Control Flow Bending:
On the Effectiveness of Control Flow Integrity

Nicholas Carlini[1], Antonio Barresi[2], Mathias Payer[3]
David Wagner[1], Thomas R. Gross[2]

[1] University of California, Berkeley
[2] ETH Zurich
[3] Purdue University

Background

Background

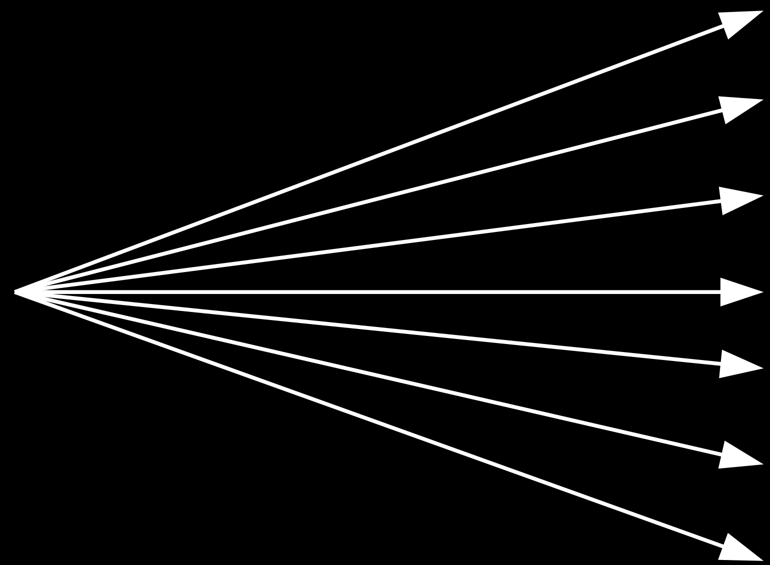
Memory Corruption → Control Flow Integrity

Main Result:
CFI does not stop all attacks.

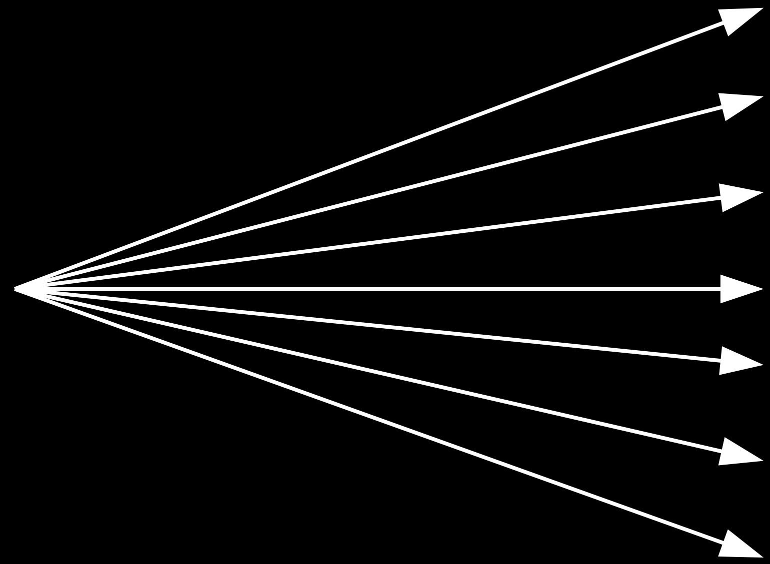
Control-Flow Integrity

Control-Flow Integrity

```
(*fn)(x);
```



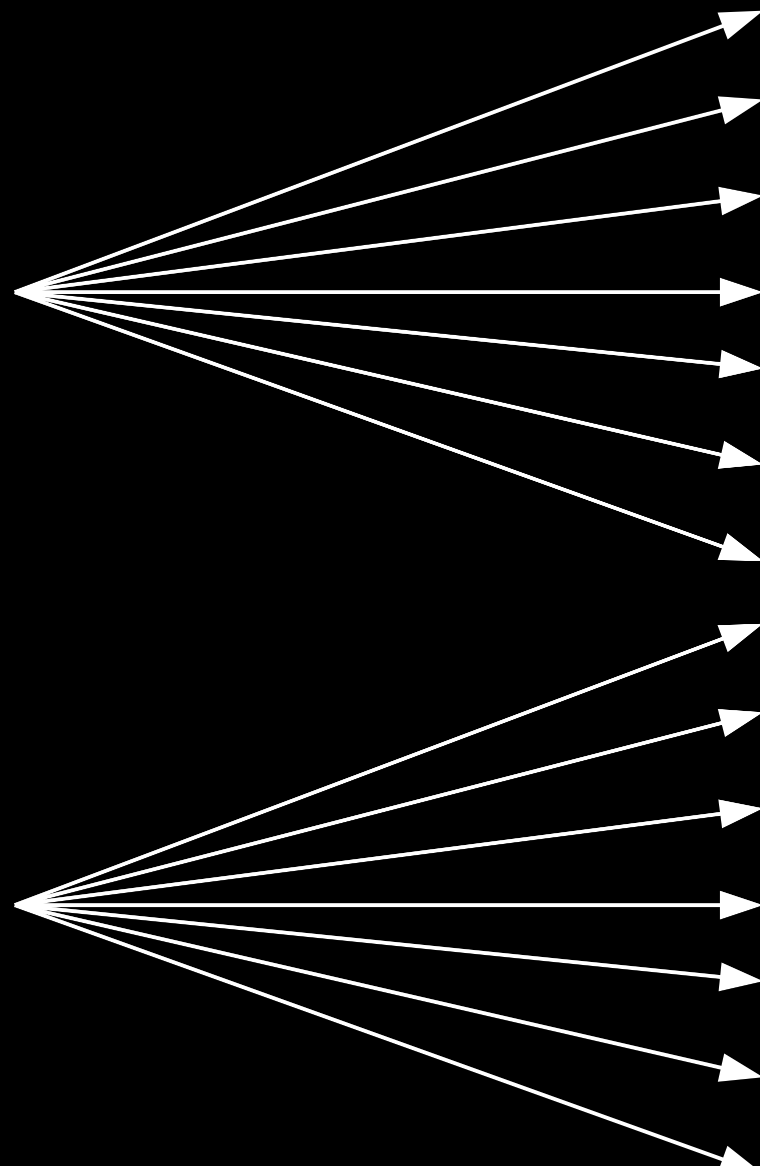
```
return 7;
```



Control-Flow Integrity

```
CHECK(fn);
(*fn)(x);
```

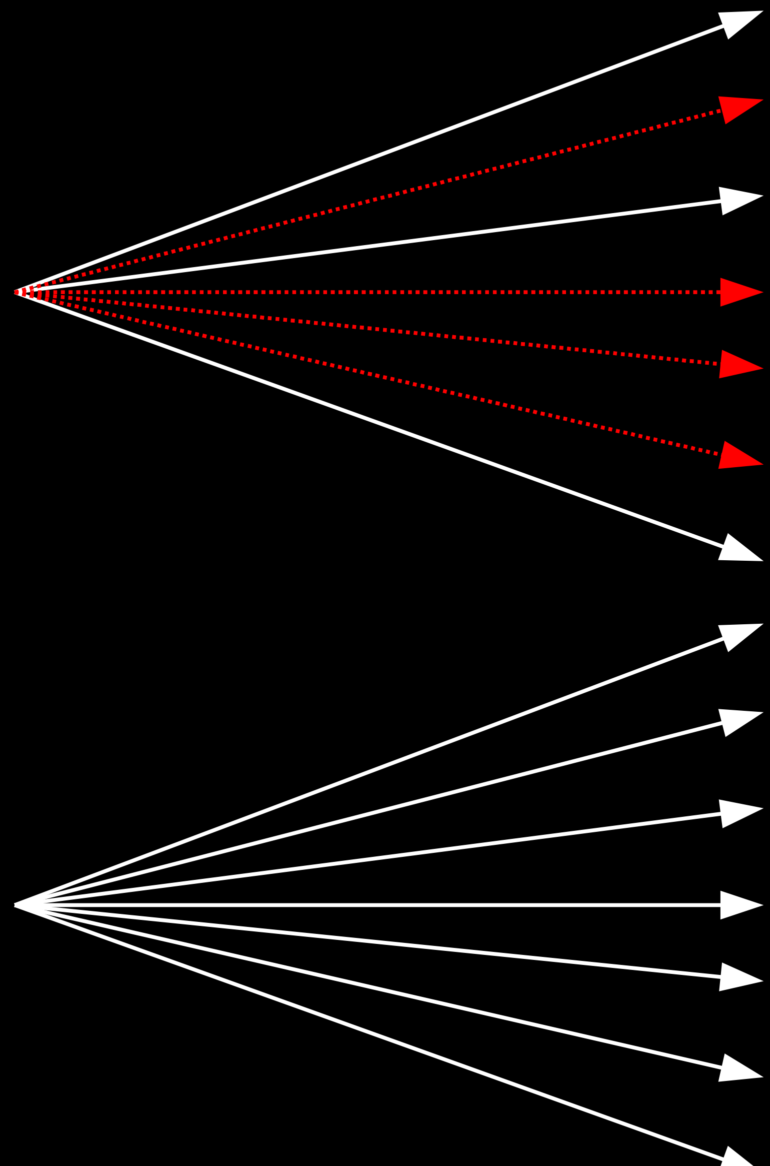
```
return 7;
```



Control-Flow Integrity

```
CHECK(fn);
(*fn)(x);
```

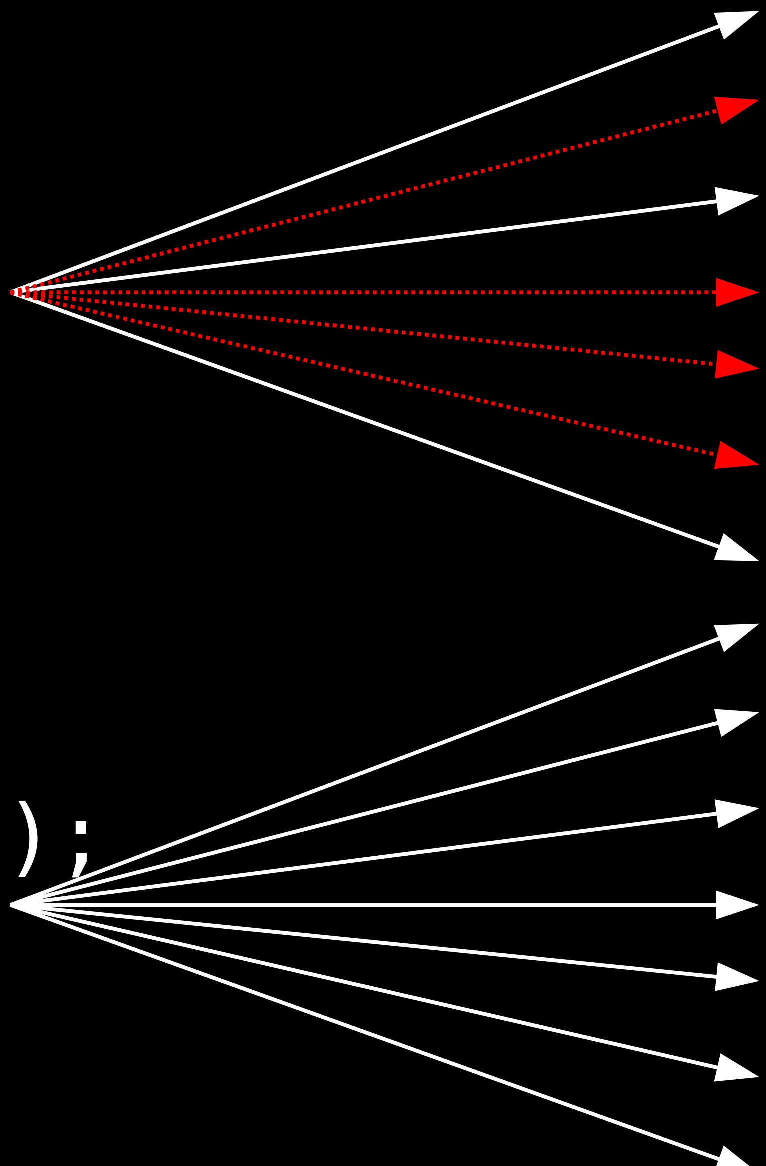
```
return 7;
```



Control-Flow Integrity

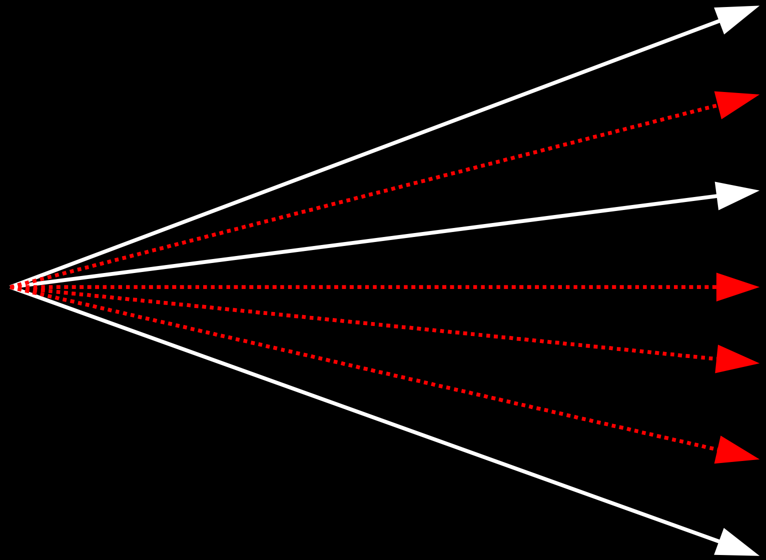
```
CHECK(fn);
(*fn)(x);
```

```
CHECK_RET();
return 7;
```

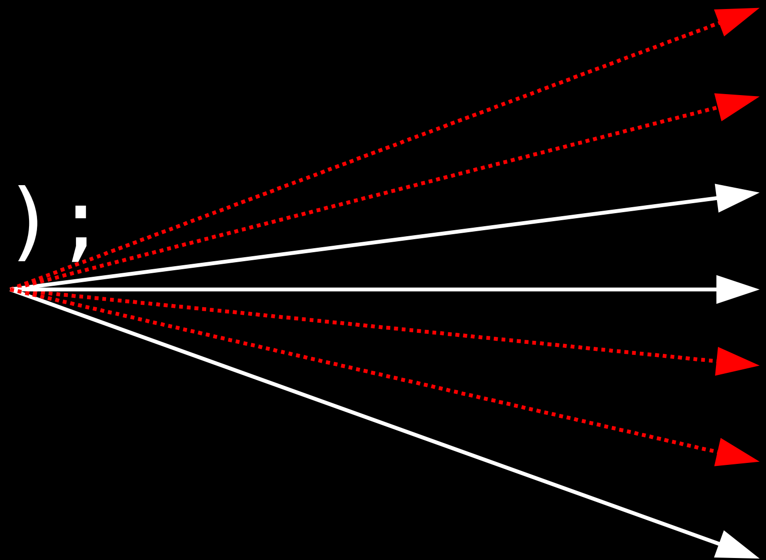


Control-Flow Integrity

```
CHECK(fn);
(*fn)(x);
```



```
CHECK_RET();
return 7;
```

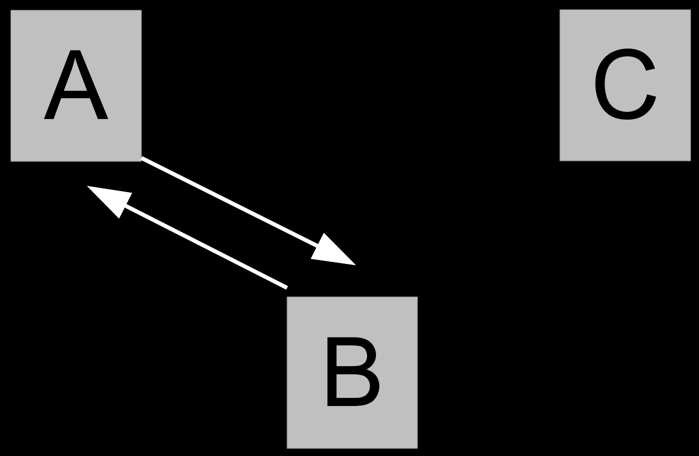


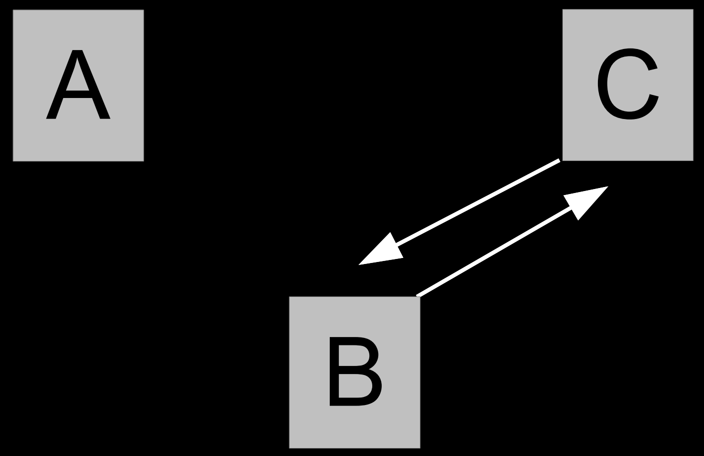
Shadow Stacks

- Dynamic restrictions on return instructions

Shadow Stacks

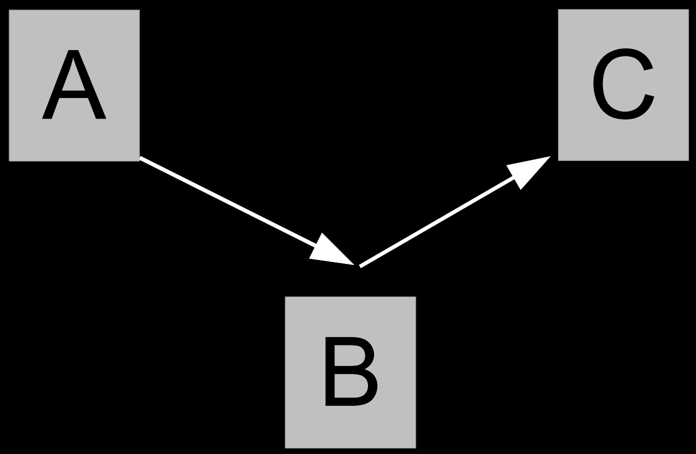
- Dynamic restrictions on return instructions





Shadow Stacks

- Dynamic restrictions on return instructions



Two ways to weaken CFI:

(1) Coarsen CFG
(2) Remove Shadow Stack

Prior Work: Weak CFI is broken

- Göktaş *et al.* IEEE S&P '14
  "Out Of Control: Overcoming Control-Flow Integrity"
- Carlini *et al.* USENIX Security '14
  "ROP is Still Dangerous: Breaking Modern Defenses"
- Davi *et al.* USENIX Security '14
  "Stitching the Gadgets: On the Ineffectiveness of Coarse-Grained Control-Flow Integrity Protection"
- Göktaş *et al*. USENIX Security '14
  "Size Does Matter: Why Using Gadget-Chain Length to Prevent Code-Reuse Attacks is Hard"

Our question:
How secure are other CFI variants?

Our question:
How secure are other CFI variants?

1. CFI with no shadow stack
2. Fully unweakened CFI

Fully-Precise Static CFI:
CFI in its best possible form

A transfer from X to Y is allowed
if and only if
some benign execution uses it

How secure is half-weakened CFI?

(Shadow stack removed)

Can we just do ROP?

Can we just do ROP?

No.

Can we just do ROP?

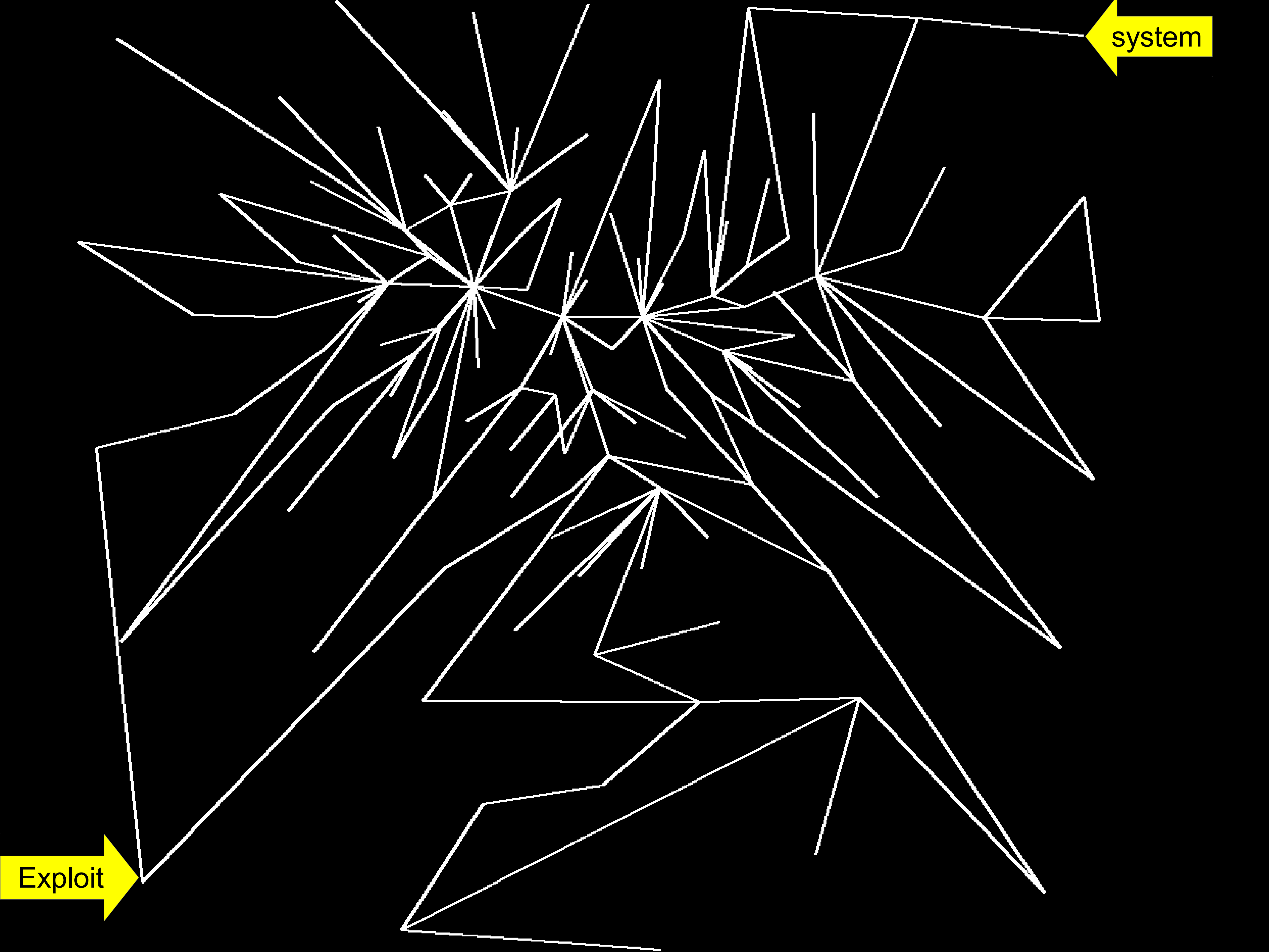
No.

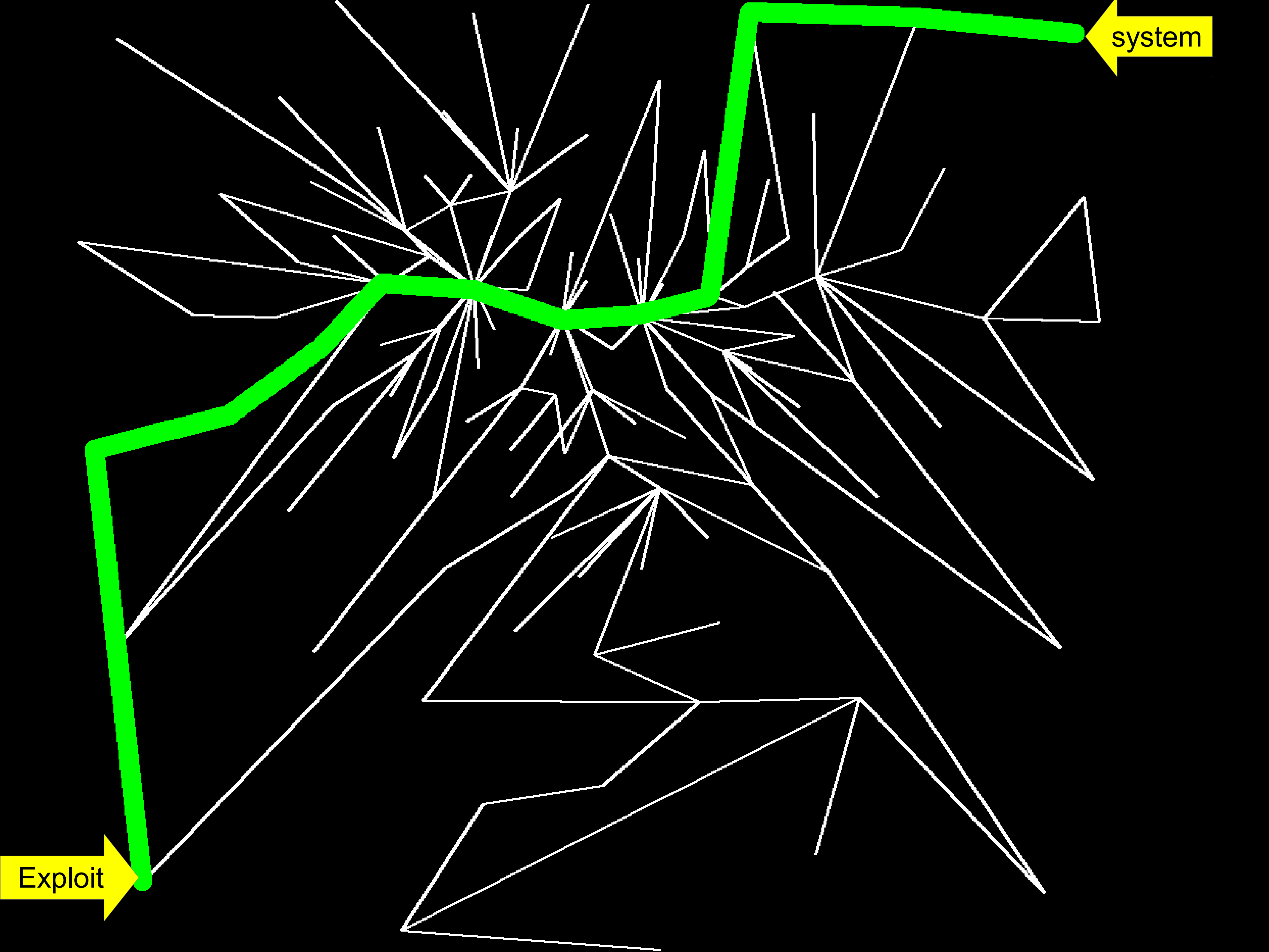
How about return-to-libc?

Return to Libc: Challenges

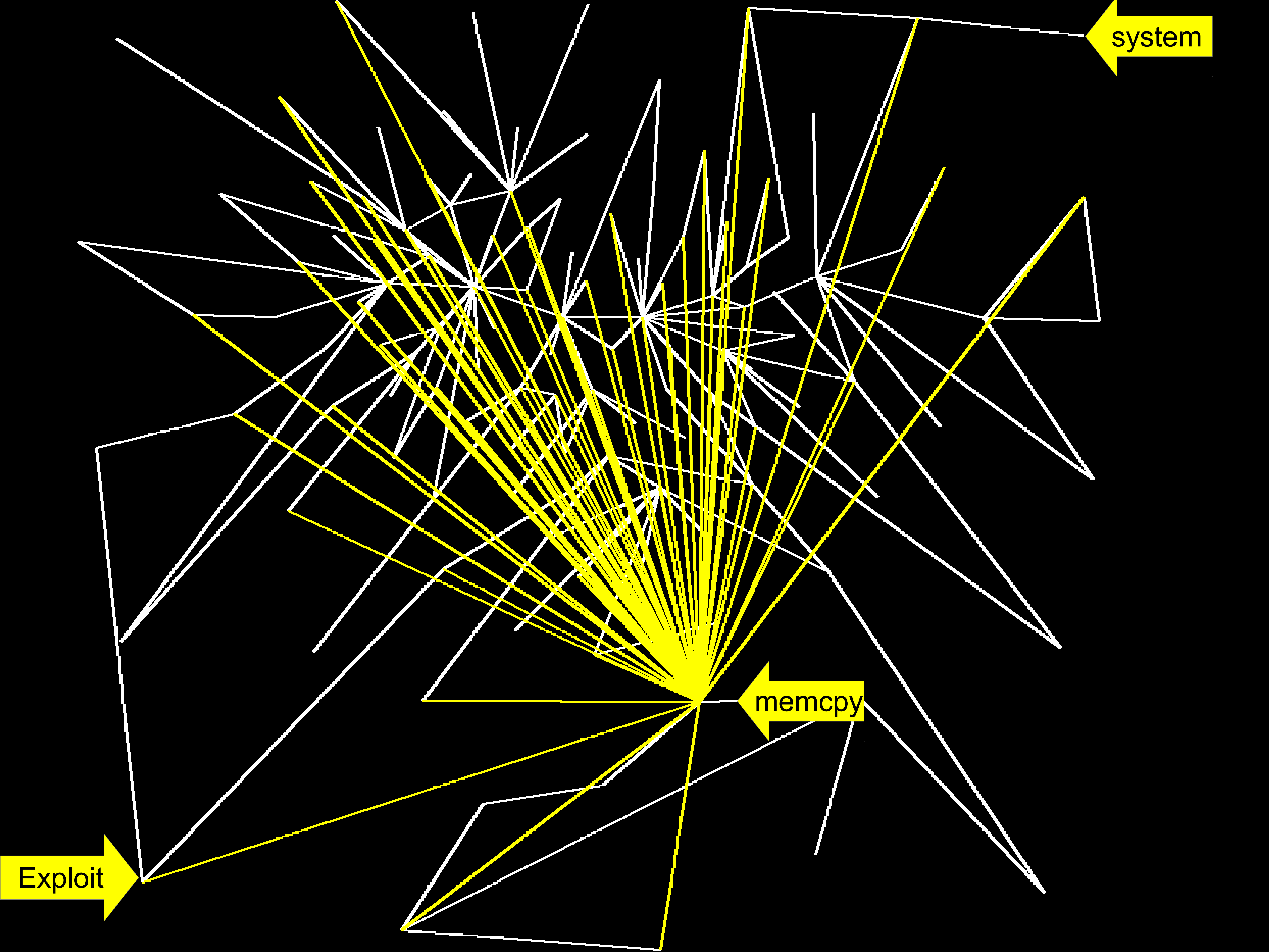
1. Find path to `system()` in CFG.
2. Divert control flow down this path.
3. Control `system()` arguments.

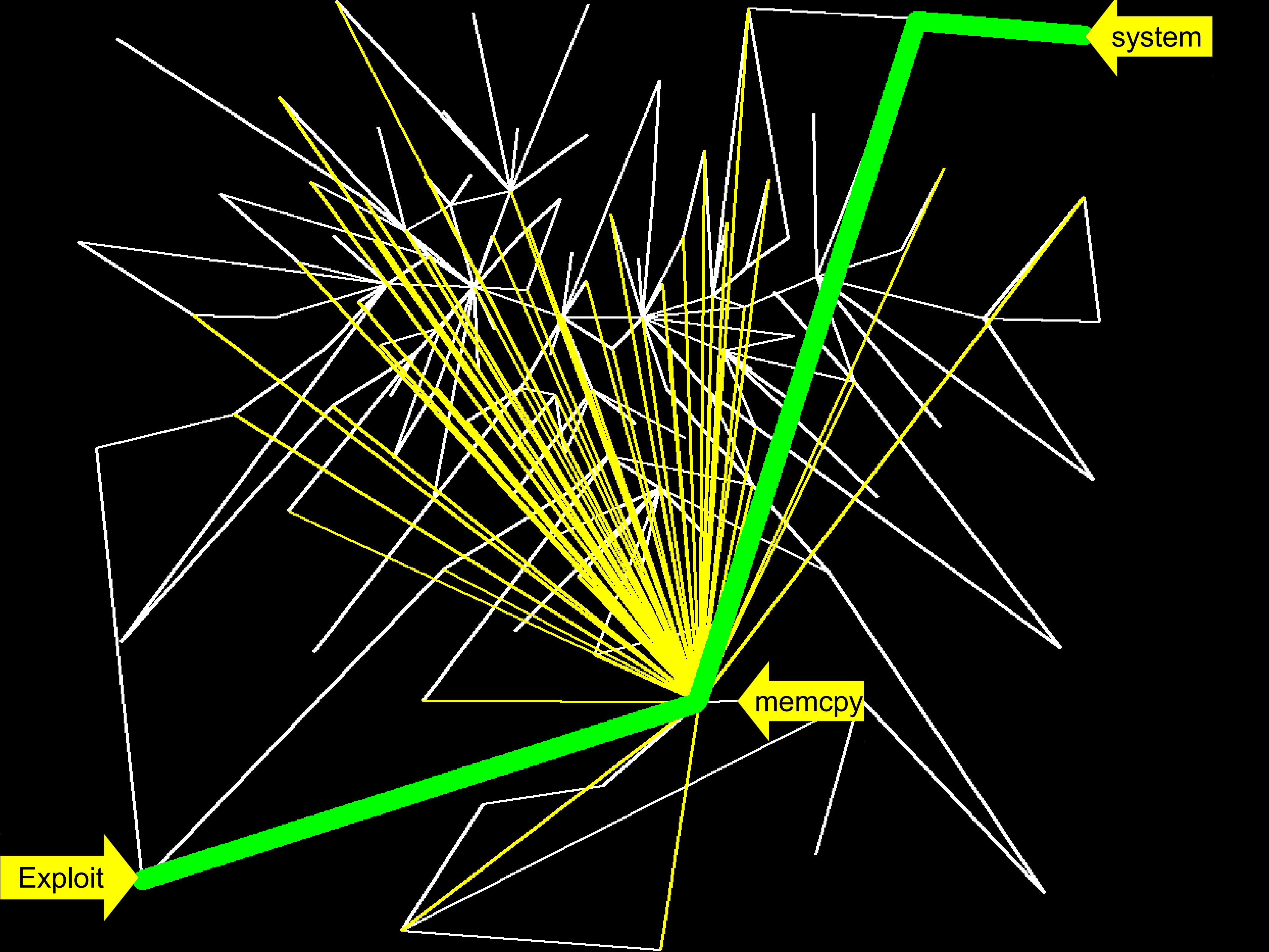
What does a CFG look like?


system
Exploit


system
Exploit

That picture is a lie


system
memcpy
Exploit

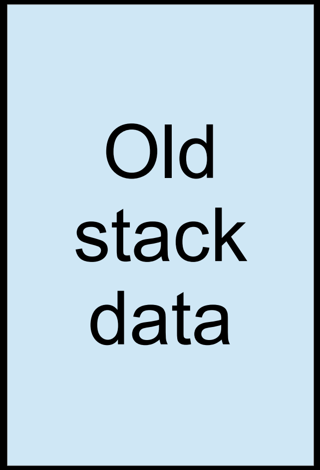

system
memcpy
Exploit

So simple paths between arbitrary points exist. But are they feasible?

Dispatcher Functions

memcpy(dst, src, 8)

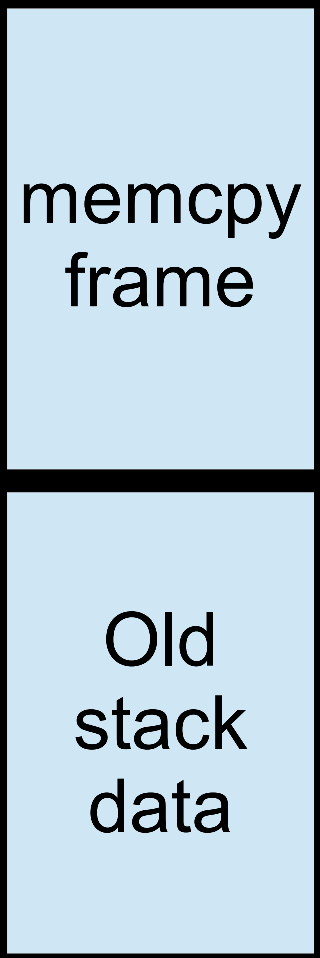
Stack



Dispatcher Functions

memcpy(dst, src, 8)

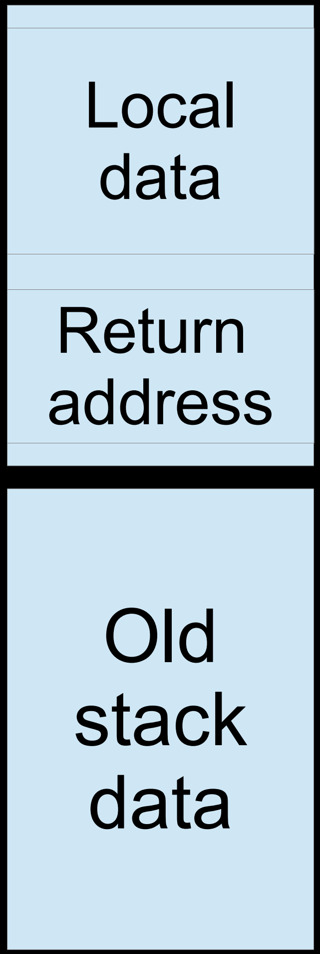
Stack



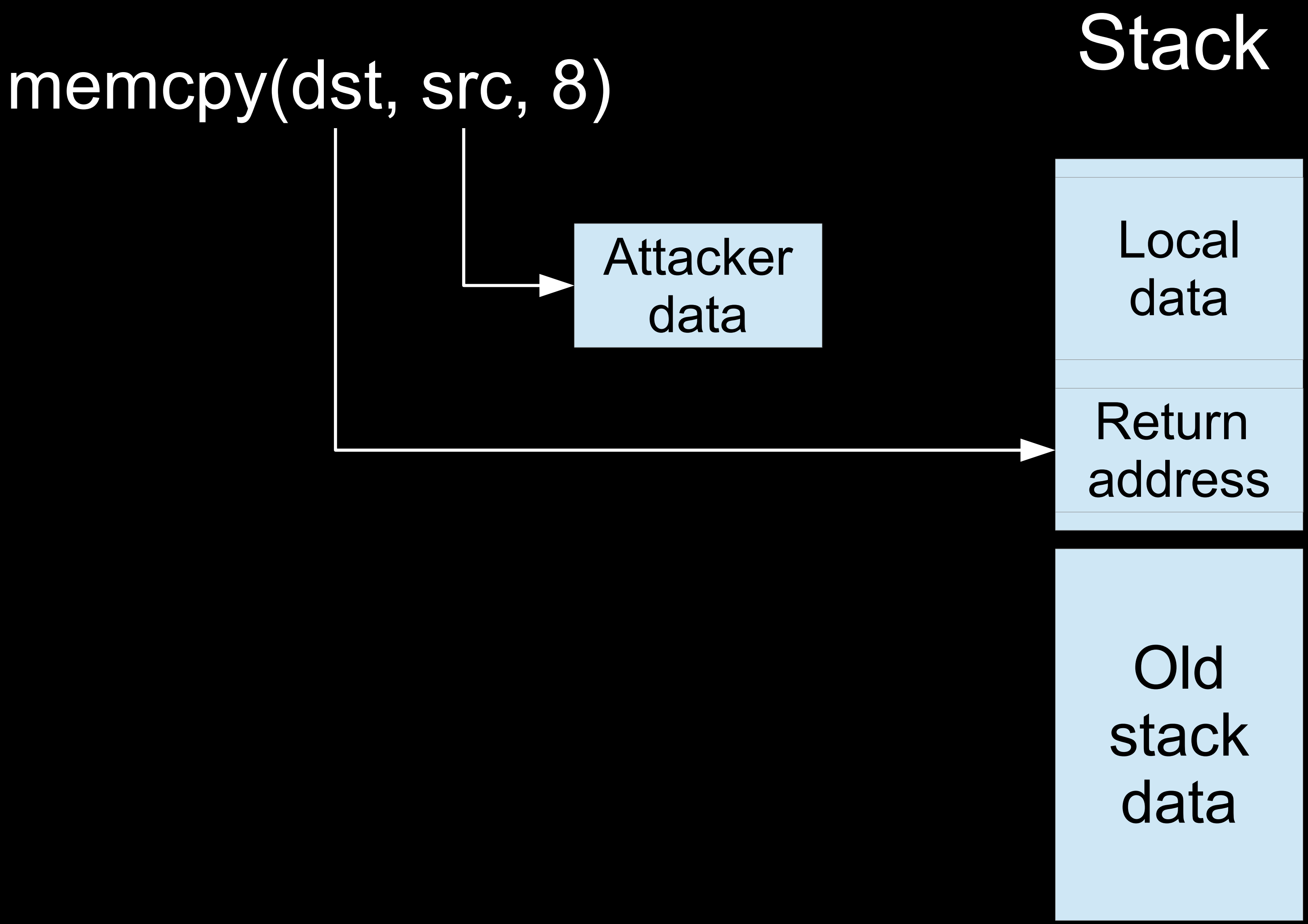
Dispatcher Functions

memcpy(dst, src, 8)

Stack



Dispatcher Functions



Dispatcher Functions

1. Commonly called
2. Arguments under attacker control
3. Can overwrite its own return address

Return to Libc: Challenges

1. Find path to `system()` in CFG. ✔
2. Divert control flow down this path. ✔
3. Control `system()` arguments. ✔

Evaluation (part 1)

- Analyzed six vulnerable binaries, assuming fully-precise static CFI was in place.
- What attacks are possible?

- Direct arbitrary code execution in 3 of the 6
- File system access in 2 of the 6
- Attack prevented in 1 of the 6

CFI without shadow stack is broken – even with fully precise static CFI.

How secure is un-weakened CFI?

This time: no dispatcher functions

Return to Libc: Challenges

1. Find path to `system()` in CFG.

2. Divert control flow down this path.

3. Control `system()` arguments.

Evaluation (part 2)

- Evaluate same 6 binaries; with shadow stack

- Direct arbitrary code execution in 1 of the 6
- File system access in 2 of the 6
- Confined code execution in 2 of the 6
- Attack prevented in 1 of the 6

ROP gives us Turing-complete attacks. Can we do this with CFI and a shadow stack?

Printf-Oriented Programming

- A Turing complete domain-specific language
- Program → Format String
- Program Counter → Format String Counter

Printf-Oriented Programming

- Memory Reads → %s
- Memory Writes → %n
- Conditional → %.*d
- Loops? Write over the format specifier counter.

```
printf("%.0d%.0d%.0d%.0d%.0d%.0d%.0d%.0da%hhn%hhn%sa%hhn%hhn%sa%hhn%hhn%sa%hhn%hhn%s%hhn%hhn%sa%hhn%hhn%sa%hhn%hhn%sa%hhn%hhn%s%hhn%hhn%sa%hhn%hhn%sa%hhn%hhn%sa%hhn
hhn%sa%hhn%hhn%s%s%s%.*d%hhn%hhn%s%s%.*d%hhn%hhn%s%s%hhn%hhn%s%s%s%.*d%hhn%hhn%s%s%.*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s
hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%s%s%s%.*d%hhn%h
%.*d%hhn%hhn%s%s%.*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s
hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%s%s%s%.*d%hhn%hhn%s%s%.*d%hhn%hhn%s%s%hhn%hhn%s%s%s%.*d%hhn%hhn%s%s%.*d%hhn%hhn%s
hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%
%hhn%s%s%hhn%hhn%s%s%s%.*d%hhn%hhn%s%s%.*d%hhn%hhn%s%s%hhn%hhn%s%s%s%.*d%hhn%hhn%s%s%.*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s
hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%s%s%s%.*d%hhn%h
%.*d%hhn%hhn%s%s%.*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s
hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%s%s%s%.*d%hhn%hhn%s%s%.*d%hhn%hhn%s%s%hhn%hhn%s%s%s%.*d%hhn%hhn%s%s%.*d%hhn%hhn%s
hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%
%hhn%s%s%hhn%hhn%s%s%s%.*d%hhn%hhn%s%s%.*d%hhn%hhn%s%s%hhn%hhn%s%s%s%.*d%hhn%hhn%s%s%.*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s
hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%s%s%s%.*d%hhn%h
%.*d%hhn%hhn%s%s%.*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s
hhn%s%s%s%hhn%hhn%s%s%hhn%hhn%s%s%*d%hhn%hhn%s%s%hhn%hhn%s%s%s%hhn%hhn%s%s%hhn", 0,0,0,0,0,0,0,0, &var[32], &args[2], pad, &var[33], &args[5], pad, &var[34], &args[
&var[36], &args[14], pad, &var[37], &args[17], pad, &var[38], &args[20], pad, &var[39], &args[23], pad, &var[64], &args[26], pad, &var[65], &args[29], pad, &var[66]
&args[35], pad, &var[68], &args[38], pad, &var[69], &args[41], pad, &var[70], &args[44], pad, &var[71], &args[47], pad, &var[300], &var[32], 255, 0, &var[999], &arg
&var[999], &args[60], pad, &var[999], &var[996], &args[64], pad, &var[996], &var[64], 255, 0, &var[999], &args[71], pad, &var[999], 255, 0, &var[999], &args[77], pa
pad, &var[996], 255, 0, &var[999], &args[87], pad, &var[999], &var[998], &args[91], pad, &var[64], 255, 0, &var[999], &args[97], pad, &var[999], &var[997], &args[10
&var[999], &args[106], pad, &var[999], &var[999], &args[110], pad, &var[999], 255, 0, &var[999], &args[116], pad, &var[999], &var[193], &args[120], pad, &var[300],
&var[999], &var[998], &args[130], pad, &var[32], 255, 0, &var[999], &args[136], pad, &var[999], &var[997], &args[140], pad, &var[998], &var[997], &var[999], &args[1
&args[149], pad, &var[999], 255, 0, &var[999], &args[155], pad, &var[999], &var[194], &args[159], pad, &var[193], &var[194], &var[999], &args[164], pad, &var[999],
&var[33], 255, 0, &var[999], &args[175], pad, &var[999], 255, 0, &var[999], &args[181], pad, &var[999], &var[996], &args[185], pad, &var[996], &var[65], 255, 0, &va
255, 0, &var[999], &args[198], pad, &var[999], &var[1], &args[202], pad, &var[996], 255, 0, &var[999], &args[208], pad, &var[999], &var[998], &args[212], pad, &var[
pad, &var[999], &var[997], &args[222], pad, &var[998], &var[997], &var[999], &args[227], pad, &var[999], &var[999], &args[231], pad, &var[999], 255, 0, &var[999], &
&var[193], &args[241], pad, &var[96], 255, 0, &var[999], &args[247], pad, &var[999], &var[998], &args[251], pad, &var[33], 255, 0, &var[999], &args[257], pad, &var[
&var[998], &var[997], &var[999], &args[266], pad, &var[999], &var[999], &args[270], pad, &var[999], 255, 0, &var[999], &args[276], pad, &var[999], &var[194], &args[
&var[999], &args[285], pad, &var[999], &var[97], &args[289], pad, &var[97], &var[34], 255, 0, &var[999], &args[296], pad, &var[999], 255, 0, &var[999], &args[302],
&args[306], pad, &var[996], &var[66], 255, 0, &var[999], &args[313], pad, &var[999], 255, 0, &var[999], &args[319], pad, &var[999], &var[2], &args[323], pad, &var[9
pad, &var[999], &var[998], &args[333], pad, &var[66], 255, 0, &var[999], &args[339], pad, &var[999], &var[997], &args[343], pad, &var[998], &var[997], &var[999], &a
&args[352], pad, &var[999], 255, 0, &var[999], &args[358], pad, &var[999], &var[193], &args[362], pad, &var[97], 255, 0, &var[999], &args[368], pad, &var[999], &var
255, 0, &var[999], &args[378], pad, &var[999], &var[997], &args[382], pad, &var[998], &var[997], &var[999], &args[387], pad, &var[999], &var[999], &args[391], pad,
&args[397], pad, &var[999], &var[194], &args[401], pad, &var[193], &var[194], &var[999], &args[406], pad, &var[999], &var[98], &args[410], pad, &var[98], &var[35],
&var[999], 255, 0, &var[999], &args[423], pad, &var[999], &var[996], &args[427], pad, &var[996], &var[67], 255, 0, &var[999], &args[434], pad, &var[999], 255, 0, &v
&var[3], &args[444], pad, &var[996], 255, 0, &var[999], &args[450], pad, &var[999], &var[998], &args[454], pad, &var[67], 255, 0, &var[999], &args[460], pad, &var[9
&var[998], &var[997], &var[999], &args[469], pad, &var[999], &var[999], &args[473], pad, &var[999], 255, 0, &var[999], &args[479], pad, &var[999], &var[193], &args[
&var[999], &args[489], pad, &var[999], &var[998], &args[493], pad, &var[35], 255, 0, &var[999], &args[499], pad, &var[999], &var[997], &args[503], pad, &var[998], &
pad, &var[999], &var[999], &args[512], pad, &var[999], 255, 0, &var[999], &args[518], pad, &var[999], &var[194], &args[522], pad, &var[193], &var[194], &var[999], &
&args[531], pad, &var[99], &var[36], 255, 0, &var[999], &args[538], pad, &var[999], 255, 0, &var[999], &args[544], pad, &var[999], &var[996], &args[548], pad, &var[
&args[555], pad, &var[999], 255, 0, &var[999], &args[561], pad, &var[999], &var[4], &args[565], pad, &var[996], 255, 0, &var[999], &args[571], pad, &var[999], &var[
255, 0, &var[999], &args[581], pad, &var[999], &var[997], &args[585], pad, &var[998], &var[997], &var[999], &args[590], pad, &var[999], &var[999], &args[594], pad,
&args[600], pad, &var[999], &var[193], &args[604], pad, &var[99], 255, 0, &var[999], &args[610], pad, &var[999], &var[998], &args[614], pad, &var[36], 255, 0, &var[
&var[997], &args[624], pad, &var[998], &var[997], &var[999], &args[629], pad, &var[999], &var[999], &args[633], pad, &var[999], 255, 0, &var[999], &args[639], pad,
pad, &var[193], &var[194], &var[999], &args[648], pad, &var[999], &var[100], &args[652], pad, &var[100], &var[37], 255, 0, &var[999], &args[659], pad, &var[999], 25
&var[999], &var[996], &args[669], pad, &var[996], &var[69], 255, 0, &var[999], &args[676], pad, &var[999], 255, 0, &var[999], &args[682], pad, &var[999], &var[5], &
&var[999], &args[692], pad, &var[999], &var[998], &args[696], pad, &var[69], 255, 0, &var[999], &args[702], pad, &var[999], &var[997], &args[706], pad, &var[998], &
pad, &var[999], &var[999], &args[715], pad, &var[999], 255, 0, &var[999], &args[721], pad, &var[999], &var[193], &args[725], pad, &var[100], 255, 0, &var[999], &arg
&args[735], pad, &var[37], 255, 0, &var[999], &args[741], pad, &var[999], &var[997], &args[745], pad, &var[998], &var[997], &var[999], &args[750], pad, &var[999], &
&var[999], 255, 0, &var[999], &args[760], pad, &var[999], &var[194], &args[764], pad, &var[193], &var[194], &var[999], &args[769], pad, &var[999], &var[101], &args[
0, &var[999], &args[780], pad, &var[999], 255, 0, &var[999], &args[786], pad, &var[999], &var[996], &args[790], pad, &var[996], &var[70], 255, 0, &var[999], &args[7
&var[999], &args[803], pad, &var[999], &var[6], &args[807], pad, &var[996], 255, 0, &var[999], &args[813], pad, &var[999], &var[998], &args[817], pad, &var[70], 255
&var[999], &var[997], &args[827], pad, &var[998], &var[997], &var[999], &args[832], pad, &var[999], &var[999], &args[836], pad, &var[999], 255, 0, &var[999], &args[
&args[846], pad, &var[101], 255, 0, &var[999], &args[852], pad, &var[999], &var[998], &args[856], pad, &var[38], 255, 0, &var[999], &args[862], pad, &var[999], &var
&var[997], &var[999], &args[871], pad, &var[999], &var[999], &args[875], pad, &var[999], 255, 0, &var[999], &args[881], pad, &var[999], &var[194], &args[885], pad,
&args[890], pad, &var[999], &var[102], &args[894], pad, &var[102], &var[39], 255, 0, &var[999], &args[901], pad, &var[999], 255, 0, &var[999], &args[907], pad, &var
&var[996], &var[71], 255, 0, &var[999], &args[918], pad, &var[999], 255, 0, &var[999], &args[924], pad, &var[999], &var[7], &args[928], pad, &var[996], 255, 0, &var
&var[998], &args[938], pad, &var[71], 255, 0, &var[999], &args[944], pad, &var[999], &var[997], &args[948], pad, &var[998], &var[997], &var[999], &args[953], pad, &
pad, &var[999], 255, 0, &var[999], &args[963], pad, &var[999], &var[193], &args[967], pad, &var[102], 255, 0, &var[999], &args[973], pad, &var[999], &var[998], &arg
&var[999], &args[983], pad, &var[999], &var[997], &args[987], pad, &var[998], &var[997], &var[999], &args[992], pad, &var[999], &var[999], &args[996], pad, &var[999
pad, &var[999], &var[194], &args[1006], pad, &var[193], &var[194], &var[999], &args[1011], pad, &var[999], &var[103]);
```

Conclusion

CFI with shadow stack stops some attacks.

Conclusion

CFI, in it's best form, can not stop all attacks.

Conclusion

Shadow stacks significantly complicates attacks.

Conclusion

Enforcing memory integrity may be more effective than control-flow integrity.

Questions?

… but wait, you didn't break CFI!

- The goal of CFI is to make programs secure.
- The mechanism CFI uses is enforcing a CFG.
- We do not break the mechanism.
- We do break the goal.