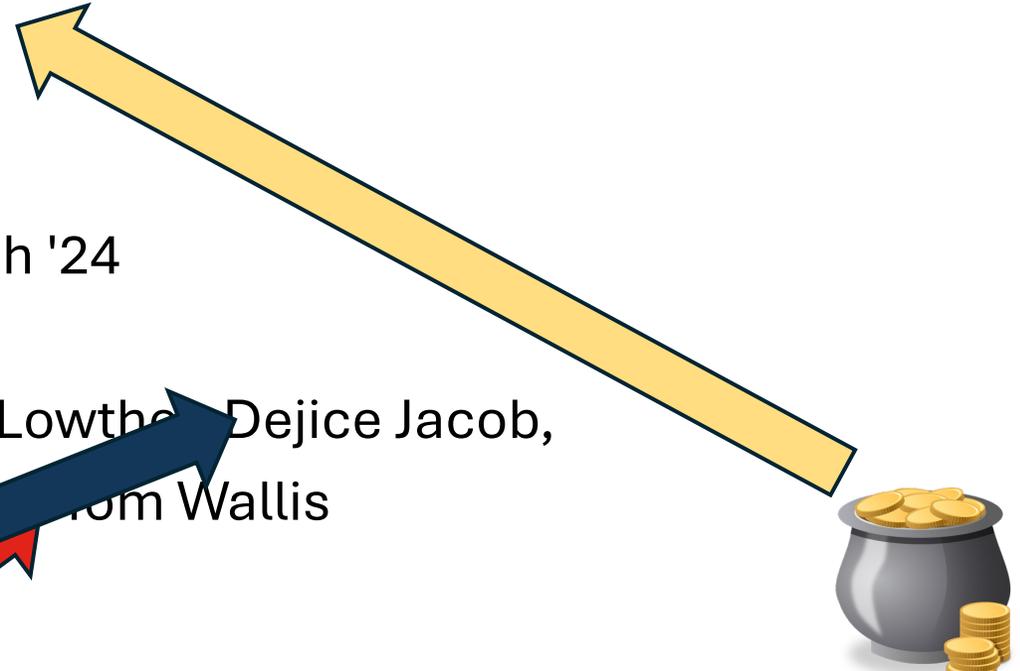


Capable VMs

23 Apr 24 - CHERITech '24

Jacob Bramley, Andrei Lasca, Duncan Lowther, Dejice Jacob,
Jeremy Sizer, Laurie Tratt, Tom Wallis



University
of Glasgow



arm



Virtual machines (VMs, also known as managed language runtimes) are ubiquitous components in the modern software stack. They power the web, running in client-side browsers, server-side applications, and smartphone apps. In any ranking of popular programming languages, at least **half of the top ten languages run on VMs** (e.g. Python, Java, C#, JavaScript, PHP).

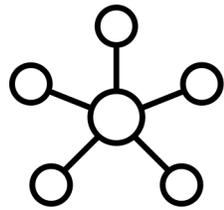
A key problem is that **VM security** has traditionally been a secondary concern relative to performance. Industrial strength VMs have large, complex code-bases, and large numbers of hand-crafted optimizations. Not only are they beyond any one person's ability to understand, but security has tended to be treated reactively: mature, widely used VMs such as HotSpot (the standard Java VM) regularly have **50-100 CVEs per year**.

The **CapableVMs** project hypothesises that **CHERI** hardware-enforced **capabilities** are the first realistic technique to **make VM security proactive**.

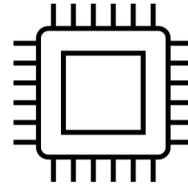
Why are virtual machines special?



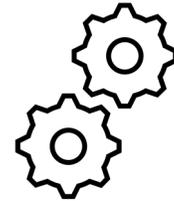
lots of low-level
platform-specific
systems code



multiple interacting
dynamic
components



intensive
allocation and
garbage collection

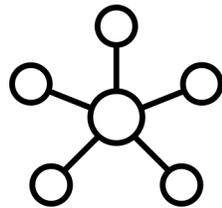
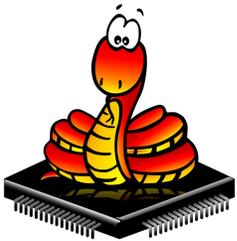


runtime code
generation

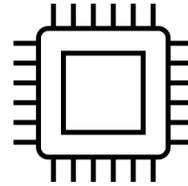
Why are virtual machines special?



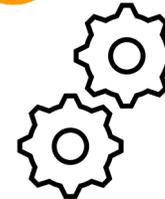
lots of low-level
platform-specific
systems code



multiple interacting
dynamic
components



intensive
allocation and
garbage collection

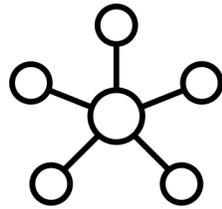
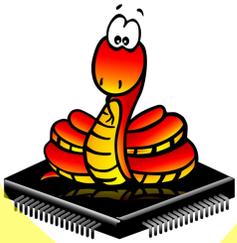


runtime code
generation

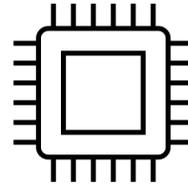
Why are virtual machines special?



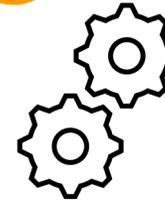
lots of low-level
platform-specific
systems code



multiple interacting
dynamic
components



intensive
allocation and
garbage collection



runtime code
generation

1. Low-level system-specific code

- CHERIfication
- Specific porting process
- Measure proportion of LoC altered
- KDE: 0.026%
- higher for systems code
 - 0.18% for MicroPython
 - 1% for smalloc

Cherifying Linux: A Practical View on using CHERI

Kui Wang
Dmitry Kasatkin

Vincent Ahlrichs
Lukas Auer

Jan-Erik Ekberg
jan.erik.ekberg@huawei.com
Huawei Technologies

Morello MicroPython: A Python Interpreter for CHERI

Duncan Lowther
University of Glasgow
Glasgow, United Kingdom
duncan.lowther@glasgow.ac.uk

Dejice Jacob
University of Glasgow
Glasgow, United Kingdom
dejice.jacob@glasgow.ac.uk

Jeremy Singer
University of Glasgow
Glasgow, United Kingdom
jeremy.singer@glasgow.ac.uk

Abstract

Arm Morello is a prototype system that supports CHERI hardware capabilities for improving runtime security. As Morello becomes more widely available, there is a growing effort to port open source code projects to this novel platform. Although high-level applications generally need minimal code refactoring for CHERI compatibility, low-level systems code bases require significant modification to comply with the stringent memory safety constraints that are dynamically enforced by Morello. In this paper, we describe our work on porting the MicroPython interpreter to Morello with the CheriBSD OS. Our key contribution is to present a set of generic lessons for adapting managed runtime execution environments to CHERI, including (1) a characterization of necessary source code changes, (2) an evaluation of runtime performance of the interpreter on Morello, and (3) a demonstration of pragmatic memory safety bug detection. Although MicroPython is a lightweight interpreter, mostly written in C, we believe that the changes we have implemented and the lessons we have learned are more widely applicable. To the best of our knowledge, this is the first published description of meaningful experience for scripting language runtime engineering with CHERI and Morello.

1 Introduction

In 2021, Arm released a prototype platform code-named 'Morello' [2, 9] which realizes the CHERI hardware capability concept [22, 25] in an industrial strength microprocessor. A *capability* is a double-width 'fat' pointer that includes metadata for address bounds and access permissions. Additionally, CHERI capabilities have an out-of-band *tag* to ensure pointer validity. The premise of hardware capabilities is that entire classes of memory vulnerabilities can be eliminated, including spatial bugs (i.e. out-of-bounds reads and writes) and temporal bugs (i.e. use-after-free bugs) [12].

In this paper, we describe our experience and lessons learned during a full port of the MicroPython framework to Morello. We modify the C source code of MicroPython in order to provide runtime awareness of CHERI capabilities. There were two logical stages to this work: firstly the MicroPython code was refactored to eliminate compiler errors and warnings, as described in Section 3; secondly memory safety enforcement of capabilities was leveraged to generate tight bounds on runtime allocations, as described in Section 4. In Section 5, we analyse test coverage for MicroPython on the Morello CheriBSD platform, which has almost identical results to the AArch64 FreeBSD platform. Section 6 charac-

```
import dodgylib

tiny1 = bytearray(3)
tiny2 = bytearray(12)

# setup 'Oh' raw string
tiny1[0] = 0x4f # O
tiny1[1] = 0x68 # h
tiny1[2] = 0x00 # \0

# setup 'Hello' raw string
tiny2[0] = 0x48 # H
tiny2[1] = 0x65 # e
tiny2[2] = 0x6c # l
tiny2[3] = 0x6c # l
tiny2[4] = 0x6f # o
tiny2[5] = 0x00 # \0
```

```
print(tiny1.decode('utf-8'))
dodgylib.dodgy(tiny1)
print(tiny2.decode('utf-8'))
```

```
root@amarena:~ #
./micropython-hybrid
exploit.py
```

Oh

HACK!!

```
import ctypes as uct

def dodgy(x):
    ptr = uct.addressof(x)
    unsafe =
uct.bytearray_at(ptr, 2000)
    i = 0
    while unsafe[i] != 0x65 or
unsafe[i+1] != 0x6c:
        i += 1
        if i > 2000:
            break
    unsafe[i] = 0x41
    unsafe[i+1] = 0x43
    unsafe[i+2] = 0x4b
    unsafe[i+3] = 0x21
    unsafe[i+4] = 0x21
    unsafe[i+5] = 0x00
    return
```

```
print(tiny1.decode('utf-8'))
dodgylib.dodgy(tiny1)
print(tiny2.decode('utf-8'))
```

```
root@amarena:~ #
./micropython-purecap
exploit.py
```

Oh

```
In-address space security
exception (core dumped)
```

CVE-2023-7158 Detail

Description

A vulnerability was found in MicroPython up to 1.21.0. It has been classified as critical. Affected is the function `slice_indices` of the file `objslice.c`. The manipulation leads to **heap-based buffer overflow** and can be used to execute the attack remotely. The exploit has been disclosed to the public and may be used. Upgrading to version 1.22.0 is able to mitigate the vulnerability. It is recommended to upgrade the affected component. The identifier assigned to this vulnerability is VDB-249180.

CVE-2023-7152 Detail

Description

A vulnerability, which was classified as critical, has been found in MicroPython 1.21.0/1.22.0-preview. Affected by this issue is the function `poll_set_add_fd` of the file `extmod/modselect.c`. The manipulation leads to **use after free**. The exploit has been disclosed to the public and may be used. The patch is identified as `8b24aa36ba978eafc6114b6798b47b7bfecdca26`. It is recommended to apply a patch to fix this issue. VDB-249158 is the identifier assigned to this vulnerability.

Other findings from MicroPython

- Pointer size assumptions
 - don't affect correctness only
 - they also have an impact on performance
- Porting to a variety of platforms
 - Morello: github.com/glasgowPLI/micropython
 - working on CHERI IoT RISC-V Ibex core



PROG
DONE
LD8
C158
C157
R242
J10
C156
FB3
R249
R250
GND VCC
R176
R183
R241
R176
R179
R167
C17
IC8
C166
LD9
C150
FB2
IC3
IC4
R177
R168
R182

QC.OK
2

ARTY A7

DIGILENT®

00183E046354

C117
C118
C107
C108
IC6
C103
R197
R198
R195
R196
R194
R193
C105
C104
R203
R204
R193
C104

ARTIX-7
100T
CSG324
Xilinx

AVNET®

20K17
D9P1X
IC7



J11
LD11
R278
LD7
LD6
LD5
LD4
LD3
LD2
LD1
LD0
R64
R68
R72
R63
R67
R71
R46
R50
R54
R45
R49
R53
SW3
SW2
SW1
SW0
BTN3
BTN2
BTN1
BTN0

V_P
V_N
XGND
XVREF
A6
A7
A8
A9
A10
A11
J5
R154
J7
J6
CK RST
C34

C100
C99
C98
C97
C96
C95
C94
C93

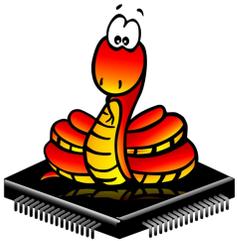
IC11
R260
R261
R258
R257
R265
R266
R217
R217
C171
C170
L1
R254
C179
C178
C177
C176
C175
C174
C173
C172
C171
C170
R253
C183
L2
C168
C167
C166
C165
C164
C163
C162
C161
C160
C159
C158
C157
C156
C155
C154
C153
C152
C151
C150
C149
C148
C147
C146
C145
C144
C143
C142
C141
C140
C139
C138
C137
C136
C135
C134
C133
C132
C131
C130
C129
C128
C127
C126
C125
C124
C123
C122
C121
C120
C119
C118
C117
C116
C115
C114
C113
C112
C111
C110
C109
C108
C107
C106
C105
C104
C103
C102
C101
C100
C99
C98
C97
C96
C95
C94
C93
C92
C91
C90
C89
C88
C87
C86
C85
C84
C83
C82
C81
C80
C79
C78
C77
C76
C75
C74
C73
C72
C71
C70
C69
C68
C67
C66
C65
C64
C63
C62
C61
C60
C59
C58
C57
C56
C55
C54
C53
C52
C51
C50
C49
C48
C47
C46
C45
C44
C43
C42
C41
C40
C39
C38
C37
C36
C35
C34
C33
C32
C31
C30
C29
C28
C27
C26
C25
C24
C23
C22
C21
C20
C19
C18
C17
C16
C15
C14
C13
C12
C11
C10
C9
C8
C7
C6
C5
C4
C3
C2
C1
C0

J2
J3
J4
J5
J6
J7
J8
J9
J10
J11
J12
J13
J14
J15
J16
J17
J18
J19
J20
J21
J22
J23
J24
J25
J26
J27
J28
J29
J30
J31
J32
J33
J34
J35
J36
J37
J38
J39
J40
J41
J42
J43
J44
J45
J46
J47
J48
J49
J50
J51
J52
J53
J54
J55
J56
J57
J58
J59
J60
J61
J62
J63
J64
J65
J66
J67
J68
J69
J70
J71
J72
J73
J74
J75
J76
J77
J78
J79
J80
J81
J82
J83
J84
J85
J86
J87
J88
J89
J90
J91
J92
J93
J94
J95
J96
J97
J98
J99
J100
J101
J102
J103
J104
J105
J106
J107
J108
J109
J110
J111
J112
J113
J114
J115
J116
J117
J118
J119
J120
J121
J122
J123
J124
J125
J126
J127
J128
J129
J130
J131
J132
J133
J134
J135
J136
J137
J138
J139
J140
J141
J142
J143
J144
J145
J146
J147
J148
J149
J150
J151
J152
J153
J154
J155
J156
J157
J158
J159
J160
J161
J162
J163
J164
J165
J166
J167
J168
J169
J170
J171
J172
J173
J174
J175
J176
J177
J178
J179
J180
J181
J182
J183
J184
J185
J186
J187
J188
J189
J190
J191
J192
J193
J194
J195
J196
J197
J198
J199
J200
J201
J202
J203
J204
J205
J206
J207
J208
J209
J210
J211
J212
J213
J214
J215
J216
J217
J218
J219
J220
J221
J222
J223
J224
J225
J226
J227
J228
J229
J230
J231
J232
J233
J234
J235
J236
J237
J238
J239
J240
J241
J242
J243
J244
J245
J246
J247
J248
J249
J250
J251
J252
J253
J254
J255
J256
J257
J258
J259
J260
J261
J262
J263
J264
J265
J266
J267
J268
J269
J270
J271
J272
J273
J274
J275
J276
J277
J278
J279
J280
J281
J282
J283
J284
J285
J286
J287
J288
J289
J290
J291
J292
J293
J294
J295
J296
J297
J298
J299
J300
J301
J302
J303
J304
J305
J306
J307
J308
J309
J310
J311
J312
J313
J314
J315
J316
J317
J318
J319
J320
J321
J322
J323
J324
J325
J326
J327
J328
J329
J330
J331
J332
J333
J334
J335
J336
J337
J338
J339
J340
J341
J342
J343
J344
J345
J346
J347
J348
J349
J350
J351
J352
J353
J354
J355
J356
J357
J358
J359
J360
J361
J362
J363
J364
J365
J366
J367
J368
J369
J370
J371
J372
J373
J374
J375
J376
J377
J378
J379
J380
J381
J382
J383
J384
J385
J386
J387
J388
J389
J390
J391
J392
J393
J394
J395
J396
J397
J398
J399
J400
J401
J402
J403
J404
J405
J406
J407
J408
J409
J410
J411
J412
J413
J414
J415
J416
J417
J418
J419
J420
J421
J422
J423
J424
J425
J426
J427
J428
J429
J430
J431
J432
J433
J434
J435
J436
J437
J438
J439
J440
J441
J442
J443
J444
J445
J446
J447
J448
J449
J450
J451
J452
J453
J454
J455
J456
J457
J458
J459
J460
J461
J462
J463
J464
J465
J466
J467
J468
J469
J470
J471
J472
J473
J474
J475
J476
J477
J478
J479
J480
J481
J482
J483
J484
J485
J486
J487
J488
J489
J490
J491
J492
J493
J494
J495
J496
J497
J498
J499
J500
J501
J502
J503
J504
J505
J506
J507
J508
J509
J510
J511
J512
J513
J514
J515
J516
J517
J518
J519
J520
J521
J522
J523
J524
J525
J526
J527
J528
J529
J530
J531
J532
J533
J534
J535
J536
J537
J538
J539
J540
J541
J542
J543
J544
J545
J546
J547
J548
J549
J550
J551
J552
J553
J554
J555
J556
J557
J558
J559
J560
J561
J562
J563
J564
J565
J566
J567
J568
J569
J570
J571
J572
J573
J574
J575
J576
J577
J578
J579
J580
J581
J582
J583
J584
J585
J586
J587
J588
J589
J590
J591
J592
J593
J594
J595
J596
J597
J598
J599
J600
J601
J602
J603
J604
J605
J606
J607
J608
J609
J610
J611
J612
J613
J614
J615
J616
J617
J618
J619
J620
J621
J622
J623
J624
J625
J626
J627
J628
J629
J630
J631
J632
J633
J634
J635
J636
J637
J638
J639
J640
J641
J642
J643
J644
J645
J646
J647
J648
J649
J650
J651
J652
J653
J654
J655
J656
J657
J658
J659
J660
J661
J662
J663
J664
J665
J666
J667
J668
J669
J670
J671
J672
J673
J674
J675
J676
J677
J678
J679
J680
J681
J682
J683
J684
J685
J686
J687
J688
J689
J690
J691
J692
J693
J694
J695
J696
J697
J698
J699
J700
J701
J702
J703
J704
J705
J706
J707
J708
J709
J710
J711
J712
J713
J714
J715
J716
J717
J718
J719
J720
J721
J722
J723
J724
J725
J726
J727
J728
J729
J730
J731
J732
J733
J734
J735
J736
J737
J738
J739
J740
J741
J742
J743
J744
J745
J746
J747
J748
J749
J750
J751
J752
J753
J754
J755
J756
J757
J758
J759
J760
J761
J762
J763
J764
J765
J766
J767
J768
J769
J770
J771
J772
J773
J774
J775
J776
J777
J778
J779
J780
J781
J782
J783
J784
J785
J786
J787
J788
J789
J790
J791
J792
J793
J794
J795
J796
J797
J798
J799
J800
J801
J802
J803
J804
J805
J806
J807
J808
J809
J810
J811
J812
J813
J814
J815
J816
J817
J818
J819
J820
J821
J822
J823
J824
J825
J826
J827
J828
J829
J830
J831
J832
J833
J834
J835
J836
J837
J838
J839
J840
J841
J842
J843
J844
J845
J846
J847
J848
J849
J850
J851
J852
J853
J854
J855
J856
J857
J858
J859
J860
J861
J862
J863
J864
J865
J866
J867
J868
J869
J870
J871
J872
J873
J874
J875
J876
J877
J878
J879
J880
J881
J882
J883
J884
J885
J886
J887
J888
J889
J890
J891
J892
J893
J894
J895
J896
J897
J898
J899
J900
J901
J902
J903
J904
J905
J906
J907
J908
J909
J910
J911
J912
J913
J914
J915
J916
J917
J918
J919
J920
J921
J922
J923
J924
J925
J926
J927
J928
J929
J930
J931
J932
J933
J934
J935
J936
J937
J938
J939
J940
J941
J942
J943
J944
J945
J946
J947
J948
J949
J950
J951
J952
J953
J954
J955
J956
J957
J958
J959
J960
J961
J962
J963
J964
J965
J966
J967
J968
J969
J970
J971
J972
J973
J974
J975
J976
J977
J978
J979
J980
J981
J982
J983
J984
J985
J986
J987
J988
J989
J990
J991
J992
J993
J994
J995
J996
J997
J998
J999
J1000
J1001
J1002
J1003
J1004
J1005
J1006
J1007
J1008
J1009
J1010
J1011
J1012
J1013
J1014
J1015
J1016
J1017
J1018
J1019
J1020
J1021
J1022
J1023
J1024
J1025
J1026
J1027
J1028
J1029
J1030
J1031
J1032
J1033
J1034
J1035
J1036
J1037
J1038
J1039
J1040
J1041
J1042
J1043
J1044
J1045
J1046
J1047
J1048
J1049
J1050
J1051
J1052
J1053
J1054
J1055
J1056
J1057
J1058
J1059
J1060
J1061
J1062
J1063
J1064
J1065
J1066
J1067
J1068
J1069
J1070
J1071
J1072
J1073
J1074
J1075
J1076
J1077
J1078
J1079
J1080
J1081
J1082
J1083
J1084
J1085
J1086
J1087
J1088
J1089
J1090
J1091
J1092
J1093
J1094
J1095
J1096
J1097
J1098
J1099
J1100
J1101
J1102
J1103
J1104
J1105
J1106
J1107
J1108
J1109
J1110
J1111
J1112
J1113
J1114
J1115
J1116
J1117
J1118
J1119
J1120
J1121
J1122
J1123
J1124
J1125
J1126
J1127
J1128
J1129
J1130
J1131
J1132
J1133
J1134
J1135
J1136
J1137
J1138
J1139
J1140
J1141
J1142
J1143
J1144
J1145
J1146
J1147
J1148
J1149
J1150
J1151
J1152
J1153
J1154
J1155
J1156
J1157
J1158
J1159
J1160
J1161
J1162
J1163
J1164
J1165
J1166
J1167
J1168
J1169
J1170
J1171
J1172
J1173
J1174
J1175
J1176
J1177
J1178
J1179
J1180
J1181
J1182
J1183
J1184
J1185
J1186
J1187
J1188
J1189
J1190
J1191
J1192
J1193
J1194
J1195
J1196
J1197
J1198
J1199
J1200
J1201
J1202
J1203
J1204
J1205
J1206
J1207
J1208
J1209
J1210
J1211
J1212
J1213
J1214
J1215
J1216
J1217
J1218
J1219
J1220
J1221
J1222
J1223
J1224
J1225
J1226
J1227
J1228
J1229
J1230
J1231
J1232
J1233
J1234
J1235
J1236
J1237
J1238
J1239
J1240
J1241
J1242
J1243
J1244
J1245
J1246
J1247
J1248
J1249
J1250
J1251
J1252
J1253
J1254
J1255
J1256
J1257
J1258
J1259
J1260
J1261
J1262
J1263
J1264
J1265
J1266
J1267
J1268
J1269
J1270
J1271
J1272
J1273
J1274
J1275
J1276
J1277
J1278
J1279
J1280
J1281
J1282
J1283
J1284
J1285
J1286
J1287
J1288
J1289
J1290
J1291
J1292
J1293
J1294
J1295
J1296
J1297
J1298
J1299
J1300
J1301
J1302
J1303
J1304
J1305
J1306
J1307
J1308
J1309
J1310
J1311
J1312
J1313
J1314
J1315
J1316
J1317
J1318
J1319
J1320
J1321
J1322
J1323
J1324
J1325
J1326
J1327
J1328
J1329
J1330
J1331
J1332
J1333
J1334
J1335
J1336
J1337
J1338
J1339
J1340
J1341
J1342
J1343
J1344
J1345
J1346
J1347
J1348
J1349
J1350
J1351
J1352
J1353
J1354
J1355
J1356
J1357
J1358
J1359
J1360
J1361
J1362
J1363
J1364
J1365
J1366
J1367
J1368
J1369
J1370
J1371
J1372
J1373
J1374
J1375
J1376
J1377
J1378
J1379
J1380
J1381
J1382
J1383
J1384
J1385
J1386
J1387
J1388
J1389
J1390
J1391
J1392
J1393
J1394
J1395
J1396
J1397
J1398
J1399
J1400
J1401
J1402
J1403
J1404
J1405
J1406
J1407
J1408
J1409
J1410
J1411
J1412
J1413
J1414
J1415
J1416
J1417
J1418
J1419
J1420
J1421
J1422
J1423
J1424
J1425
J1426
J1427
J1428
J1429
J1430
J1431
J1432
J1433
J1434
J1435
J1436
J1437
J1438
J1439
J1440
J1441
J1442
J1443
J1444
J1445
J1446
J1447
J1448
J1449
J1450
J1451
J1452
J1453
J1454
J1455
J1456
J1457
J1458
J1459
J1460
J1461
J1462
J1463
J1464
J1465
J1466
J1467
J1468
J1469
J1470
J1471
J1472
J1473
J1474
J1475
J1476
J1477
J1478
J1479
J1480
J1481
J1482
J1483
J1484
J1485
J1486
J1487
J1488
J1489
J1490
J1491
J1492
J1493
J1494
J1495
J1496
J1497
J1498
J1499
J1500
J1501
J1502
J1503
J1504
J1505
J1506
J1507
J1508
J1509
J1510
J1511
J1512
J1513
J1514
J1515
J1516
J1517
J1518
J1519
J1520
J1521
J1522
J1523
J1524
J1525
J1526
J1527
J1528
J1529
J1530
J1531
J1532
J1533
J1534
J1535
J1536
J1537
J1538
J1539
J1540
J1541
J1542
J1543
J1544
J1545
J1546
J1547
J1548
J1549
J1550
J1551
J1552
J1553
J1554
J1555
J1556
J1557
J1558
J1559
J1560
J1561
J1562
J1563
J1564
J1565
J1566
J1567
J1568
J1569
J1570
J1571
J1572
J1573
J1574
J1575
J1576
J1577
J1578
J1579
J1580
J1581
J1582
J1583
J1584
J1585
J1586
J1587
J1588
J1589
J1590
J1591
J1592
J1593
J1594
J1595
J1596
J1597
J1598
J1599
J1600
J1601
J1602
J1603
J1604
J1605
J1606
J1607
J1608
J1609

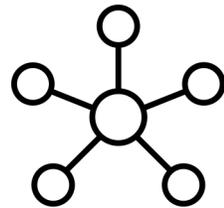
Why are virtual machines special?



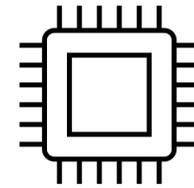
lots of low-level
platform-specific
systems code



multiple interacting
dynamic
components



Lua



intensive
allocation and
garbage collection



runtime code
generation

2. Multiple interacting components

- compartmentalization (c18n) is **lightweight isolation**
- hybrid code enables DDC-based isolation
- need a **compartment switcher**
- need a **libc** per compartment
- need clever tricks to handle dynamic loading
- overhead - how small should each compartment be?
 - compartment per *function*
 - compartment per *shared object*
 - alternative compartment boundaries?

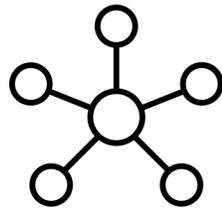
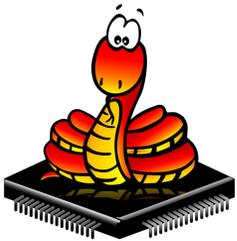
Alternative c18n strategy

- For purecap MicroPython code
- We isolate at FFI boundaries
 - e.g. calls to external C libraries
 - (work in progress)

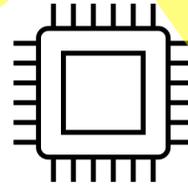
Why are virtual machines special?



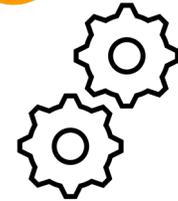
lots of low-level
platform-specific
systems code



multiple interacting
dynamic
components



intensive
allocation and
garbage collection



runtime code
generation

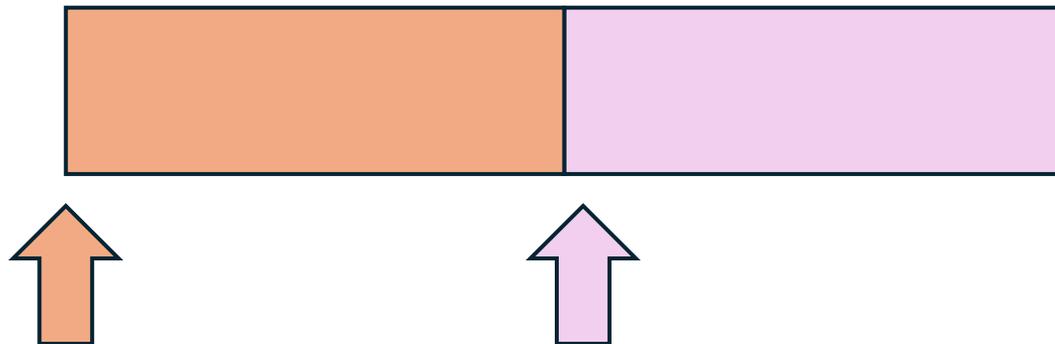
3. Malloc and GC

Complications include:

- finding and tracing the root set
 - scanning the full heap
 - moving objects
-
- We have studied **BDWGC**
 - Morello & RISC-V: github.com/capablevms/bdwgc

Observations about purecap GC

- Can't afford to lose capability tags
- conservative -> precise
- overhead reduction!
- issue with sealed caps in userspace code
- issue with coalescing



```
int main() {
    int i, size, *buffer;

    srand(SEED);

    for (i=0; i<NUM_ALLOCS; i++) {
        size = rand() % 1024;
        // printf("alloc buffer size %d\n", size);
        buffer = (int *)malloc(size * sizeof(int));

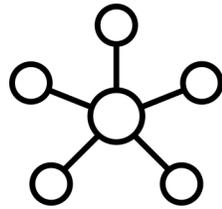
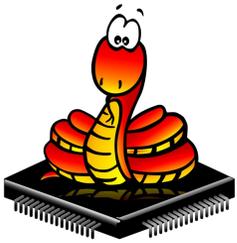
        // confuse conservative collector?
        savebuffer[i%(NUM_ALLOCS>>5)] = (long int)buffer;
    }

    return 0;
}
```

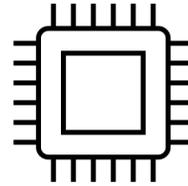
Why are virtual machines special?



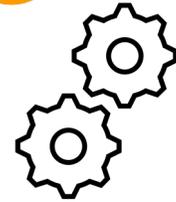
lots of low-level
platform-specific
systems code



multiple interacting
dynamic
components



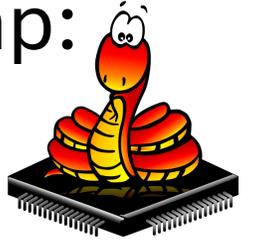
intensive
allocation and
garbage collection



runtime code
generation

4. Runtime code generation

- Several baseline interpreters ported to Morello purecap: WARDuino, MicroPython, JSC
- Some investigations on runtime code generation: JSC (& v8)
- This is work-in-progress



Summary

- Our Capable VMs project has demonstrated that

CHERI *does* provide defence-in-depth against VM-based exploits

Challenges include:

1. how to quantify additional **defence**?
2. how to measure **performance**?
3. how to encourage **adoption**?