

Analyzing Raw Binary Images in Ghidra

A Look at Cisco IOS

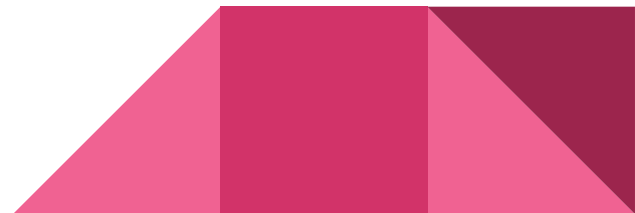
Who am I?

Nick Starke - Security Researcher located in Des Moines.

Works for Aruba Networks/HPE as a Threat Researcher / Reverse Engineer

Likes building and breaking

Bugcrowd MVP



What is Ghidra?

Reverse Engineer / Binary Analysis Platform built by NSA Research Directorate

Released in early 2019

Competitor (in some ways) to IDA Pro, Binary Ninja, and Radare2

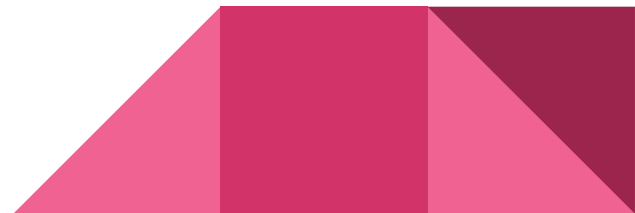


The Story so far...

In Mid 2018 I bought a Cisco Catalyst 3750 off eBay.

I was gonna hack a switch.

I had no idea what I was doing.

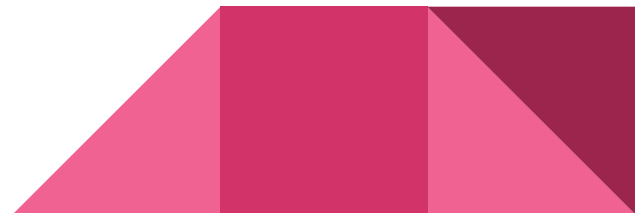


Firmware File Layout

First 112 (0x70) Bytes are a proprietary header

Firmware image begins as index 112.

How did I figure that out?



Binwalk

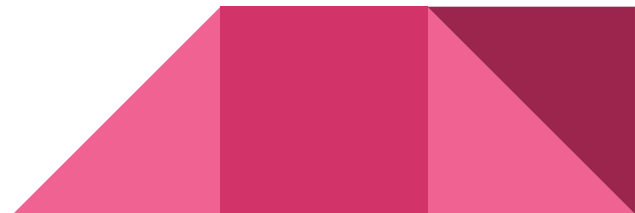
<https://github.com/refirmlabs/binwalk>

Run

```
$ binwalk -eM firmware_file.bin
```

-e for auto extract

-M for recursive



Bzip2 Data at 0x70

```
(~/Documents/research/cisco-catalyst-3750)
└─> binwalk c3750-ipervicesk9-mz.122-50.SE3.bin
```

DECIMAL	HEXADECIMAL	DESCRIPTION
112	0x70	bzip2 compressed data, block size = 900k

Now that we have the bunzipped image...

We need to figure out what CPU architecture the image is built for:

```
└─> binwalk -m /usr/local/lib/python2.7/dist-packages/binwalk/magic/binarch_c3750-ipservicesk9-mz.122-50.SE3.0002.bin.extract  
ed/70
```

DECIMAL	HEXADECIMAL	DESCRIPTION
24	0x18	PowerPC big endian instructions, function prologue
1360	0x550	PowerPC big endian instructions, function epilogue
1364	0x554	PowerPC big endian instructions, function epilogue
1372	0x55C	PowerPC big endian instructions, function epilogue
1380	0x564	PowerPC big endian instructions, function epilogue
1388	0x56C	PowerPC big endian instructions, function prologue
1612	0x64C	PowerPC big endian instructions, function epilogue
1648	0x670	PowerPC big endian instructions, function epilogue
1656	0x678	PowerPC big endian instructions, function prologue
3224	0xC98	PowerPC big endian instructions, function epilogue
3232	0xCA0	PowerPC big endian instructions, function prologue
6772	0x1A74	PowerPC big endian instructions, function epilogue
6780	0x1A7C	PowerPC big endian instructions, function prologue

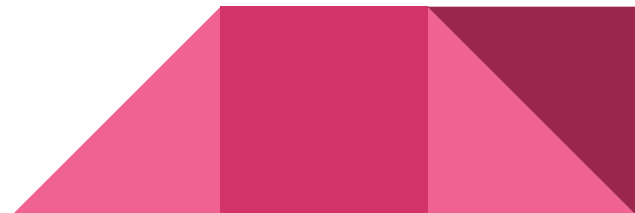
Binwalk magic mode

Binwalk's magic numbers can be reconfigured.

Binwalk comes with a magic definition file called "binarch"

We use this file with the "-m" flag to produce the aforementioned output

Clearly, it is likely this is a Big Endian PowerPC firmware image.



Finding Offset values

A Switch OS does not start at memory address 0x00000000

We must go deeper...



Show version

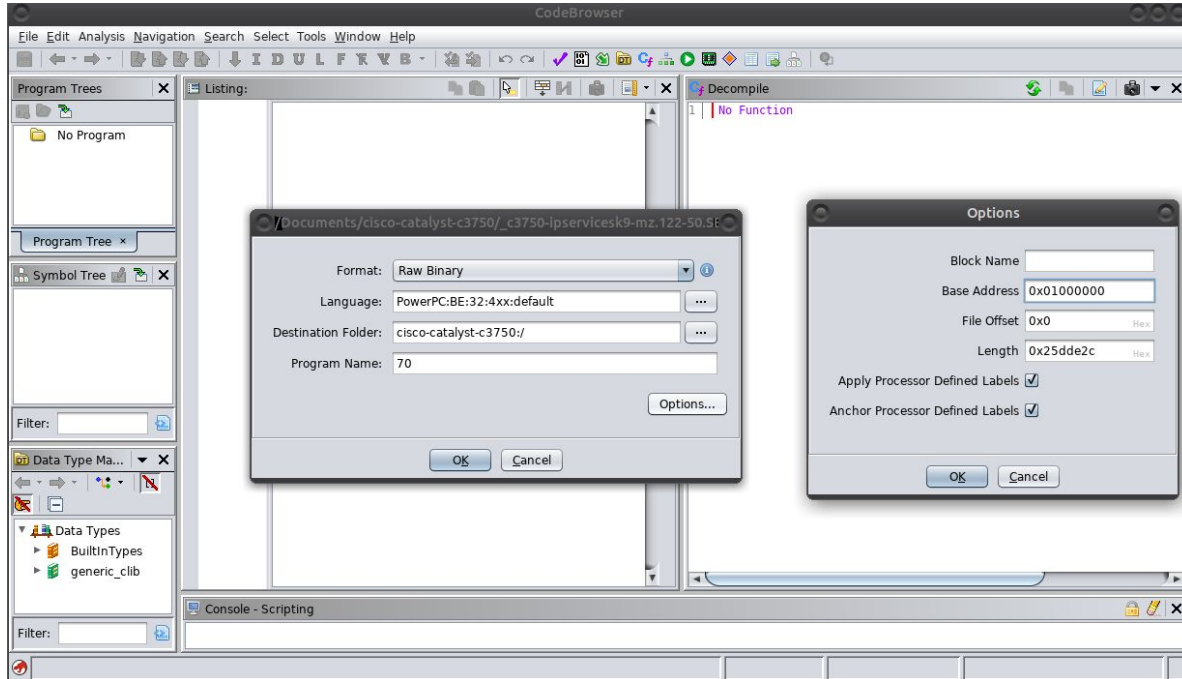
From the serial or ssh console, we run `show version`

That gives us an OS load address of 0x01000000

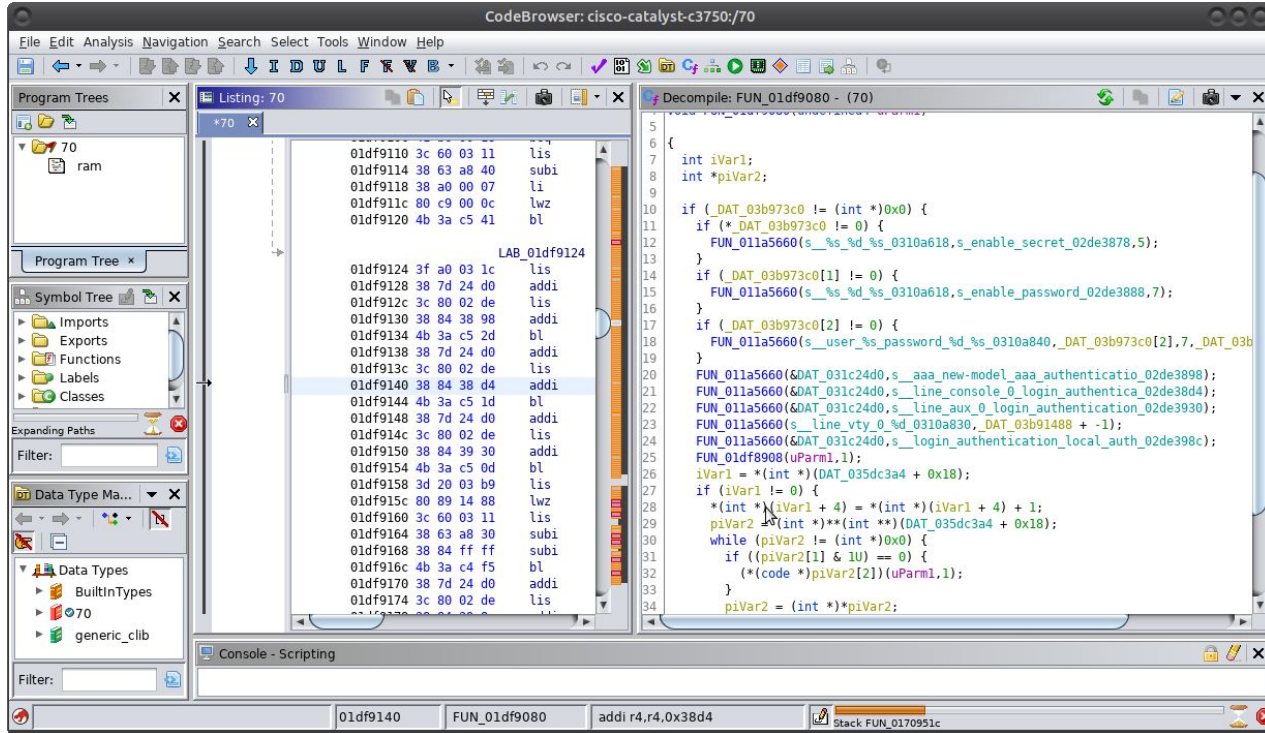
```
Switchy>show version
Cisco IOS Software, C3750 Software (C3750-IPSERVICESK9-M), Version 12.2(55)SE, RELEASE SOFTWARE (fc2)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2010 by Cisco Systems, Inc.
Compiled Sat 07-Aug-10 22:45 by prod_rel_team
Image text-base: 0x01000000, data-base: 0x02F00000
```

Time to Fire up Ghidra!

We need to load our b unzipped firmware image into Ghidra:



Ghidra will Churn



Setting the Data-base

The screenshot shows the CodeBrowser interface for a file named 'cisco-catalyst-c3750:770'. A 'Memory Map' dialog box is open, displaying a table of memory blocks. The table has columns for Name, Start, End, Length, R, W, X, Volatility, Type, Initial value, Source, and Comment. The first row is highlighted, showing a 'ram' block starting at 01000000 and ending at 035dde2b, with a length of 0x25dde2c. The background shows a decompiled function 'FUN_01df9080' with a warning about global symbols and some assembly-like code.

CodeBrowser: cisco-catalyst-c3750:770

File Edit Analysis Navigation Search Select Tools Window Help

Program Trees

Listing: 70

Decompile: FUN_01df9080 - (70)

```
2 /* WARNING: Globals starting with '_' overlap smaller symbols at the same address
3
4 void FUN_01df9080(undefined4 uParm1)
5
```

Memory Map [CodeBrowser: cisco-catalyst-c3750:770]

File Edit Tools Help

Memory Map - Image Base: 00000000

Name	Start	End	Length	R	W	X	Volat...	Type	Initial...	Source	Comment
ram	01000000	035dde2b	0x25dde2c	✓	✓	✓	☐	Default	✓	/home/nick/D...	fileOffset=0, l...

Symbol Tree

- FUN_01130480
- FUN_01130718
- FUN_01130784
- FUN_011307c0
- FUN_01130878
- FUN_01130b18
- FUN_01130b58

Filter:

Data Type Ma...

Data Types

- BuiltInTypes
- 70
- generic_clib

Filter:

```
29 piVar2 = (int *)**((int **)DAT_035dc3a4 + 0x18);
30 while (piVar2 != (int *)0x0) {
31   if ((piVar2[1] & 1U) == 0) {
```

Console - Scripting

035dde2b

Splitting Memory

The screenshot shows the CodeBrowser interface for a Cisco Catalyst device. The main window displays a memory map with a table of memory blocks:

Name	Start	End
ram	01000000	035dde2b

A 'Split Block' dialog box is open, showing the configuration for splitting the 'ram' block into two new blocks:

- Block to Split:**
 - Block Name: ram
 - Start Address: 01000000
 - End Address: 0x02de33d8
 - Block Length: 0x1de33d9
- New Block:**
 - Block Name: ram.split
 - Start Address: 02de33d9
 - End Address: 035dde2b
 - Block Length: 0x7faa53

The background shows a decompiled function 'FUN_01df9080' with the following code:

```
1 /* WARNING: Globals starting with '_' overlap smaller symbols at the
2
3 FUN_01df9080(undefined4 uParm1)
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Splitting Memory Continued (RWX)

The screenshot displays the CodeBrowser interface for a binary named 'cisco-catalyst-c3750/70'. A 'Memory Map' window is overlaid, showing the following memory blocks:

Name	Start	End	Length	R	W	X	Vola...	Type	Initial...	Source	Comment
ram	01000000	02de33d8	0x1de33d9	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Default	<input checked="" type="checkbox"/>	/home/nick/D... fileOffset=0, l...	7);
ram.split	02de33d9	035dde2b	0x7faa53	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Default	<input checked="" type="checkbox"/>		

The decompiled code in the background shows a warning: 'WARNING: Globals starting with '_' overlap smaller symbols at the' and a function definition: 'void FUN_01df9080(undefined4 uParm1)'. The console at the bottom shows the address '02de7678' and the stack frame 'Stack_FUN_026d5544'.

Resolved Strings

The screenshot displays the CodeBrowser interface for a Cisco Catalyst device. The main window shows the decompiled C code for the function `FUN_01df9080`. The code includes several conditional checks and string literals that are resolved to memory addresses. The memory dump on the left shows the resolved strings for these addresses.

Memory Dump (Listing: 70):

Address	Hex	String
02de766c	03	
02de766d	45	
02de766e	85	
02de766f	4c	
02de7670	03	
02de7671	b9	
02de7672	85	
02de7673	f6	
02de7674	00	
02de7675	00	
02de7676	00	
02de7677	00	
02de7678	20 20 20	s
	20 4c 69	
	6e 65 20 ...	
02de76bc	20 20 20	s
	20 4c 69	
	6e 65 20 ...	
02de76de	00	
02de76df	00	

Decompile: FUN_01df9080 - (70)

```
4 void FUN_01df9080(undefined4 uParm1)
5
6 {
7     int *piVar1;
8
9     if (_DAT_03b973c0 != (int *)0x0) {
10        if (*_DAT_03b973c0 != 0) {
11            FUN_011a5660("\n%s %d %s", "enable secret", 5);
12        }
13        if (_DAT_03b973c0[1] != 0) {
14            FUN_011a5660("\n%s %d %s", "enable password", 7);
15        }
16        if (_DAT_03b973c0[2] != 0) {
17            FUN_011a5660("\nuser %s password %d %s", _DAT_03b973c0[2], 7, _DAT_03b973c0[3]);
18        }
19        FUN_011a5660(&DAT_031c24d0, "\naaa new-model\naaa authentication login local_auth local");
20        FUN_011a5660(&DAT_031c24d0,
21
22            "\nline console 0\n login authentication local_auth\n exec-timeout 5 0\n transpor
23            output telnet"
24        );
25        FUN_011a5660(&DAT_031c24d0,
26
27            "\nline aux 0\n login authentication local_auth\n exec-timeout 10 0\n transport
28            output telnet"
29        );
30        FUN_011a5660("\nline vty 0 %d", _DAT_03b91488 + -1);
31        FUN_011a5660(&DAT_031c24d0, "\n login authentication local_auth\n transport input telnet");
32        FUN_01df8908(uParm1, 1);
33        if (_DAT_03af5020 != (int *)0x0) {
34            _DAT_03af5020[1] = _DAT_03af5020[1] + 1;
35        }
36    }
37 }
```

Console - Scripting

02de7678

Stack FUN_0297cf9c

Live Demo

“CISCO KITS”

<https://artkond.com/2017/04/10/cisco-catalyst-remote-code-execution>



Linky

This whole presentation is based on a write up I did:

<https://gist.github.com/nstarke/ed0aba2c882b8b3078747a567ee00520>



Questions?

Thank you!

Contact:

@nstarke on secdsm slack

@nstarke on twitter

@nstarke on github

