Critical Code Guided Directed Greybox Fuzzing for Commits

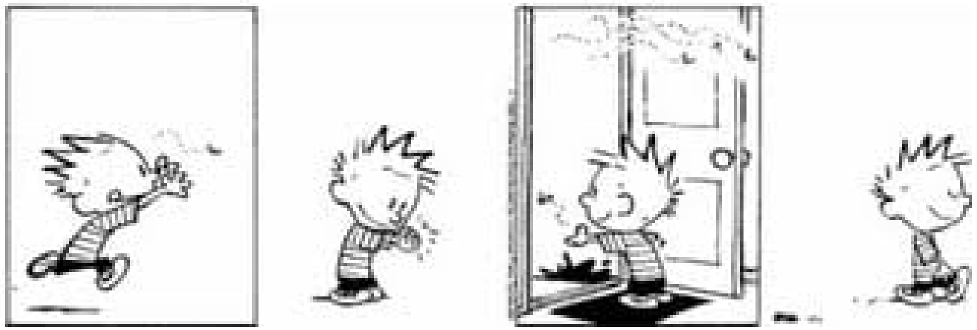
Yi Xiang, Xuhong Zhang, Peiyu Liu, Shouling Ji, Xiao Xiao, Hong Liang,

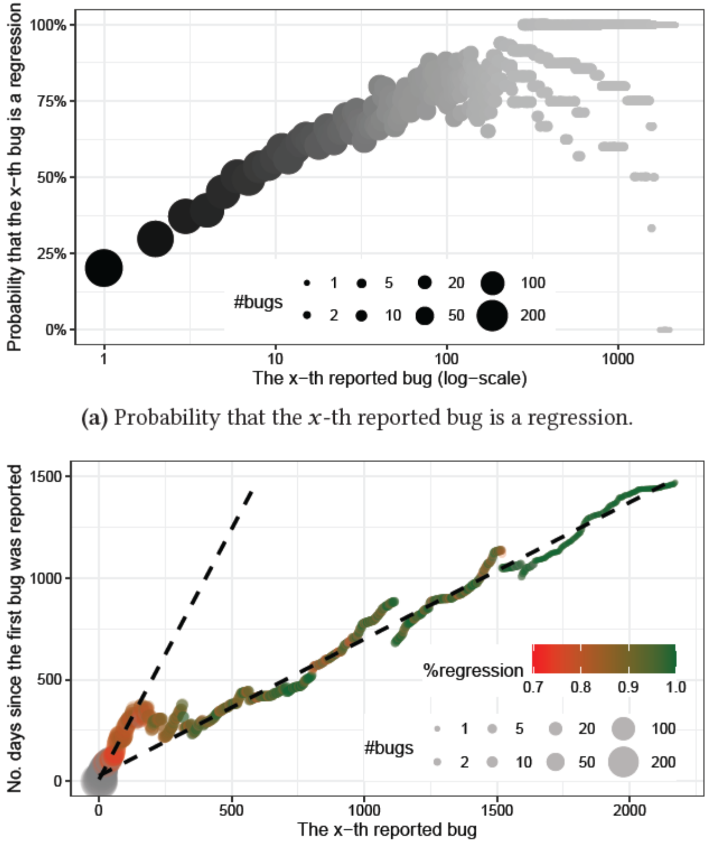
Jiacheng Xu, Wenhai Wang

Hangzhou, China

Commit Testing is Important

- Nearly 4/5 bug reports in OSSFuzz are regression bugs [1]
- Regression is initiated when a programmer fixes any bug or adds a new code for new functionality to the system [2]



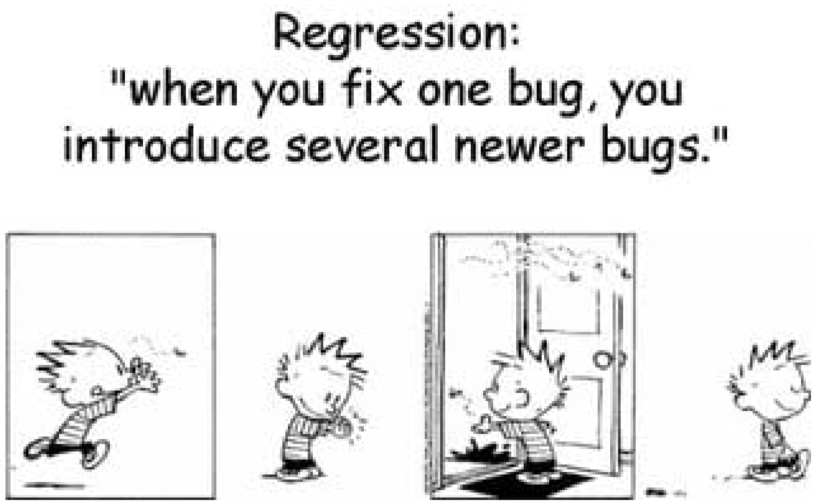


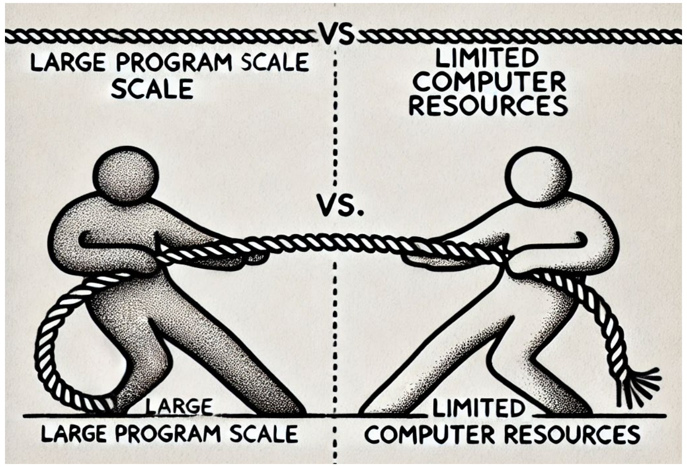
(a) Probability that the x-th reported bug is a regression.

(b) Number of days between the first and x-th bug report.

[1] Zhu, Xiaogang, and Marcel Böhme. "Regression greybox fuzzing." Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. 2021
[2] https://www.softwaretestinghelp.com/regression-testing-tools-and-methods/.

Commit Testing is Important





- Higher likelihood of newly added code introducing vulnerabilities
- Growing program scale but Limited availability of resources

➢ It is crucial to *prioritize* fuzzing commit modified code

Characteristics of Commit

- Manually identified the bug-inducing commit (BIC) of 30 real-world bugs, we observe that

Project	Program	Diff Loc 21	Same Loc 9	BIC	#Changed lines	Total BBNum
Libtiff	tiffcrop	#488		7057734d	40+,17-	13921
	tiffcrop	#498		07d79fcac	51+,26-	15256
	tiffcrop	#519		f13cf46b	9+,2-	15227
	tiffcrop		#520	e3195080	210+,72-	15706
	tiffcrop	#527		07d79fcac	51+,26-	15256
	tiffcrop	#530		f13cf46b	9+,2-	15227
	tiffcp		#548	3079627e	244+,137-	13134
	tiffinfo		#559	b90b20d3	1647+,1538-	13722
Bento4	mp4info	#652		c9f2c53	33+,18-	15621
	mp4info		#679	2e29350	1148+,742-	17216
	mp4audioclip	#732		bbb6f24	1045+,1688-	14593
	mp42aac		#751	61b2012	0+,6-	14424
Mujs	mujs	#65		8c27b126	27+,16-	6482
	mujs	#141		832e0690	87+,27-	6996
	mujs		#145	4c7f6be	41+,5-	7319
	mujs	#166		3f71a1c9	260+,47-	15791
Libjpeg	cjpeg	#493		88ae609	1999+,228-	4982
	jpegtran	#636		88ae609	1999+,228-	6075
Tcpreplay	tcprewrite		#702	0a65668a	282+,148-	4110
	tcprewrite	#718		2c76868d	45+,45-	4030
	tcpprep	#756		16442ac3	312+,338-	1855
	tcpreplay	#772		4f9158da	1+,2-	2240
Libxml2	xmllint	#535		9a82b94a	253+,176-	66472
	xmllint	#550		7e3f469b	32+,38-	66150
Poppler	pdfunite	#1282		3d35d209	16+,0-	44103
	pdfunite		#1289	3cae7773	31+,2-	1015
	pdftops	#1303		e674ca64	71+,80-	42235
	pdftoppm	#1305		aaf2e808	31+,2-	37682
	pdftoppm		#1381	245abada	20+,45-	51098
ImageMagick	magick	#6075		a107b941	103+,134-	134594

Characteristics of Commit

- Manually identified the bug-inducing commit (BIC) of 30 real-world bugs, we observe that
  - the crash site often **differ from** the commit change site

Project	Program	Diff Loc 21	Same Loc 9	BIC	#Changed lines	Total BBNum
Libtiff	tiffcrop	#488		7057734d	40+,17-	13921
	tiffcrop	#498		07d79fcac	51+,26-	15256
	tiffcrop	#519		f13cf46b	9+,2-	15227
	tiffcrop		#520	e3195080	210+,72-	15706
	tiffcrop	#527		07d79fcac	51+,26-	15256
	tiffcrop	#530		f13cf46b	9+,2-	15227
	tiffcp		#548	3079627e	244+,137-	13134
	tiffinfo		#559	b90b20d3	1647+,1538-	13722
Bento4	mp4info	#652		c9f2c53	33+,18-	15621
	mp4info		#679	2e29350	1148+,742-	17216
	mp4audioclip	#732		bbb6f24	1045+,1688-	14593
	mp42aac		#751	61b2012	0+,6-	14424
Mujs	mujs	#65		8c27b126	27+,16-	6482
	mujs	#141		832e0690	87+,27-	6996
	mujs		#145	4c7f6be	41+,5-	7319
	mujs	#166		3f71a1c9	260+,47-	15791
Libjpeg	cjpeg	#493		88ae609	1999+,228-	4982
	jpegtran	#636		88ae609	1999+,228-	6075
Tcpreplay	tcprewrite		#702	0a65668a	282+,148-	4110
	tcprewrite	#718		2c76868d	45+,45-	4030
	tcpprep	#756		16442ac3	312+,338-	1855
	tcpreplay	#772		4f9158da	1+,2-	2240
Libxml2	xmllint	#535		9a82b94a	253+,176-	66472
	xmllint	#550		7e3f469b	32+,38-	66150
Poppler	pdfunite	#1282		3d35d209	16+,0-	44103
	pdfunite		#1289	3cae7773	31+,2-	1015
	pdftops	#1303		e674ca64	71+,80-	42235
	pdftoppm	#1305		aaf2e808	31+,2-	37682
	pdftoppm		#1381	245abada	20+,45-	51098
ImageMagick	magick	#6075		a107b941	103+,134-	134594

Characteristics of Commit

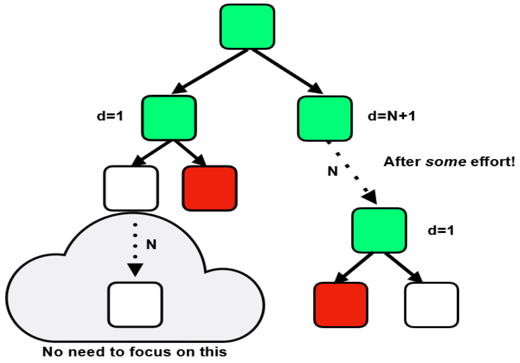
- Manually identified the bug-inducing commit (BIC) of 30 real-world bugs, we observe that
  - the crash site often **differ from** the commit change site
  - the BIC often contains **multiple** change sites

Project	Program	Diff Loc 21	Same Loc 9	BIC	#Changed lines	Total BBNum
Libtiff	tiffcrop	#488		7057734d	40+,17-	13921
	tiffcrop	#498		07d79fcac	51+,26-	15256
	tiffcrop	#519		f13cf46b	9+,2-	15227
	tiffcrop		#520	e3195080	210+,72-	15706
	tiffcrop	#527		07d79fcac	51+,26-	15256
	tiffcrop	#530		f13cf46b	9+,2-	15227
	tiffcp		#548	3079627e	244+,137-	13134
	tiffinfo		#559	b90b20d3	1647+,1538-	13722
Bento4	mp4info	#652		c9f2c53	33+,18-	15621
	mp4info		#679	2e29350	1148+,742-	17216
	mp4audioclip	#732		bbb6f24	1045+,1688-	14593
	mp42aac		#751	61b2012	0+,6-	14424
Mujs	mujs	#65		8c27b126	27+,16-	6482
	mujs	#141		832e0690	87+,27-	6996
	mujs		#145	4c7f6be	41+,5-	7319
	mujs	#166		3f71a1c9	260+,47-	15791
Libjpeg	cjpeg	#493		88ae609	1999+,228-	4982
	jpegtran	#636		88ae609	1999+,228-	6075
Tcpreplay	tcprewrite		#702	0a65668a	282+,148-	4110
	tcprewrite	#718		2c76868d	45+,45-	4030
	tcpprep	#756		16442ac3	312+,338-	1855
	tcpreplay	#772		4f9158da	1+,2-	2240
Libxml2	xmllint	#535		9a82b94a	253+,176-	66472
	xmllint	#550		7e3f469b	32+,38-	66150
Poppler	pdfunite	#1282		3d35d209	16+,0-	44103
	pdfunite		#1289	3cae7773	31+,2-	1015
	pdftops	#1303		e674ca64	71+,80-	42235
	pdftoppm	#1305		aaf2e808	31+,2-	37682
	pdftoppm		#1381	245abada	20+,45-	51098
ImageMagick	magick	#6075		a107b941	103+,134-	134594

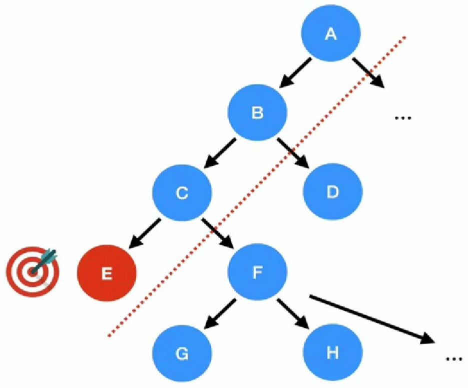
Characteristics of Commit

- Manually identified the bug-inducing commit (BIC) of 30 real-world bugs, we observe that
  - the crash site often **differ from** the commit change site
  - the BIC often contains **multiple** change sites

⟹ existing directed greybox fuzzers encounter several problems



Distance-based DGF: ALFGo (**CCS'17**)

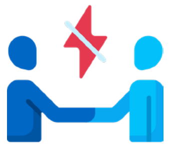


Reachability-based DGF: Beacon (**S&P'22**)

Project	Program	Diff Loc 21	Same Loc 9	BIC	#Changed lines	Total BBNum
Libtiff	tiffcrop	#488		7057734d	40+,17-	13921
	tiffcrop	#498		07d79fcac	51+,26-	15256
	tiffcrop	#519		f13cf46b	9+,2-	15227
	tiffcrop		#520	e3195080	210+,72-	15706
	tiffcrop	#527		07d79fcac	51+,26-	15256
	tiffcrop	#530		f13cf46b	9+,2-	15227
	tiffcp		#548	3079627e	244+,137-	13134
	tiffinfo		#559	b90b20d3	1647+,1538-	13722
Bento4	mp4info	#652		c9f2c53	33+,18-	15621
	mp4info		#679	2e29350	1148+,742-	17216
	mp4audioclip	#732		bbb6f24	1045+,1688-	14593
	mp42aac		#751	61b2012	0+,6-	14424
Mujs	mujs	#65		8c27b126	27+,16-	6482
	mujs	#141		832e0690	87+,27-	6996
	mujs		#145	4c7f6be	41+,5-	7319
	mujs	#166		3f71a1c9	260+,47-	15791
Libjpeg	cjpeg	#493		88ae609	1999+,228-	4982
	jpegtran	#636		88ae609	1999+,228-	6075
Tcpreplay	tcprewrite		#702	0a65668a	282+,148-	4110
	tcprewrite	#718		2c76868d	45+,45-	4030
	tcpprep	#756		16442ac3	312+,338-	1855
	tcpreplay	#772		4f9158da	1+,2-	2240
Libxml2	xmllint	#535		9a82b94a	253+,176-	66472
	xmllint	#550		7e3f469b	32+,38-	66150
Poppler	pdfunite	#1282		3d35d209	16+,0-	44103
	pdfunite		#1289	3cae7773	31+,2-	1015
	pdftops	#1303		e674ca64	71+,80-	42235
	pdftoppm	#1305		aaf2e808	31+,2-	37682
	pdftoppm		#1381	245abada	20+,45-	51098
ImageMagick	magick	#6075		a107b941	103+,134-	134594

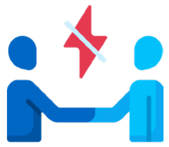
[3] Böhme, Marcel, et al. "Directed greybox fuzzing." Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. 2017.
[4] Huang, Heqing, et al. "Beacon: Directed grey-box fuzzing with provable path pruning." 2022 IEEE Symposium on Security and Privacy (SP). IEEE, 2022.

Deficiency of Existing DGFs



- The crash site often **differ from** the commit change site
- Focusing on reaching the target (change site) quickly, but neglecting **thorough testing** of affected code

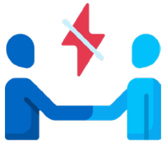
Deficiency of Existing DGFs



- The crash site often **differ from** the commit change site
- Focusing on reaching the target (change site) quickly, but neglecting **thorough testing** of affected code

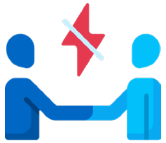
$\Longrightarrow$ ● **Failure** to detect newly introduced vulnerabilities

Deficiency of Existing DGFs



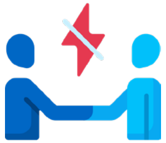
- The crash site often **differ from** the commit change site
- Focusing on reaching the target (change site) quickly, but neglecting **thorough testing** of affected code

$\Longrightarrow$ • **Failure** to detect newly introduced vulnerabilities



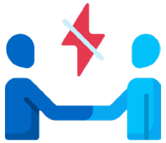
- The BIC often contains **multiple** change sites
- **Struggle** to effectively address the multi-targets issue

Deficiency of Existing DGFs



- The crash site often **differ from** the commit change site
- Focusing on reaching the target (change site) quickly, but neglecting **thorough testing** of affected code

⟹
- **Failure** to detect newly introduced vulnerabilities



- The BIC often contains **multiple** change sites
- **Struggle** to effectively address the multi-targets issue

⟹
- Degrading to coverage-based fuzzing, **lacking guidance**
- Disregarding connections between change sites, **less efficient**

Challenges

- How to quickly and thoroughly test the affected code?

- How to handle multiple site changes in a smart and lightweight manner?

Challenges

- How to quickly and thoroughly test the affected code?
  - first efficiently **reach** the change site (target)
  - **maintain** the reachability, and then generate **diverse inputs** to explore different program states of the affected code
- How to handle multiple site changes in a smart and lightweight manner?
  - guarantee the directness of **each grouped target**

Methodology

A critical code guided directed fuzzer for commit.

- ✓ Group targets and calculate distance
- ✓ Identify critical code and guide input generation strategy

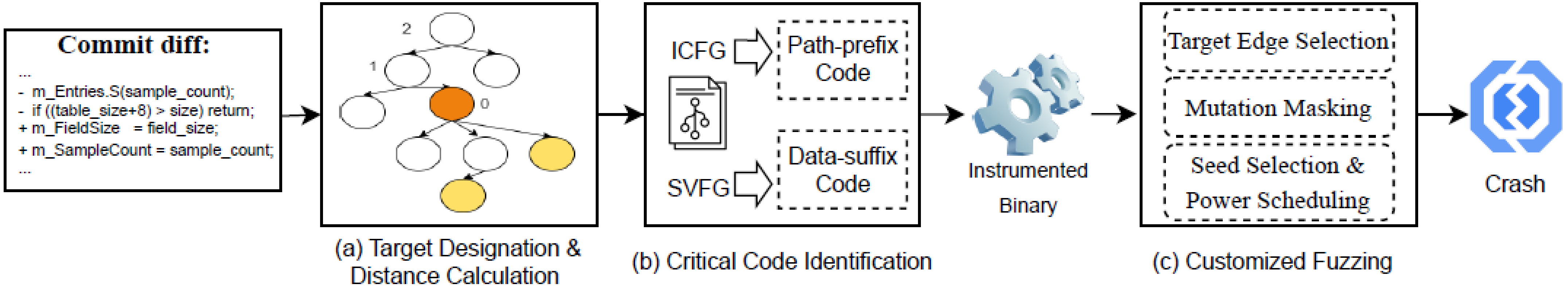


Figure 2: Architecture of WAFLGo.

Critical Code Guided Fuzzing

A **critical code** guided directed fuzzer for commit.

- Identify Critical Code
  - Path-prefix code: a, b, e, and f
  - Data-suffix code: i and k

```
1  int main(){
2    int x,y,z,w=input1();
3    char ar[10]=input2();
4    if(ar == "TEST"){
5      if(y>0){
6        if(z<0){
7          ...
8        }else{
9          if(z<10){
10           y=3;
11           if(w<5)
12             goto ...
13         }
14         x=y+5 -> x=y-5;
15         if(y>20){
16           y=0;
17         }else{
18           ar[y]="A";
19           if(x>10){
20             ar[0]="B";
21           }else{
22             ar[x]="C";
23   } } } } }
24   return 0;
25 }
```

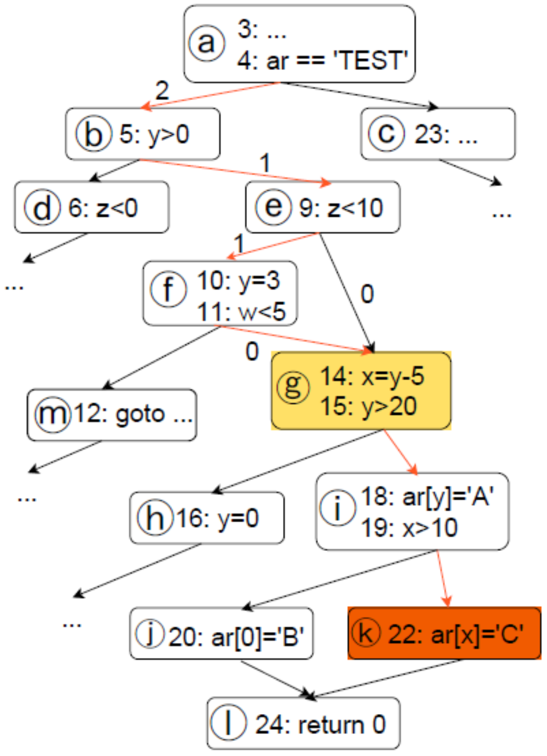


Figure 3: Illustration example. Line 14 is the change site (target) where we change $x = y + 5$ to $x = y - 5$. Line 22 is the crash site.

Critical Code Guided Fuzzing

A critical code guided directed fuzzer for commit.

- Identify Critical Code
    - Path-prefix code: a, b, e, and f
    - Data-suffix code: i and k

```
1  int main(){
2    int x,y,z,w=input1();
3    char ar[10]=input2();
4    if(ar == "TEST"){
5      if(y>0){
6        if(z<0){
7          ...
8        }else{
9          if(z<10){
10           y=3;
11           if(w<5)
12             goto ...
13         }
14         x=y+5 -> x=y-5;
15         if(y>20){
16           y=0;
17         }else{
18           ar[y]="A";
19           if(x>10){
20             ar[0]="B";
21           }else{
22             ar[x]="C";
23   } } } } }
24   return 0;
25 }
```

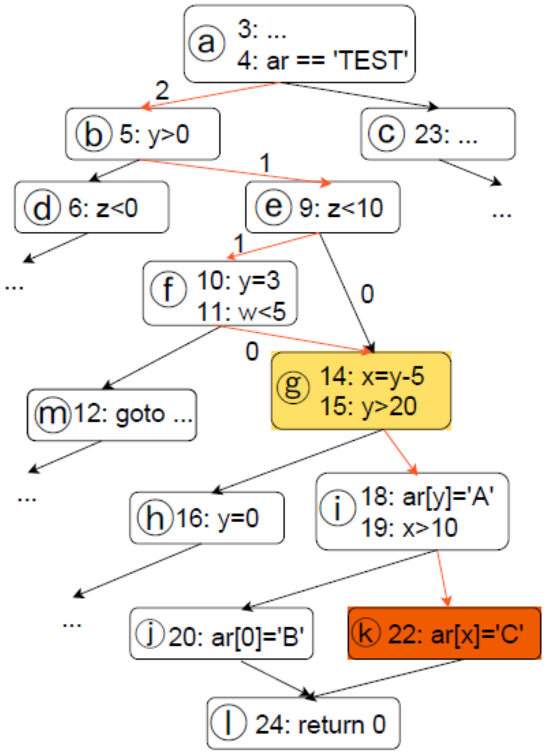


Figure 3: Illustration example. Line 14 is the change site (target) where we change $x = y + 5$ to $x = y - 5$. Line 22 is the crash site.

Critical Code Guided Fuzzing

A critical code guided directed fuzzer for commit.

- Identify Critical Code
  - Path-prefix code: a, b, e, and f
  - Data-suffix code: i and k (only consider written variable x)

```
1  int main(){
2    int x,y,z,w=input1();
3    char ar[10]=input2();
4    if(ar == "TEST"){
5      if(y>0){
6        if(z<0){
7          ...
8        }else{
9          if(z<10){
10           y=3;
11           if(w<5)
12             goto ...
13         }
14         x=y+5 -> x=y-5;
15         if(y>20){
16           y=0;
17         }else{
18           ar[y]="A";
19           if(x>10){
20             ar[0]="B";
21           }else{
22             ar[x]="C";
23   } } } } }
24   return 0;
25 }
```

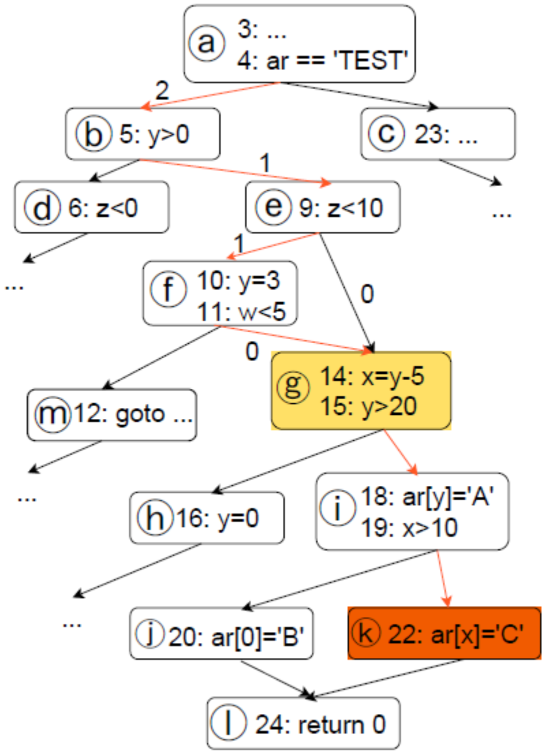


Figure 3: Illustration example. Line 14 is the change site (target) where we change $x = y + 5$ to $x = y - 5$. Line 22 is the crash site.

Critical Code Guided Fuzzing

A critical code guided directed fuzzer for commit.

- Identify Critical Code
  - Path-prefix code: a, b, e, and f
  - Data-suffix code: i and k (only consider written variable x)

Static Value-Flow Analysis Framework

```
1  int main(){
2    int x,y,z,w=input1();
3    char ar[10]=input2();
4    if(ar == "TEST"){
5      if(y>0){
6        if(z<0){
7          ...
8        }else{
9          if(z<10){
10           y=3;
11           if(w<5)
12             goto ...
13         }
14         x=y+5 -> x=y-5;
15         if(y>20){
16           y=0;
17         }else{
18           ar[y]="A";
19           if(x>10){
20             ar[0]="B";
21           }else{
22             ar[x]="C";
23   } } } } }
24   return 0;
25 }
```

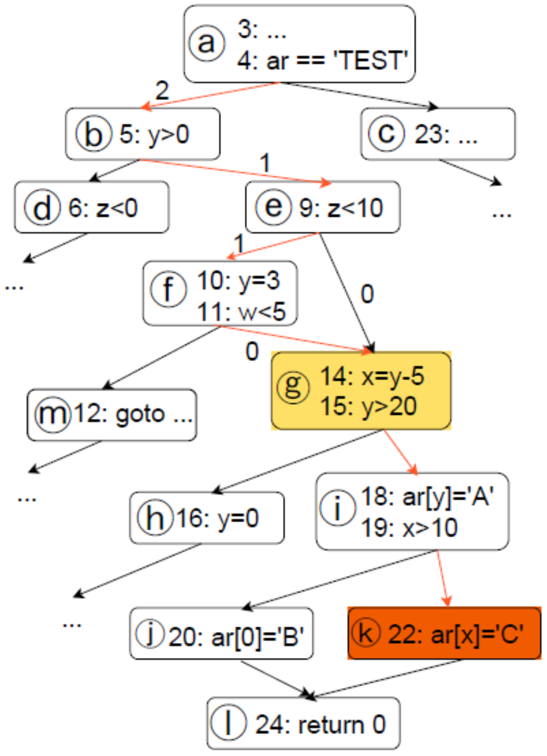


Figure 3: Illustration example. Line 14 is the change site (target) where we change $x = y+5$ to $x = y-5$. Line 22 is the crash site.

[5] https://github.com/SVF-tools/SVF

Critical Code Guided Fuzzing

A critical code guided directed fuzzer for commit.

- Identify Critical Code
  - Path-prefix code: a, b, e, and f
  - Data-suffix code: i and k (only consider written variable x)
- Input Generation Strategy
  - ✓ Key insight: preserving the execution of the critical code, while generating diverse testcases

```
1  int main(){
2    int x,y,z,w=input1();
3    char ar[10]=input2();
4    if(ar == "TEST"){
5      if(y>0){
6        if(z<0){
7          ...
8        }else{
9          if(z<10){
10           y=3;
11           if(w<5)
12             goto ...
13         }
14         x=y+5 -> x=y-5;
15         if(y>20){
16           y=0;
17         }else{
18           ar[y]="A";
19           if(x>10){
20             ar[0]="B";
21           }else{
22             ar[x]="C";
23   } } } } }
24   return 0;
25 }
```

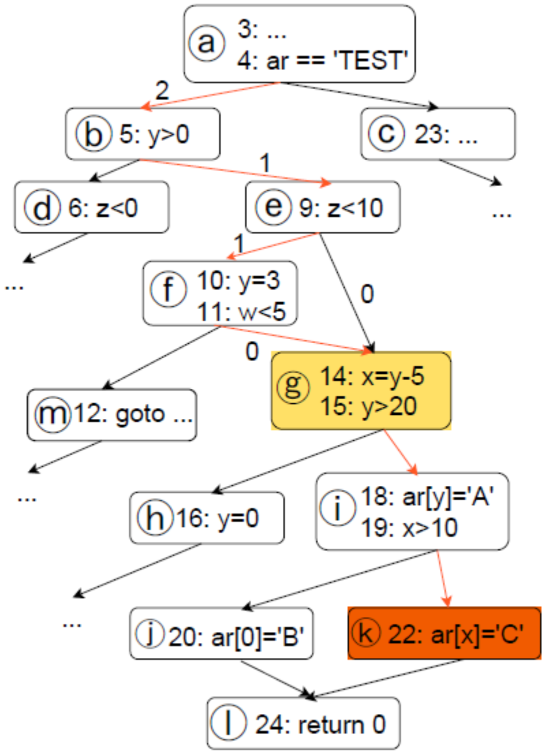


Figure 3: Illustration example. Line 14 is the change site (target) where we change $x = y+5$ to $x = y-5$. Line 22 is the crash site.

Critical Code Guided Fuzzing

A critical code guided directed fuzzer for commit.

- Identify Critical Code
  - Path-prefix code: a, b, e, and f
  - Data-suffix code: i and k (only consider written variable x)
- Input Generation Strategy
  - ✓ Key insight: preserving the execution of the critical code, while generating diverse testcases
  - Select target edge based on execution status

  seed A: a→b→e→f→m, target edge: e_{ef}, e_{be}, e_{ab}

```
1  int main(){
2    int x,y,z,w=input1();
3    char ar[10]=input2();
4    if(ar == "TEST"){
5      if(y>0){
6        if(z<0){
7          ...
8        }else{
9          if(z<10){
10           y=3;
11           if(w<5)
12             goto ...
13         }
14         x=y+5 -> x=y-5;
15         if(y>20){
16           y=0;
17         }else{
18           ar[y]="A";
19           if(x>10){
20             ar[0]="B";
21           }else{
22             ar[x]="C";
23   } } } } }
24   return 0;
25 }
```

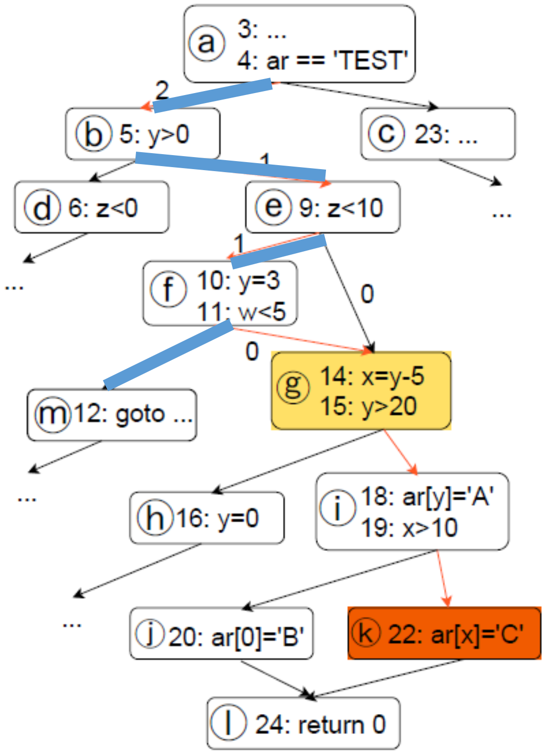


Figure 3: Illustration example. Line 14 is the change site (target) where we change $x = y + 5$ to $x = y - 5$. Line 22 is the crash site.

Critical Code Guided Fuzzing

A critical code guided directed fuzzer for commit.

- Identify Critical Code
  - Path-prefix code: a, b, e, and f
  - Data-suffix code: i and k (only consider written variable x)

- Input Generation Strategy
  - ✓ Key insight: preserving the execution of the critical code, while generating diverse testcases
  - Select target edge based on execution status

  seed A: a→b→e→f→m, target edge: e_{ef}, e_{be}, e_{ab}

```
1  int main(){
2    int x,y,z,w=input1();
3    char ar[10]=input2();
4    if(ar == "TEST"){
5      if(y>0){
6        if(z<0){
7          ...
8        }else{
9          if(z<10){
10           y=3;
11           if(w<5)
12             goto ...
13         }
14         x=y+5 -> x=y-5;
15         if(y>20){
16           y=0;
17         }else{
18           ar[y]="A";
19           if(x>10){
20             ar[0]="B";
21           }else{
22             ar[x]="C";
23   } } } } }
24   return 0;
25 }
```

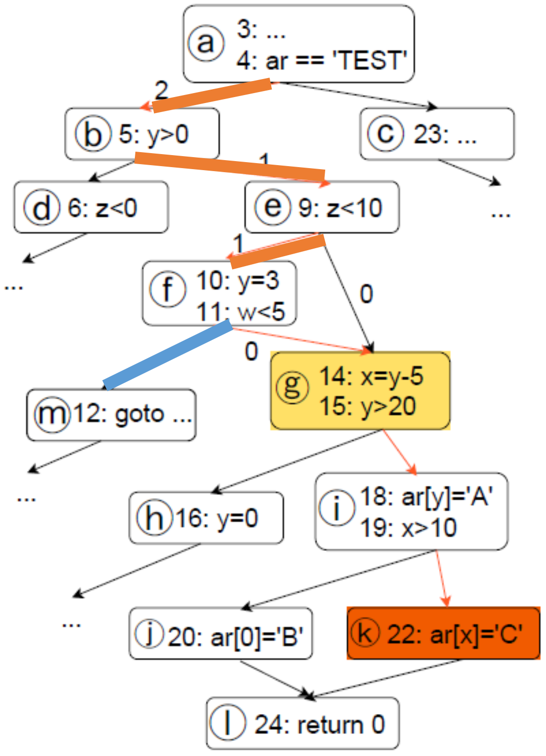


Figure 3: Illustration example. Line 14 is the change site (target) where we change $x = y+5$ to $x = y-5$. Line 22 is the crash site.

Critical Code Guided Fuzzing

A critical code guided directed fuzzer for commit.

- Identify Critical Code
  - Path-prefix code: a, b, e, and f
  - Data-suffix code: i and k (only consider written variable x)
- Input Generation Strategy
  - ✓ Key insight: preserving the execution of the critical code, while generating diverse testcases
  - Select target edge based on execution status

seed A: a→b→e→f→m, target edge: e_{ef}, e_{be}, e_{ab}

seed B: a→b→e→g→i→j→l, target edge: e_{eg}, e_{be}, e_{ab} , e_{gi}

```
1  int main(){
2    int x,y,z,w=input1();
3    char ar[10]=input2();
4    if(ar == "TEST"){
5      if(y>0){
6        if(z<0){
7          ...
8        }else{
9          if(z<10){
10           y=3;
11           if(w<5)
12             goto ...
13         }
14         x=y+5 -> x=y-5;
15         if(y>20){
16           y=0;
17         }else{
18           ar[y]="A";
19           if(x>10){
20             ar[0]="B";
21           }else{
22             ar[x]="C";
23   } } } } }
24   return 0;
25 }
```

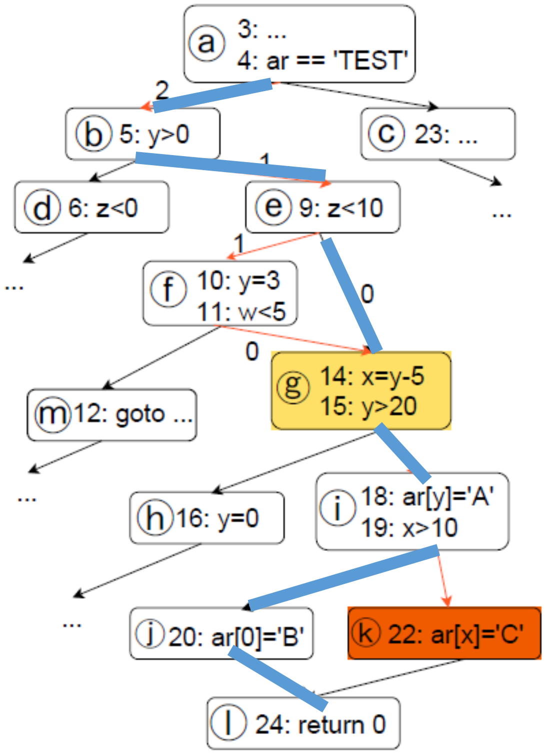


Figure 3: Illustration example. Line 14 is the change site (target) where we change $x = y + 5$ to $x = y - 5$. Line 22 is the crash site.

Critical Code Guided Fuzzing

A critical code guided directed fuzzer for commit.

- Identify Critical Code
  - Path-prefix code: a, b, e, and f
  - Data-suffix code: i and k (only consider written variable x)

- Input Generation Strategy
  - ✓ Key insight: preserving the execution of the critical code, while generating diverse testcases
  - Select target edge based on execution status

seed A: a→b→e→f→m, target edge: e_{ef}, e_{be}, e_{ab}

seed B: a→b→e→g→i→j→l, target edge: e_{eg}, e_{be}, e_{ab} , e_{gi}

```
1  int main(){
2    int x,y,z,w=input1();
3    char ar[10]=input2();
4    if(ar == "TEST"){
5      if(y>0){
6        if(z<0){
7          ...
8        }else{
9          if(z<10){
10           y=3;
11           if(w<5)
12             goto ...
13         }
14         x=y+5 -> x=y-5;
15         if(y>20){
16           y=0;
17         }else{
18           ar[y]="A";
19           if(x>10){
20             ar[0]="B";
21           }else{
22             ar[x]="C";
23   } } } } }
24   return 0;
25 }
```

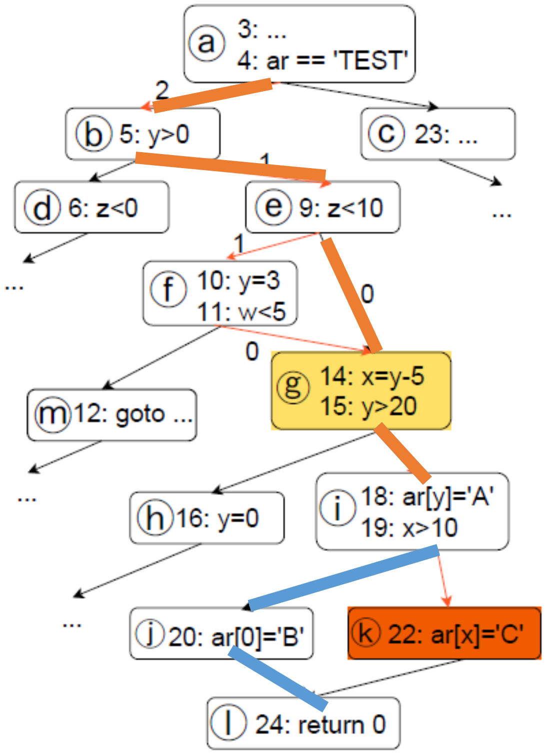


Figure 3: Illustration example. Line 14 is the change site (target) where we change $x = y + 5$ to $x = y - 5$. Line 22 is the crash site.

Critical Code Guided Fuzzing

A critical code guided directed fuzzer for commit.

- Identify Critical Code
  - Path-prefix code: a, b, e, and f
  - Data-suffix code: i and k (only consider written variable x)

- Input Generation Strategy
  - ✓ Key insight: preserving the execution of the critical code, while generating diverse testcases
  - Select target edge based on execution status

    seed A: a→b→e→f→m, target edge: e_{ef}, e_{be}, e_{ab}

    seed B: a→b→e→g→i→j→l, target edge: e_{eg}, e_{be}, e_{ab} , e_{gi}

  - Use mutation masks to sustain target edge execution

```
1  int main(){
2    int x,y,z,w=input1();
3    char ar[10]=input2();
4    if(ar == "TEST"){
5      if(y>0){
6        if(z<0){
7          ...
8        }else{
9          if(z<10){
10           y=3;
11           if(w<5)
12             goto ...
13         }
14         x=y+5 -> x=y-5;
15         if(y>20){
16           y=0;
17         }else{
18           ar[y]="A";
19           if(x>10){
20             ar[0]="B";
21           }else{
22             ar[x]="C";
23   } } } } }
24   return 0;
25 }
```

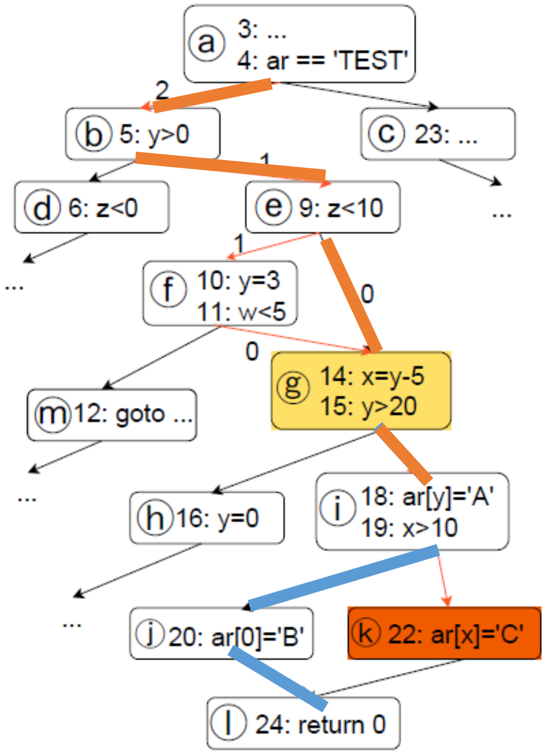


Figure 3: Illustration example. Line 14 is the change site (target) where we change $x = y + 5$ to $x = y - 5$. Line 22 is the crash site.

Commit Fuzzer

A critical code guided directed fuzzer for commit.

- Group targets based on the same preconditions (within the same function)

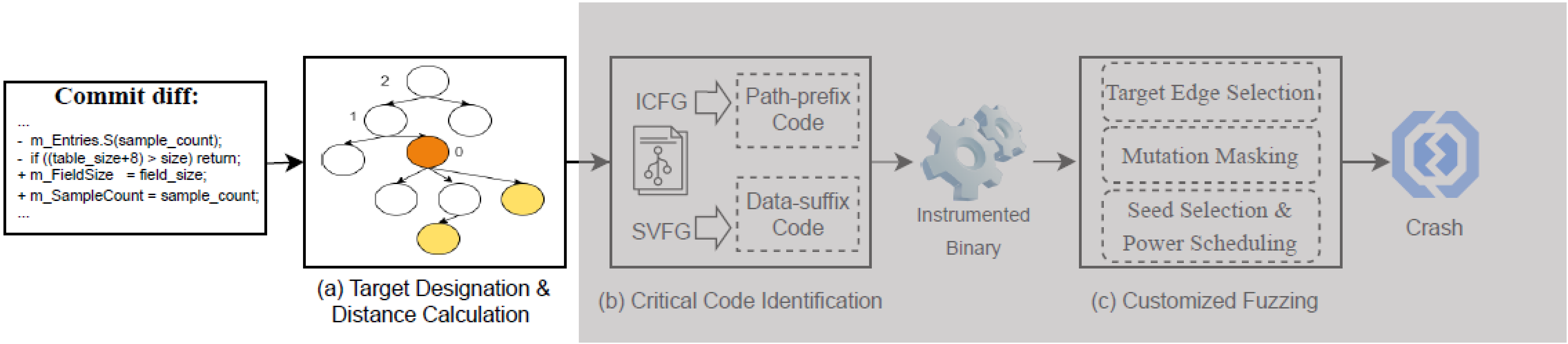


Figure 2: Architecture of WAFLGo.

Commit Fuzzer

A critical code guided directed fuzzer for commit.

- Group targets based on the same preconditions (within the same function)
- Calculate input distance for the **rarest** executed target (similar with AFLGo)

$$d_s(s, T_b) = \frac{\sum_{m \in \xi(s)} d_b(m, T_b)}{|\xi(s)|}$$

$$\xi(s) = \{m \mid m \in \delta(s) \text{ and } d_b(m, T_b) \neq \text{NaN}\}$$

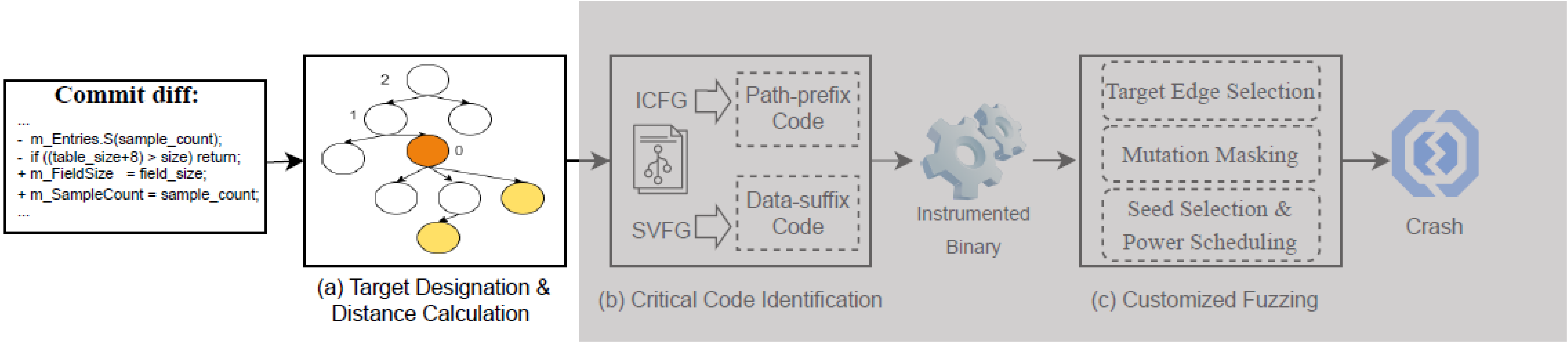


Figure 2: Architecture of WAFLGo.

Effectiveness of Bug Reproducing

How effective is WAFLGO in discovering bugs introduced by commits?

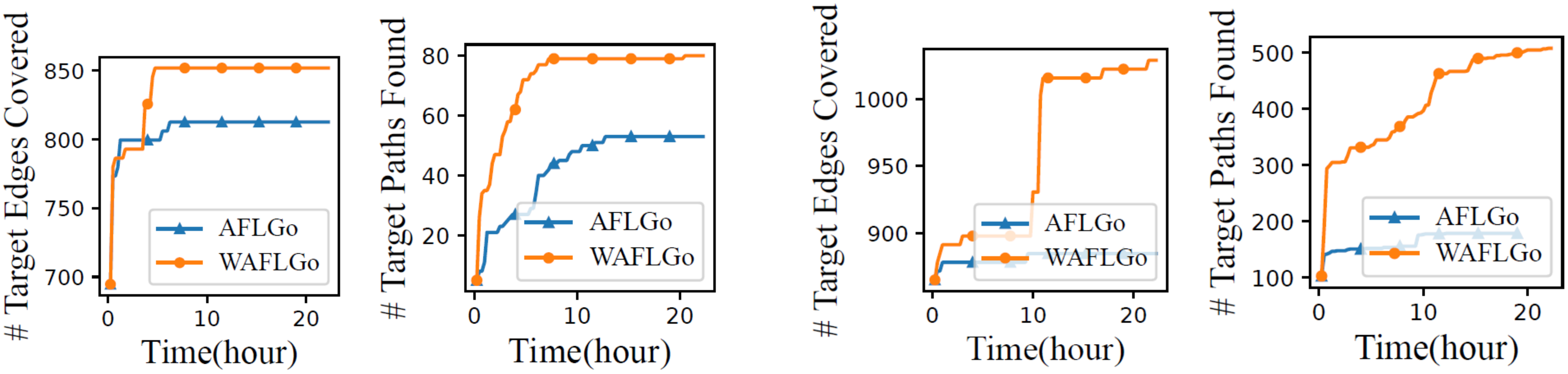
- ✓ WAFLGo effectively reproduces 21/30, achieving the highest success rate among all the fuzzers
- ✓ WAFLGO achieves an average speedup of 10.3× compared to others in reproducing bug time

No.	Issue-id	Program	Time-to-Exposure(hour)									Factor							
			WAFLGo	AFLGo	Wind.	Selc.	Fish.	AFL	AFL++	Fair.	AFLC.	AFLGo	Wind.	Selc.	Fish.	AFL	AFL++	Fair.	AFLC.
1	#488	tiffcrop	6.247	T.O.	T.O.	T.O.	T.O.	T.O.	T.O.	7.956	T.O.	3.8	3.8	3.8	3.8	3.8	3.8	1.3	3.8
2	#498	tiffcrop	0.001	0.011	0.004	0.003	0.012	0.005	0.005	0.001	0.003	9.5	3.9	2.6	10.4	4.6	4.4	1.0	2.5
3	#519	tiffcrop	0.286	6.059	3.002	2.081	3.955	6.426	0.613	0.617	10.958	21.2	10.5	7.3	13.8	22.4	2.1	2.2	38.3
4	#520	tiffcrop	0.940	3.230	1.301	1.230	T.O.	6.080	2.305	5.367	1.913	3.4	1.4	1.3	25.5	6.5	2.5	5.7	2.0
5	#527	tiffcrop	13.903	T.O.	T.O.	17.596	17.354	T.O.	16.071	19.717	T.O.	1.7	1.4	1.3	1.2	1.7	1.2	1.4	1.7
6	#530	tiffcrop	9.759	T.O.	19.842	T.O.	15.428	T.O.	T.O.	T.O.	13.340	2.5	2.5	2.5	1.6	2.5	2.5	2.5	1.4
7	#548	tiffcp	2.593	23.401	14.723	11.489	T.O.	22.143	3.610	9.008	T.O.	9.0	5.7	4.4	9.3	8.5	1.4	3.5	9.3
8	#559	tiffinfo	0.656	1.826	4.906	14.600	T.O.	2.726	2.617	1.084	5.509	2.8	7.5	22.3	36.6	4.2	4.0	1.7	8.4
9	#732	mp3aud.	0.010	0.134	0.069	0.050	0.064	0.055	0.076	0.062	0.055	13.0	6.7	4.9	6.3	5.4	7.4	6.0	5.4
10	#751	mp42aac	12.617	T.O.	T.O.	T.O.	T.O.	T.O.	14.768	T.O.	T.O.	1.9	1.9	1.9	1.9	1.9	1.2	1.9	1.9
11	#145	mujs	0.019	0.082	0.069	0.669	6.789	0.100	0.087	0.100	0.104	4.4	3.7	35.5	359.9	5.3	4.6	5.3	5.5
12	#493	cjpeg	0.028	0.509	0.633	0.158	0.523	0.831	0.850	3.900	0.368	17.9	22.2	5.6	18.4	29.2	29.9	137.0	12.9
13	#636	jpegtran	0.016	0.054	0.100	0.043	0.082	0.021	0.019	0.050	0.051	3.5	6.4	2.7	5.2	1.4	1.2	3.2	3.2
14	#702	tcprewrite	0.124	1.012	1.955	0.150	0.236	1.160	1.557	1.265	0.679	8.2	15.8	1.2	1.9	9.3	12.5	10.2	5.5
15	#718	tcprewrite	0.714	1.514	1.651	0.285	1.063	8.977	3.119	3.257	8.539	2.1	2.3	0.4	1.5	12.6	4.4	4.6	12.0
16	#756	tcpprep	0.401	6.671	0.535	1.746	0.867	6.881	T.O.	3.097	6.722	16.7	1.3	4.4	2.2	17.2	59.9	7.7	16.8
17	#772	tcpreplay	0.027	0.076	0.173	0.157	0.071	0.071	0.026	0.070	0.070	2.8	6.4	5.8	2.6	2.6	0.9	2.6	2.6
18	#535	xmllint	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
19	#1289	pdfunite	0.382	T.O.	1.891	1.716	13.155	T.O.	0.651	12.811	11.441	62.8	5.0	4.5	34.4	62.8	1.7	33.5	30.0
20	#1305	pdftoppm	6.672	11.891	17.470	T.O.	16.537	12.901	14.154	12.382	11.218	1.8	2.6	3.6	2.5	1.9	2.1	1.9	1.7
21	#6075	magick	10.989	T.O.	T.O.	T.O.	T.O.	T.O.	T.O.	T.O.	T.O.	2.2	2.2	2.2	2.2	2.2	2.2	2.2	2.2
#Reproduced / Average			21	15	17	16	15	15	17	18	16	9.1	5.4	5.7	25.8	9.9	7.2	11.2	8.0

Coverage Improvement

Does the guidance toward critical code improve the efficiency of fuzzing?

✓ WAFLGO demonstrates an average 11.7% increase in edge coverage and nearly 2× (181.5%) more path discoveries compared to AFLGo after 24 hours.



(a) Tcpreplay #718 (b) Poppler #1289

Multi-target Case

Does the multi-target optimizations improve the efficiency of fuzzing?

- ✓ Case study:

  For issue #1289, AFLGo overlooks target 0, while the seed distribution in FishFuzz[4] is similar to that of WAFLGO.

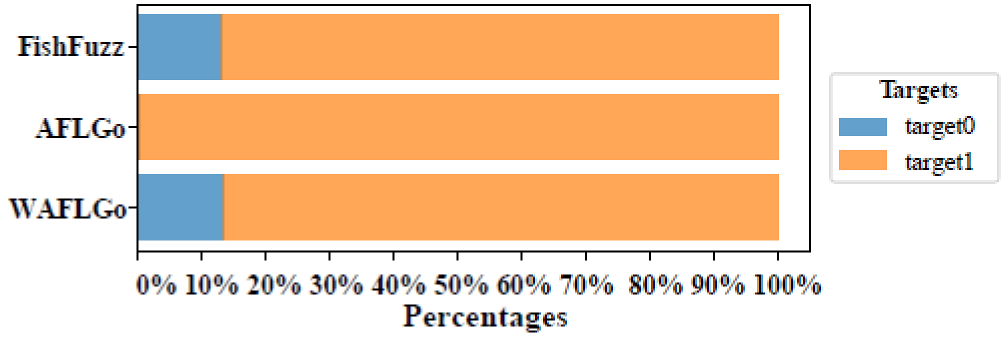


Figure 6: Target reached seeds.

[6] Zheng, Han, et al. "FISHFUZZ: Catch Deeper Bugs by Throwing Larger Nets." 32nd USENIX Security Symposium (USENIX Security 23). 2023.

Real-world Vulnerabilities

Can WAFLGO detect new vulnerabilities in real-world programs?

- ✓ WAFLGO discover seven new bugs, including four CVEs.
- ✓ Case study:
  The CVE-2023-34631 is introduced by the fixing commit (6678ad8) for the CVE-2023-34630.

Table 4: New vulnerabilities detected by WAFLGo

Program	Bug Type	Status	ID
tiffcrop	segmentation fault	patched	CVE-2023-3618
fig2dev	null pointer dereference	patched	CVE-2023-34629
fig2dev	segmentation fault	patched	CVE-2023-34630
fig2dev	memory leak	patched	CVE-2023-34631
swftophp	heap buffer overflow	reported	issue-271
swftophp	heap buffer overflow	reported	issue-270
swftophp	heap buffer overflow	reported	issue-269

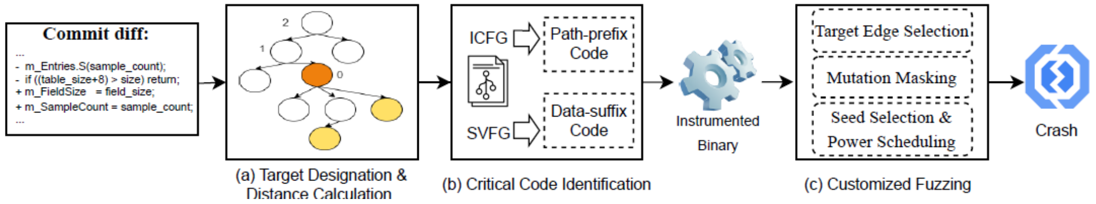
Critical Code Guided Directed Greybox Fuzzing for Commits

Real World Dataset

Project	Program	Diff Loc 21	Same Loc 9	BIC	#Changed lines	Total BBNum
Libtiff	tiffcrop	#488		7057734d	40+,17-	13921
	tiffcrop	#498		07d79fcac	51+,26-	15256
	tiffcrop	#519		f13cf46b	9+,2-	15227
	tiffcrop		#520	e3195080	210+,72-	15706
	tiffcrop	#527		07d79fcac	51+,26-	15256
	tiffcrop	#530		f13cf46b	9+,2-	15227
	tiffcp		#548	3079627e	244+,137-	13134
	tiffinfo		#559	b90b20d3	1647+,1538-	13722
Bento4	mp4info	#652		c9f2c53	33+,18-	15621
	mp4info		#679	2e29350	1148+,742-	17216
	mp4audioclip	#732		bbb6f24	1045+,1688-	14593
	mp42aac		#751	61b2012	0+,6-	14424
Mujs	mujs	#65		8c27b126	27+,16-	6482
	mujs	#141		832e0690	87+,27-	6996
	mujs		#145	4c7f6be	41+,5-	7319
	mujs	#166		3f71a1c9	260+,47-	15791
Libjpeg	cjpeg	#493		88ae609	1999+,228-	4982
	jpegtran	#636		88ae609	1999+,228-	6075
Tcpreplay	tcprewrite		#702	0a65668a	282+,148-	4110
	tcprewrite	#718		2c76868d	45+,45-	4030
	tcpprep	#756		16442ac3	312+,338-	1855
	tcpreplay	#772		4f9158da	1+,2-	2240
Libxml2	xmllint	#535		9a82b94a	253+,176-	66472
	xmllint	#550		7e3f469b	32+,38-	66150
Poppler	pdfunite	#1282		3d35d209	16+,0-	44103
	pdfunite		#1289	3cae7773	31+,2-	1015
	pdftops	#1303		e674ca64	71+,80-	42235
	pdftoppm	#1305		aaf2e808	31+,2-	37682
	pdftoppm		#1381	245abada	20+,45-	51098
ImageMagick	magick	#6075		a107b941	103+,134-	134594

- Crash site often **differ from** the commit change site
- BIC often contains **multiple** change sites

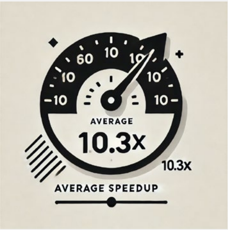
Summary of WAFLGo



Fuzzing framework for program commit

Experimental Result

- Highest bug reproduction success rate
- Average speedup of 10.3x
- Seven new bugs, 4 CVEs



Email Address:
xiangyi0406@zju.edu.cn