
109	read-cross-segment-data-to-heap-ptr	81
110	read-cross-segment-data-to-rodata-index	25 28
111	read-cross-segment-data-to-rodata-ptr	81
112	read-cross-segment-rodata-to-stack-index	26 29
113	read-cross-segment-rodata-to-stack-ptr	82
114	read-cross-segment-rodata-to-heap-index	26 29
115	read-cross-segment-rodata-to-heap-ptr	82
116	read-cross-segment-rodata-to-data-index	26 29
117	read-cross-segment-rodata-to-data-ptr	82
118	write-cross-segment-stack-to-heap-index	92
119	write-cross-segment-stack-to-heap-ptr	93
120	write-cross-segment-stack-to-data-index	92
121	write-cross-segment-stack-to-data-ptr	93
122	write-cross-segment-heap-to-stack-index	58, 55
123	write-cross-segment-heap-to-stack-ptr	84, 84
124	write-cross-segment-heap-to-data-index	58, 55
125	write-cross-segment-heap-to-data-ptr	84, 84
126	write-cross-segment-data-to-stack-index	59, 56
127	write-cross-segment-data-to-stack-ptr	85, 85
128	write-cross-segment-data-to-heap-index	59, 56
129	write-cross-segment-data-to-heap-ptr	85, 85

Following the previous table.

ID	Name	Dependency
130	write-spray-cross-object-stack	83
131	write-spray-cross-object-heap	84 84
132	write-spray-cross-object-data	85 85
133	write-spray-cross-frame	89
134	write-spray-cross-page-in-stack	93
135	write-spray-cross-page-in-heap	131
136	read-scalar-cast-to-array-stack-overflow	8
137	read-scalar-cast-to-array-heap-overflow	9
138	read-scalar-cast-to-array-data-overflow	10
139	read-scalar-cast-to-array-rodata-overflow	11
140	read-scalar-cast-to-scalar-stack-overflow	8
141	read-scalar-cast-to-scalar-heap-overflow	9
142	read-scalar-cast-to-scalar-data-overflow	10
143	read-scalar-cast-to-scalar-rodata-overflow	11

C. Memory temporal safety (*mts*)

ID	Name	Dependency
144	double-free	155
145	write-by-double-free-reallocate	144
146	access-by-double-free-reallocate	144
147	access-after-free-alias-stack	148
148	reallocate-stack	—
149	access-after-reclaim-stack	148
150	write-after-free-stack	—
151	write-before-reclaim-stack	150 148
152	write-after-reclaim-stack	148
153	access-after-free-org-heap	155
154	access-after-free-alias-heap	155 148
155	reallocate-heap	—
156	access-after-reclaim-heap	155
157	write-after-free-heap	—
158	write-before-reclaim-heap	157 155
159	write-after-reclaim-heap	155

D. Code pointer integrity (*cpi*)

ID	Name	Dependency
160	read-stack-vtable-pointer	167
161	read-heap-vtable-pointer	167
162	read-data-vtable-pointer	163
163	read-rodata-vtable-pointer	—
164	write-stack-vtable-pointer	—
165	write-heap-vtable-pointer	—
166	write-data-vtable-pointer	—
167	func-pointer-assign	—
168	func-pointer-arithmetic	167 & (0 1 2 3)
169	modify-GOT	21

E. Backward control-flow Integrity (*cfi-b*)

ID	Name	Dependency
170	cfi-return-to-parent-non-call-site-by-asmfunc	—
171	cfi-return-to-parent-non-call-site-by-vfunc	—
172	cfi-return-to-parent-non-call-site	176 & 170
173	cfi-return-to-parent-same-call-site	73 ((18 17) & 66)
174	cfi-return-to-parent-same-call-site-diffargs	73 ((18 17) & 66)
175	cfi-return-to-parent-wrong-call-site-fakefunc-offset	173
176	cfi-return-to-parent-wrong-call-site-asm-offset	173
177	cfi-return-to-peer-asm-func	—
178	cfi-return-to-peer-func	177
179	cfi-return-to-peer-mfunc	177
180	cfi-return-to-peer-vfunc	177
181	cfi-return-to-libc	178
182	cfi-return-without-call	19 & 167
183	cfi-return-to-instruction-in-rodata	172 & 18 & 223
184	cfi-return-to-instruction-in-data	172 & 18 & 224
185	cfi-return-to-instruction-in-stack	172 & 18 & 225
186	cfi-return-to-instruction-in-heap	172 & 18 & 226

F. Forward control-flow integrity (*cfi-f*)

ID	Name	Dependency
187	cfi-call-mid-func	189,168 & 16
188	cfi-call-wrong-func-within-static-analysis	16 167
189	cfi-call-wrong-func	188
190	cfi-call-fake-vtable-with-func-stack	196
191	cfi-call-fake-vtable-with-func-heap	196
192	cfi-call-fake-vtable-with-func-data	196
193	cfi-call-fake-vtable-arg-num	196
194	cfi-call-fake-vtable-arg-type	196
195	cfi-call-fake-vtable-arg-type-modified	194
196	cfi-call-fake-vtable	165 & 161, 164 & 160, 166 & 162
197	cfi-call-wrong-vtable-func-num	205
198	cfi-call-wrong-vtable-arg-num	205
199	cfi-call-wrong-vtable-arg-type	205
200	cfi-call-wrong-vtable-arg-type-modified	199
201	cfi-call-wrong-vtable-parent	203
202	cfi-call-wrong-vtable-sibling	203
203	cfi-call-wrong-vtable-child	165 & 161, 164 & 160, 166 & 162
204	cfi-call-wrong-vtable-offset	205 168
205	cfi-call-wrong-vtable	203
206	cfi-call-wrong-vtable-released	205 153 150
207	cfi-call-wrong-num-arg-func	198 189
208	cfi-call-wrong-type-arg-int2double-func	199 189
209	cfi-call-wrong-type-arg-op2double-func	199 189
210	cfi-call-wrong-type-arg-op2intp-func	199 189
211	cfi-call-wrong-type-arg-fp2dp-func	199 189
212	cfi-call-wrong-type-arg-dp2fp-func-rodata	211
213	cfi-call-wrong-type-arg-dp2fp-func-data	211 208 209 210
214	cfi-call-wrong-type-arg-dp2fp-func-stack	211
215	cfi-call-wrong-type-arg-dp2fp-func-heap	211
216	cfi-call-instruction-in-rodata	223
217	cfi-call-instruction-in-data	224 & 192
218	cfi-call-instruction-in-stack	225 & 190
219	cfi-call-instruction-in-heap	226 & 191
220	cfi-jump-mid-func	16 167
221	cfi-jump-func-ra-from-heap-memcpy-explicit-arith	12 226
222	cfi-jump-func-ra-from-heap-memcpy-implicit-arith	13 226
223	cfi-jump-instruction-in-rodata	220 & 212
224	cfi-jump-instruction-in-data	220 & 213
225	cfi-jump-instruction-in-stack	220 & 214
226	cfi-jump-instruction-in-heap	220 & 215

III. LIST OF ASSEMBLY MACROS AND FUNCTIONS

- `GET_DISTANCE(dis, pa, pb)`: Return the distance between pointer `pa` and `pb` by `dis`.
- `READ_STACK_DAT(dat, offset)`: Return the stack data `[SP+offset]` by `dat`.
- `READ_STACK_DAT_IMM(dat, offset)`: Return the stack data `[SP+offset]` by `dat`. (`offset` is an immediate number)
- `GET_RA_ADDR(ra_addr)`: Return the default location of `RA` of the current stack frame.
- `MOD_STACK_DAT(dat, offset)`: Revise stack data `[SP+offset]` to `dat`.
- `SET_MEM(ptr, var)`: Revise memory data `[ptr]` to `var`.
- `JMP_DAT(ptr)`: Jump to `ptr`.
- `JMP_DAT_PTR(ptr)`: Jump to `[ptr]`.
- `PASS_INT_ARG0_IMM(arg)`: Set the first numeric argument for the next function call to `arg` according to ABI.
- `PUSH_FAKE_RET(ra, fsize)`: Allocate a fake stack frame of `fsize*8` bytes with a fake `RA`.
- `FUNC_MACHINE_CODE`: A snippet of machine code embedded with an illegal instruction.
- `GET_SP_LOC(loc)`: Return `SP` by `loc`.
- `get_got_func(void **gotp, void *label, int cet)`: Return function `label()`'s location in GOT by `gotp`. Intel CET is effective when `cet` is true.