



# Automagiczne znajdowanie błędów z użyciem fuzzera AFL

Andrzej Jackowski

# Fuzz

## Generacja Losowych Wejść

- Wypisuje losowe ciągi znaków
- Potrafi robić odstępy czasowe
- Można zawęzić zbiór znaków
- Można dać stałego seeda

“An Empirical Study of the Reliability of UNIX Utilities”

Barton P. Miller, Lars Fredriksen, Bryan So, 1990, Univ. of Wisconsin, Madison



**GitHub**

<https://github.com/alipourm/fuzz>

Na Mint 18.2 potrzebne były zmiany:

```
GRUB_CMDLINE_LINUX="pty.legacy_count=256" /etc/default/grub  
sudo chmod 777 /dev/pty* && sudo chmod 777 /dev/tty*  
ptyjig.c linia 187: int executing -> volatile int executing
```

as

emacs

Utility	VAX (v)	Sun (s)	HP (h)	i386 (x)	AIX 1.1 (a)	Sequent (d)
adb	•○	•	•	○	•	•
as	•			•	•	•
awk						
bc				•○		
bib						
calendar						
cat						
cb	•		•	•	○	•
cc						
/lib/ccom						•com
checkeq						
checknr						
col	•○	•	•	•○	•	•
colcrt						
colrm						
comm						
compress						
/lib/cpp						
csh	•○	○	○	•	○	○
dbx		*				
dc				○		
deqn		•				
deroff	•	•	•	•	•	•
diction	•		•			•
diff						
ditroff	•○	•				
dtbl						
emacs	•	•	○			
eqn		•	•	•		
expand						
f77	•					
fmt						
fold						
ftp	•	•	•		•	•
graph						
grep						
gm						
head						
ideal						
indent	•○	•○	•			•
join		⊕				
latex						
lex	•	•	•	•	•	•
lint						
lisp						
look	•	○	•	•		•

Table 2: List of Utilities Tested and the Systems on which They Were Tested (part 1)

• = utility crashed, ○ = utility hung, \* = crashed on SunOS 3.2 but not on SunOS 4.0,  
 ⊕ = crashed only on SunOS 4.0, not 3.2. - = utility unavailable on that system.  
 ! = utility caused the operating system to crash.

make

telnet

vi

od 24.5%  
do 33.3%

Utility	VAX (v)	Sun (s)	HP (h)	i386 (x)	AIX 1.1 (a)	Sequent (d)
m4				•		
mail						
make			•			
more					-	
nm						
nroff						
pc				•		
pic						
plot	-	○	•			
pr			•			
prolog	•○	•○	•○			
psdit						
ptx		•	•	○		○
refer	•	*	•			!•
rev						
sed						
sh						
soelim						
sort						
spell	•○	•	•	○	•	•
spline						
split						
sql						
strings						
strip						
style	•		•			•
sum						
tail						
tbl						
tee						
telnet	•	•	•		•	○
tex						
tr						
troff						
tsort	•	*	•	•	•	•
ul	•	•	•			•
uniq	•	•	•	•	•	•
units	•○	•	•	•	•	•
vgrind	•					
vi	•		•			
wc						
yacc						
# tested	85	83	75	55	49	73
# crashed/hung	25	21	25	16	12	19
%	29.4%	25.3%	33.3%	29.1%	24.5%	26.0%

Table 2: List of Utilities Tested and the Systems on which They Were Tested (part 2)

• = utility crashed, ○ = utility hung, \* = crashed on SunOS 3.2 but not on SunOS 4.0,  
 ⊕ = crashed only on SunOS 4.0, not 3.2. - = utility unavailable on that system.  
 ! = utility caused the operating system to crash.

as

Utility	SunOS		HP-UX		AIX		Solaris		Irix	Ultrix	NEXT	GNU	Linux
	90	95	90	95	90	95	95	95	95	95	95	95	95
adb	●	●	●	●	×	×	●	○	×	×	×	×	×
as					●						●		
awk												×	
bc											●		×
bib		●	×		×	×	×	×				×	
calendar												×	×
cat													
cb			●		○	○			●		●	×	×
cc											●		
ccom					×	×	○				●		×
checkeq				×		●		×				×	×
checknr					×			×				×	×
col	●	○	●		●	●	●	●		○	●	×	
colcrt				×	×	×	×	×	×			×	
colrm				×	×		×	×				×	
comm													
compress					×								
cpp												×	
csh	○		○		○								
ctags	×		×		×	×	●				●		○
ctree	×	×	×	×	×		×	×				×	×
dbx	●		×								×		●
dc								●			●	●	×
deroff	●	●	●		●		●		●		●	×	×
diction	×	●	●	●	×		×	×	●		●	×	×
diff													
ditroff	●	●	×	●		●	●		●		●		
eqn	●	●	●	●		○			●	●	●		×
ex			●									×	

Table 3: List of Utilities Tested and Results of Those Tests

● = crashed, ○ = hung, × = not available

telnet

od 6%  
do 43%

Utility	SunOS		HP-UX		AIX		Solaris	Irix	Ultrix	NEXT	GNU	Linux
	90	95	90	95	90	95	95	95	95	95	95	95
sed												
sh												
soelim						x						x
sort												
spell	●	●	●		●		●		●	●	x	x
spline				x	x			x		●		x
split												
strings					x		●					
strip												
style	x	●	●	●	x		x	x		●	x	x
sum												
tail												
tbl												
tee												
telnet	●	●	●		●	●		●	●	●	x	
tex			x		x	x	x	x			x	
tr												
tsort	●		●		●					●	x	
ul	●	●	●	●	x	●	●	●	●	●	x	●
uniq	●	●	●		●		●	●	●	●☾		
units	●	●	●		●	●	●	●		●	x	x
vgrind			x		x					●	x	x
wc												
yacc												
# tested	77	80	72	74	49	74	70	60	80	75	47	55
# crash/hang	22	18	24	13	12	15	16	9	17	32	3	5
%	29%	23%	33%	18%	24%	20%	23%	15%	21%	43%	6%	9%

Table 3: List of Utilities Tested and Results of Those Tests

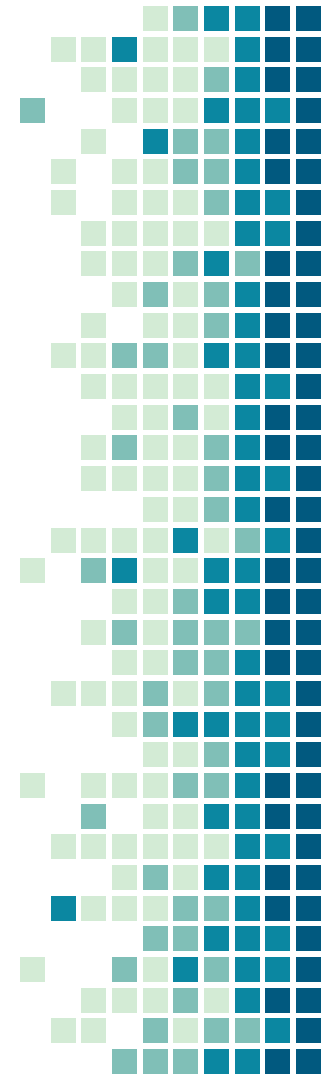
● = crashed, ○ = hung, × = not available

# Mangleme

Program CGI  
~120 linii w C

Napisany w 2004  
Przez Michała Zalewskiego  
twórcę American Fuzzy Lop

Generuje HTML  
W celu znalezienia  
błędów w przeglądarkach

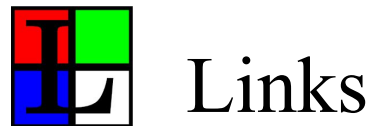


# Przykłady źle obsługiwanych plików HTML

```
<HTML>  
<TBODY>  
<COL SPAN=999999999>
```

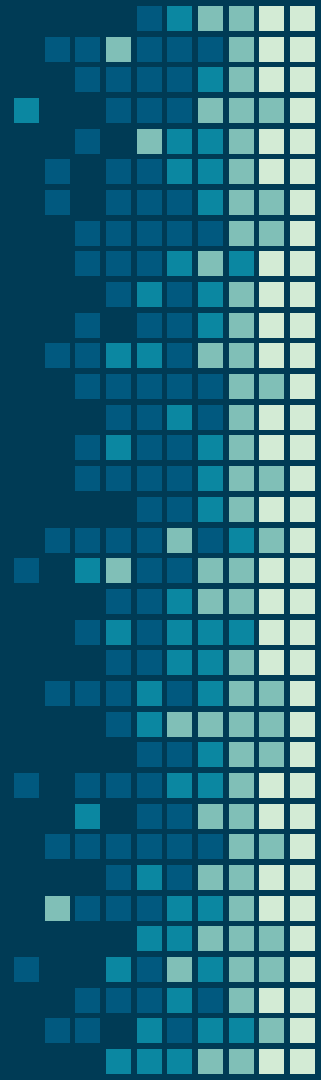
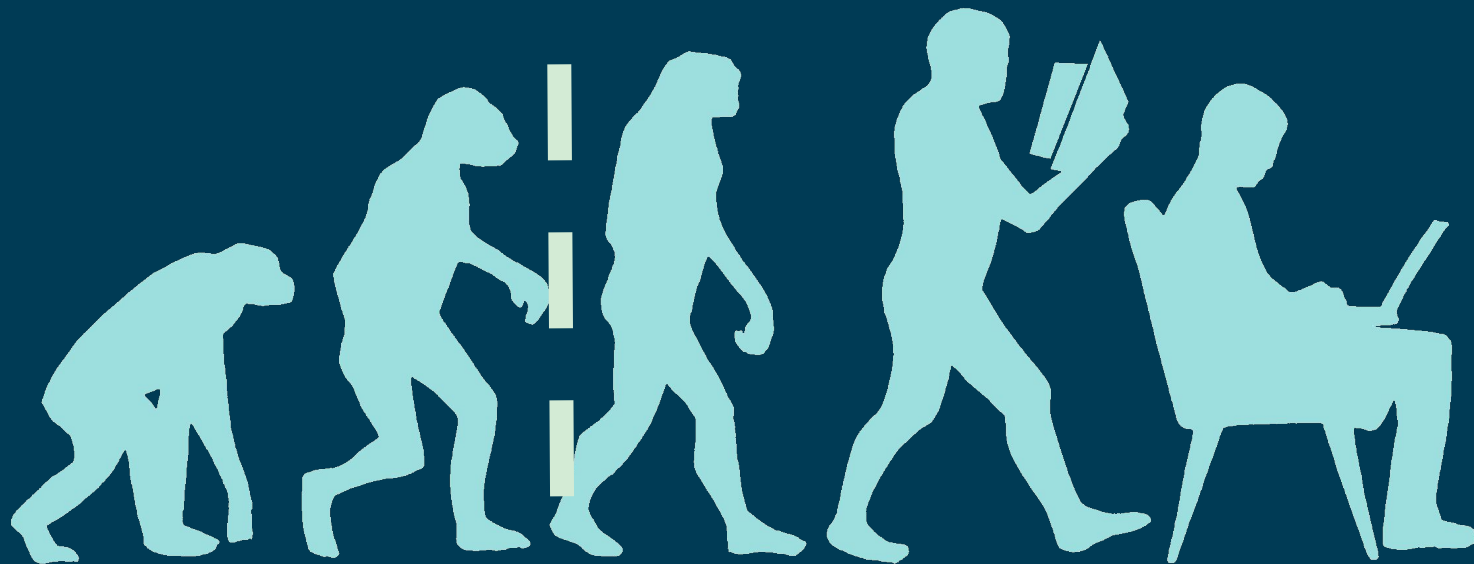
```
<HTML>  
<HEAD>  
<MARQUEE>  
<TABLE>  
<MARQUEE HEIGHT=100000000>  
... w sumie 11 razy MARQUEE  
<MARQUEE HEIGHT=100000000>  
<TBODY>  
Attack of the marquees!
```

```
<HTML>  
<TABLE>  
<TR>  
<TD ROWSPAN=2000000000>
```



Simple  
Fuzzing

Smart  
Fuzzing



# Smart Fuzzing

## Instrumentacja

Wykorzystanie wiedzy o zachowaniu programu przy generowaniu fuzz testów

## Uczenie maszynowe

Między innymi algorytmy genetyczne i sieci neuronowe

## Heurystyki

Przyspieszone efekty fuzzingu dla typowych przypadków

## Interakcja z ludźmi

Czasami komputer nie da rady bez pomocy człowieka

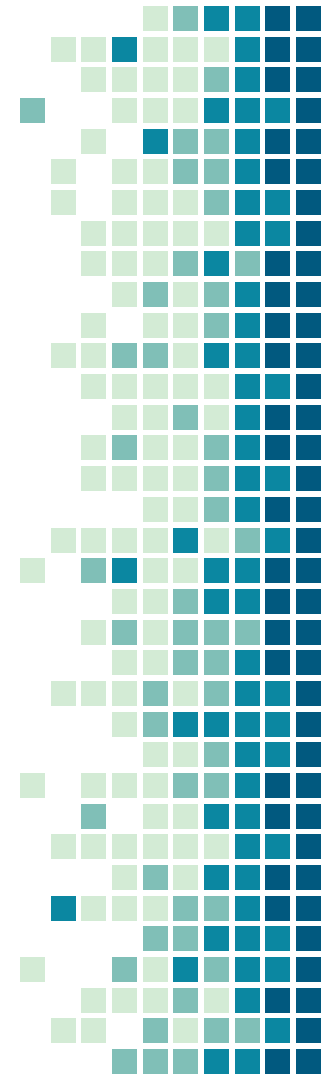
## SMT\* Solvers

Logiczne rozwiązywanie problemów decyzyjnych

\* Satisfiability modulo theories

## Inne

Dopasowane techniki dla konkretnego problemu



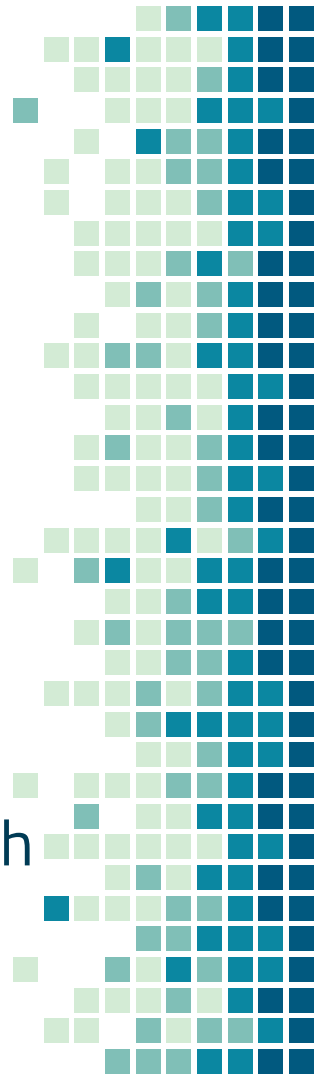




# AFL

## American Fuzzy Lop

- State of the Art
- Smart Fuzzing
- Open Source
- Integracja z GCC/Clang
- Znalazł bugi w 100+ programach





OpenSSL  
Cryptography and SSL/TLS Toolkit



flac  
free lossless audio codec

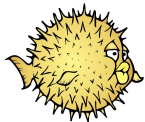


Perl

php



NGINX



OpenBSD

Qt



QEMU



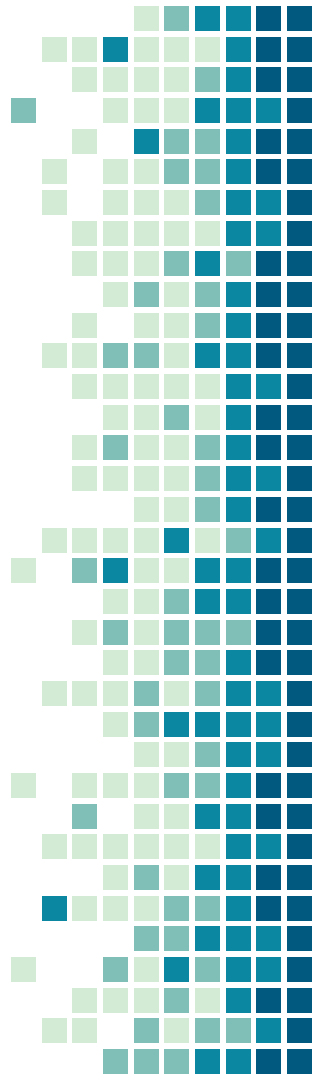
SQLite



Ext4  
File System



LibreOffice®

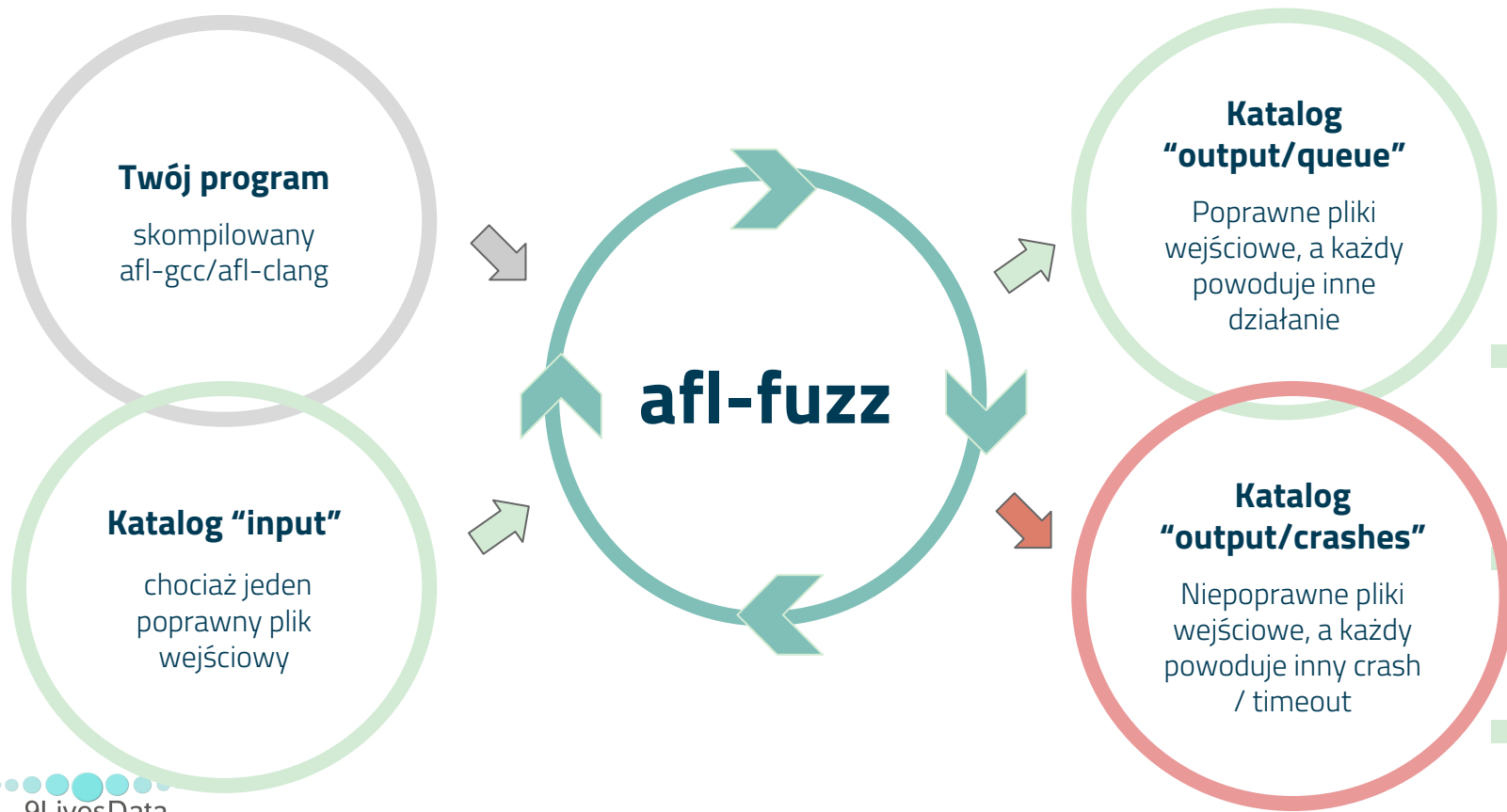




DALSZA CZĘŚĆ  
PREZENTACJI  
ZAWIERA JEDYNNIE  
UPROSZCZONY  
SCHEMAT  
DZIAŁANIA **AFL**

IMPLEMENTACJA  
ZAWIERA LICZNE  
OPTYMALIZACJE I  
HACKI O KTÓRYCH  
NIE ZDAŻĘ  
OPOWIEDZIEĆ

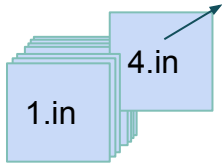
# Wejście/Wyjście AFL



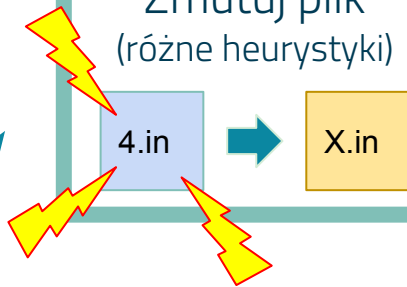
# Jak działa afl-fuzz?

start

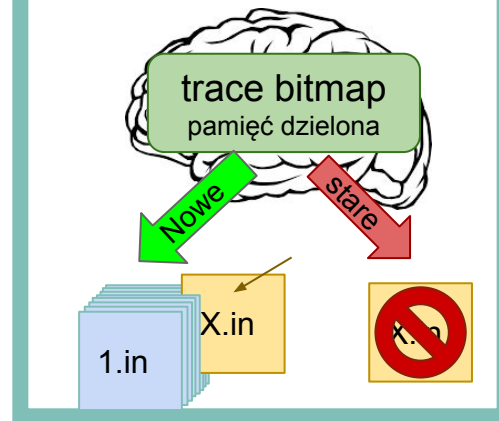
Wybierz plik z kolejki



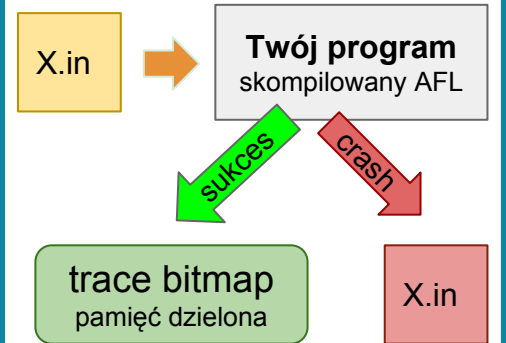
Zmutuj plik  
(różne heurystyki)



Porównaj zachowanie



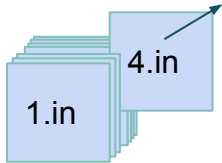
Uruchom program



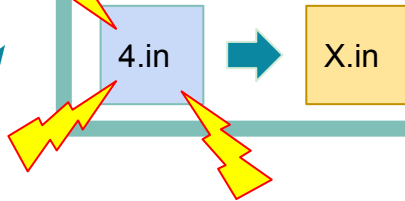
# Jak działa afl-fuzz?

start

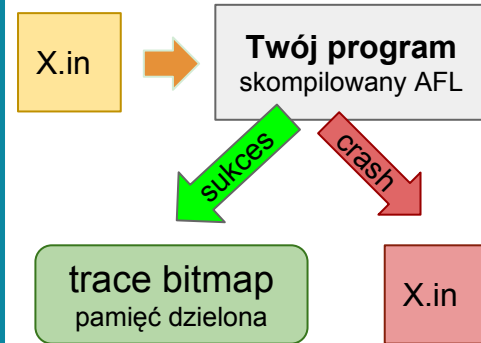
Wybierz plik z kolejki



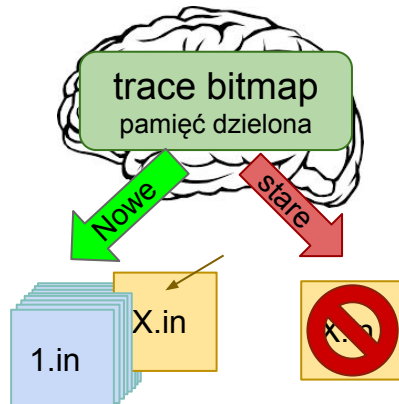
Zmutuj plik  
(różne heurystyki)



Uruchom program



Porównaj zachowanie



# Heurystyki mutacji pliku

## Deterministyczne (na początku)

### Odwracanie bitów

Okienko przesuwają się bity po bicie i odwracają grupy bitów różnej długości (1, 2, 4 itd.)

### Odwracanie bajtów

Okienko przesuwają się bajty po bajcie i odwracają grupy bitów różnej długości

### Arytmetyka

Dodajemy do grup bajtów wartości z przedziału  $(-35, 35)$ . Operacje zarówno grubo- jak i cienko-końcówkowe

### Magiczne stałe

Nadpisujemy grupy bajtów magicznymi stałymi np. 0, -1, 64, 100, 255

### Spustoszenie (HAVOC)

Losowe jednoczesne użycie deterministycznych mutacji w różnych miejscach pliku

### Splatanie (SPLICE)

Wyciąga **dwa** pliki z kolejki, uciną w losowych miejscach, skleja i jeszcze na koniec wykonuje HAVOC

## Losowe (w nieskończoność)



*In the past six years or so, I've also seen a fair number of academic papers that dealt with smart fuzzing [...] I'm unconvinced how practical most of these experiments were;*

*I suspect that many of them suffer from the bunny-the-fuzzer's curse of being cool on paper and in carefully designed experiments, but failing the ultimate test of being able to find new, worthwhile security bugs in otherwise well-fuzzed, real-world software.*

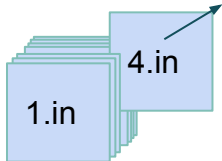
Michał Zalewski, AFL Historical notes



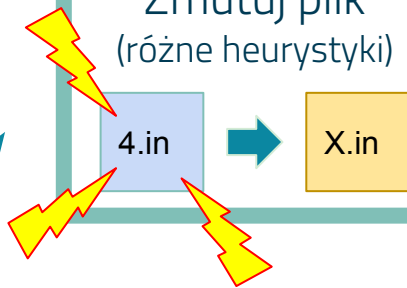
# Jak działa afl-fuzz?

start

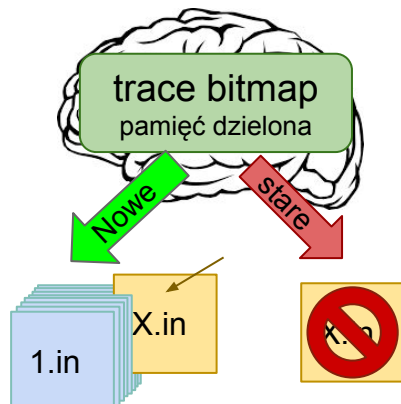
Wybierz plik z kolejki



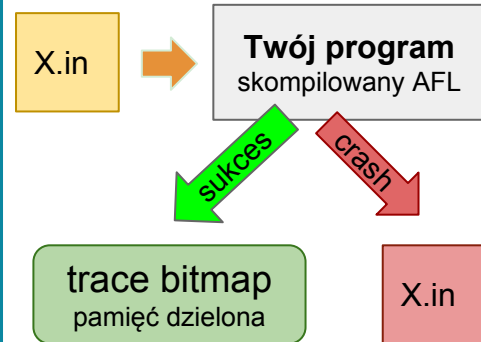
Zmutuj plik  
(różne heurystyki)



Porównaj zachowanie



Uruchom program



# afl-as to podstawa afl-gcc/afl-clang

```
int main(int argc, char *argv[]){  
    if(argc == 10) return 1;  
    return 0;  
}
```

gcc

```
main:  
    ; ustaw eax (32-bit) na 0  
    xorl    %eax, %eax  
    ; porównaj 10 i argc  
    cmpl    $10, %edi  
    ; ustaw 1 jeśli równe  
    sete    %al ; al to 8 bitów z eax  
    ; zwracamy eax  
    ret
```

afl-as

```
main:  
    /* --- AFL TRAMPOLINE (64-BIT) --- */  
    .align 4  
    leaq -(128+24)(%rsp), %rsp  
    movq %rdx, 0(%rsp)  
    movq %rcx, 8(%rsp)  
    movq %rax, 16(%rsp)  
    movq $0x000005b0, %rcx  
    call __afl_maybe_log  
    movq 16(%rsp), %rax  
    movq 8(%rsp), %rcx  
    movq 0(%rsp), %rdx  
    leaq (128+24)(%rsp), %rsp  
    /* --- END --- */
```

```
xorl    %eax, %eax  
cmpl    $10, %edi  
sete    %al  
ret
```

+ 250 linii kodu \_\_afl\_maybe\_log

as

## Wzbogacanie asm w afl-as

- Czytaj plik \*.s linia po linii
- Jeśli znajdziesz rozgałęzienie kodu (np. jmp):
  - Wylosuj **id** od 0 do 65535
  - Wstaw **trampoline\*** z wylosowanym id
- Dodaj na koniec kod implementujący trampoline

## Gdy program trafi na **trampoline**

- Tworzy parę  $p = \langle \text{poprzednie\_id}, \text{id} \rangle$
- Inkrementuje **licznik** odpowiadający  $p$
- **Licznik** jest trzymany w **trace\_bitmap** (pamięć dzielona przez nasz program i AFL)

### Trace\_bitmap:

$\langle \text{id1}, \text{id1} \rangle \rightarrow \text{licznik\_1};$

$\langle \text{id1}, \text{id2} \rangle \rightarrow \text{licznik\_2};$

$\langle \text{id2}, \text{id2} \rangle \rightarrow \text{licznik\_3};$

## Po zakończeniu działania programu

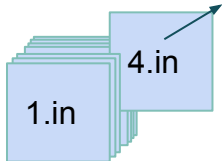
- AFL porównuje trace\_bitmap z mapami pozostałych wykonań programu
- **Zmutowane plik wejściowy** jest zapisywane do *output/queue* tylko jeśli ma nowe wartości **liczników** w **trace\_bitmap**

# Jak działa afl-fuzz?

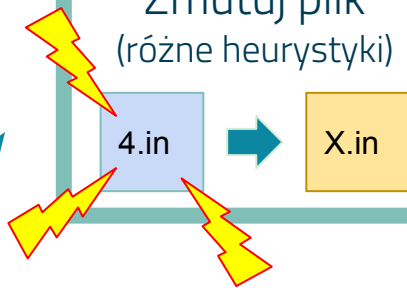


start

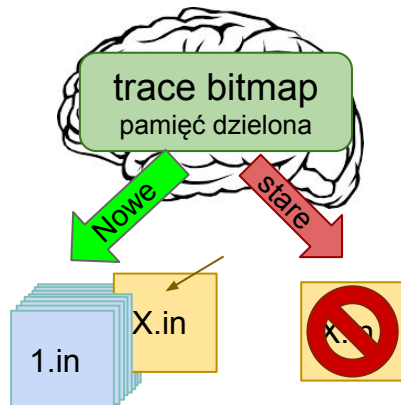
Wybierz plik z kolejki



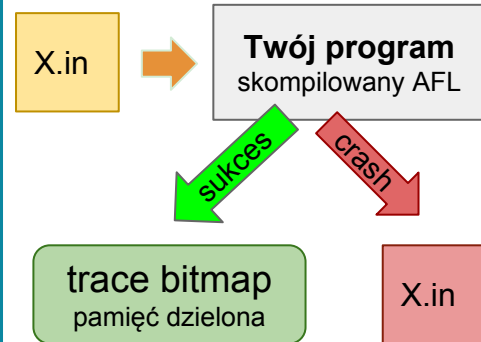
Zmutuj plik  
(różne heurystyki)



Porównaj zachowanie



Uruchom program



# Wykrywanie błędów podczas fuzzingu



## Asercje w kodzie

Programiści dostarczają informacje pozwalające na crash programu podczas wykonania

## Limit czasu i zasobów

Sprawdzamy czy nie przekroczyliśmy limitu zasobów oraz wykrywamy nieskończone pętle i deadlocki.

## Sanitizery

- Address Sanitizer
- Memory Sanitizer
- Thread Sanitizer
- Undefined B. Sanitizer
- ....
- nawet libstdc++ debug mode

## Weryfikacja wyjścia

- Poprawność formatu
- Rozmiar wyjścia
- Przybliżone wartości
- ...



# Czas na demo

Na podstawie

“How Heartbleed could've been found”

<https://blog.hboeck.de/archives/868-How-Heartbleed-couldve-been-found.html>

# HOW THE HEARTBLEED BUG WORKS:

SERVER, ARE YOU STILL THERE?  
IF SO, REPLY "POTATO" (6 LETTERS).



Secure connection using key "4538538374224".  
User Meg wants these 6 letters: **POTATO**. User  
Isabel wants pages about "irl games". Unlocking  
secure records with master key 5130985733435.  
Note: I have not read this message yet.



HMM...



BIRD



User Olivia from London wants pages about "the  
pees in car why". Note: Files for IP 375.381.  
83.17 are in /tmp/files-3843. User Meg wants  
these 4 letters: **BIRD**. There are currently 34  
connections open. User Brendan uploaded the file  
645f4e7a (contents: 834ba962e2c0b9ff89b334f8

his pages about "boats". User Meg wants  
secure connection using key "4538538374224".  
User Meg wants these 6 letters: **POTATO**. User  
Isabel wants pages about "irl games". Unlocking  
secure records with master key 5130985733435.  
Note: I have not read this message yet.



POTATO



SERVER, ARE YOU STILL THERE?  
IF SO, REPLY "HAT" (500 LETTERS).



Secure connection using key "4538538374224".  
User Meg wants these 500 letters: **HAT**. Lucas  
requests the "missed connections" page. Eve  
(administrator) wants to set server's master  
key to "14835038534". Isabel wants pages about  
snakes but not too long". User Karen wants to  
change account password to "G0tM1k3". User



SERVER, ARE YOU STILL THERE?  
IF SO, REPLY "BIRD" (4 LETTERS).



User Olivia from London wants pages about "the  
pees in car why". Note: Files for IP 375.381.  
83.17 are in /tmp/files-3843. User Meg wants  
these 4 letters: **BIRD**. There are currently 34  
connections open. User Brendan uploaded the file  
645f4e7a (contents: 834ba962e2c0b9ff89b334f8



HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about snakes but not too long". User Karen wants to change account password to "G0tM1k3". User



Secure connection using key "4538538374224".  
User Meg wants these 500 letters: **HAT**. Lucas  
requests the "missed connections" page. Eve  
(administrator) wants to set server's master  
key to "14835038534". Isabel wants pages about  
snakes but not too long". User Karen wants to  
change account password to "G0tM1k3". User

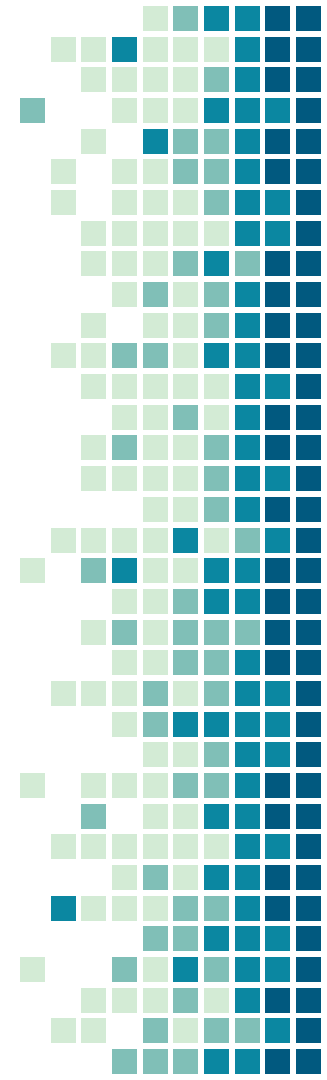
# Scenariusz demo

- **openssl-1.0.1f** oraz **selftls** skompilowane AFL z włączonym ASANem
- Katalog in/ zawiera tylko jeden plik
- Uruchomienie fuzzera AFL
- Krótki przewodnik po interfejsie
- Krótki przewodnik po plikach

Na podstawie

“How Heartbleed could've been found”

<https://blog.hboeck.de/archives/868-How-Heartbleed-couldve-been-found.html>





```

shadowword@shadowwordpl ~/HEARTBLEED/openssl-1.0.1f $ ./runError.sh afl_out/master/crashes/id\:000000\,sig\:06\,src\:000032\,op\:arith8\,pos\:0\,val\:+3
=====
==16345== ERROR: AddressSanitizer: unknown-crash on address 0x60820001220b at pc 0x7f4a98de23f7 bp 0x7fffa0b72380 sp 0x7fffa0b71b40
READ of size 25600 at 0x60820001220b thread T0
#0 0x7f4a98de23f6 (/usr/lib/x86_64-linux-gnu/libasan.so.0+0xe3f6)
#1 0x410dad in memcpy /usr/include/x86_64-linux-gnu/bits/string3.h:51
#2 0x410dad in tls1_process_heartbeat /home/shadowword/HEARTBLEED/openssl-1.0.1f/ssl/t1_lib.c:2586
#3 0x49d16c in ssl3_read_bytes /home/shadowword/HEARTBLEED/openssl-1.0.1f/ssl/s3_pkt.c:1092
#4 0x4a0ca3 in ssl3_get_message /home/shadowword/HEARTBLEED/openssl-1.0.1f/ssl/s3_both.c:457
#5 0x465a2d in ssl3_get_client_hello /home/shadowword/HEARTBLEED/openssl-1.0.1f/ssl/s3_srvr.c:941
#6 0x4764b3 in ssl3_accept /home/shadowword/HEARTBLEED/openssl-1.0.1f/ssl/s3_srvr.c:357
#7 0x402487 in main /home/shadowword/HEARTBLEED/openssl-1.0.1f/selftls.c:95
#8 0x7f4a9882cec4 (/lib/x86_64-linux-gnu/libc.so.6+0x21ec4)
#9 0x40299c in _start (/home/shadowword/HEARTBLEED/openssl-1.0.1f/selftls+0x40299c)
0x608200016748 is located 0 bytes to the right of 17736-byte region [0x608200012200,0x608200016748)
allocated by thread T0 here:
#0 0x7f4a98de941a (/usr/lib/x86_64-linux-gnu/libasan.so.0+0x1541a)
#1 0x4d7946 in CRYPTO_malloc /home/shadowword/HEARTBLEED/openssl-1.0.1f/crypto/mem.c:308

```

Zawartość pliku powodującego crash (HEX)

0x18037F00020164

0x6400 to 25600

Patrz **READ of size 25600**

**#define TLS1\_RT\_HEARTBEAT 24 {0x18}**

openssl-1.0.1f, linia 325: ssl/ssl3.h

**#define TLS1\_HB\_REQUEST 1**

openssl-1.0.1f, linia 343: ssl/ssl3.h

# Ścieżka prowadząca do błędu

- Początkowa zawartość pliku: 12341234123412
  - 12341234123412 -> 123412 (0x313233331320A)
  - 0x313233331320A -> 0x35037F0000200A
  - 0x35037F0000200A -> 0x12037F0002200A
  - 0x12037F0002200A -> 0x15037F0002200A
  - 0x15037F0002200A -> 0x15037F0002010A
  - 0x15037F0002010A -> 0x15037F00020164
  - 0x15037F00020164 -> 0x18037F00020164
- { minimalizacja }
- { Havoc\* }
- { Havoc\* }
- { Arith +3 }
- { Arith -31 }
- { Magic 100 }
- { Arith +3 }

0x18037F00020164

#define TLS1\_RT\_HEARTBEAT 24 {0x18}  
openssl-1.0.1f, linia 325: ssl/ssl3.h

#define TLS1\_HB\_REQUEST 1  
openssl-1.0.1f, linia 343: ssl/ssl3.h

0x6400 to 25600  
Patrz **READ of size 25600**

\* przypuszczalnie:

0x31 -> 0x35 to arith +4  
0x32 -> 0x03 to magic\_const 0 i arith +3  
0x33 -> 0x7F to magic\_const 127  
0x31 -> 0x00 to magic\_const 0  
0x00 to wstawione magic\_cons 0  
0x32 -> 0x20 to magic\_const 32

0x35 -> 0x to arith -35 (53 - 35 = 18)  
0x00 -> 02 to arith +2

~~Fuzzy Logic  
All things have everything~~



**Właśnie  
nie!**

## Co (może) znajdzie AFL?

- Problemy z zarządzaniem pamięcią
- Zła obsługa NULLi
- Źle łapane wyjątki
- Zakleszczenia
- Nieskończone pętle
- Undefined behaviour
- Niepoprawne zarządzanie zasobami

Czyli tylko to do czego  
przygotujemy weryfikację!

## Czego AFL nie sprawdzi?

- **Czy aplikacja robi to co powinna**
- Dlatego i tak trzeba zrobić:
  - Testy jednostkowe
  - Testy komponentów
  - Testy integracyjne
  - Testy systemowe
  - Testy manualne

Ale za to może znajdzie błędy,  
których powyższe testy nie wykryją!

# Kiedy warto robić fuzzing?

- Niezaufane osoby mają dostęp do programu
- Błąd w programie powoduje duże koszty
- Program nie jest dostatecznie dotestowany
- Program działa szybko (najlepiej >1000 plików na sekundę)
- Format pliku wejściowego jest skomplikowany
- Pliki wejściowe są małe (najlepiej < 100KB)
- Zachowuje się deterministycznie
- Program ma dużo asercji



# Jak było u mnie?

- Niezaufane osoby mają dostęp do programu { ale mógł być źle skonfigurowany }
  - Błąd w programie powoduje duże koszty
  - Program nie jest dostatecznie dotestowany
  - Program działa szybko (najlepiej >1000 plików na sekundę)
  - Format pliku wejściowego jest skomplikowany
  - Pliki wejściowe są małe (najlepiej < 100KB) { typowy input > 100MB }
  - Zachowuje się deterministycznie
  - Program ma dużo asercji
- + 8 wersji programu (algorytmów), każdy testowany osobno

Symantec Netbackup 7.5



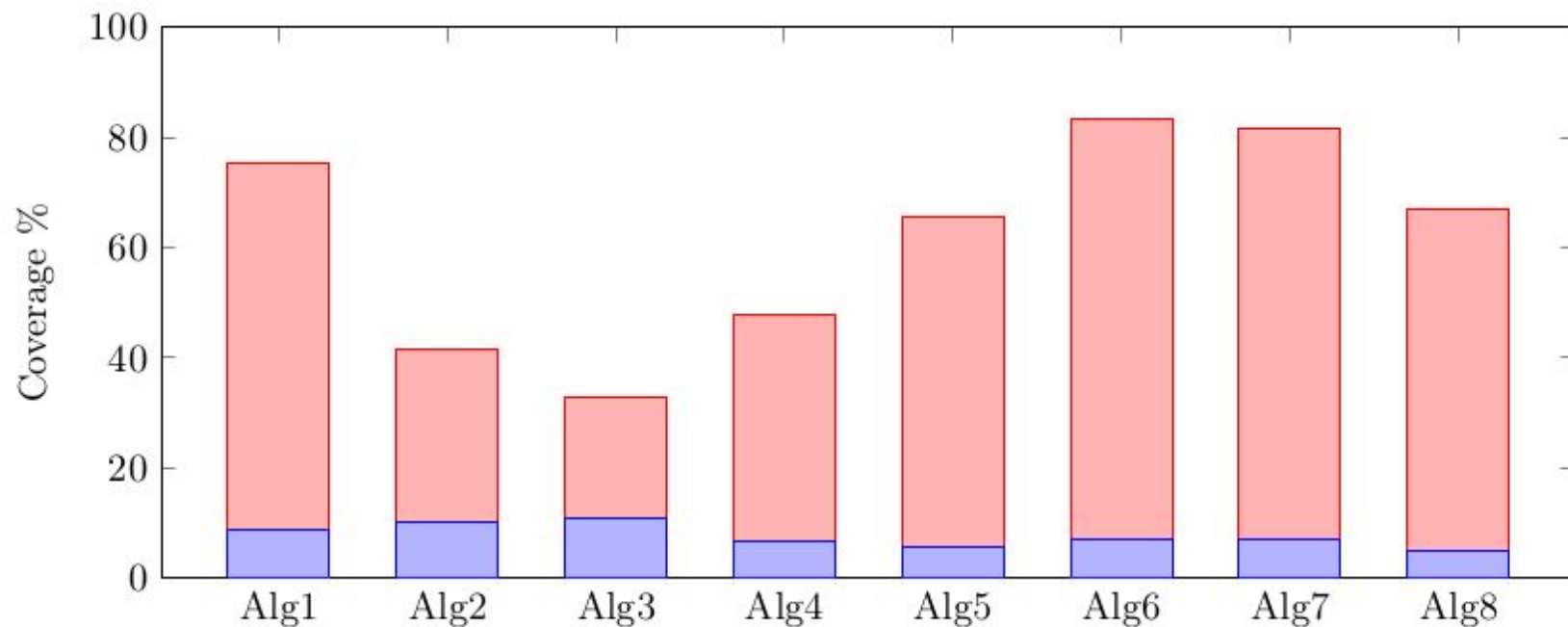
COMMVAULT™



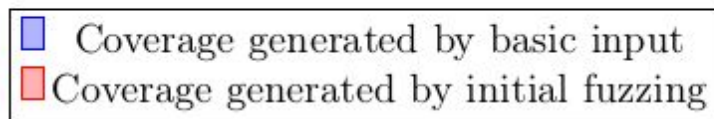
# No to jedziemy!



# Pokrycie kodu po 90 godzinach



Znaleziono  
błędy: 0



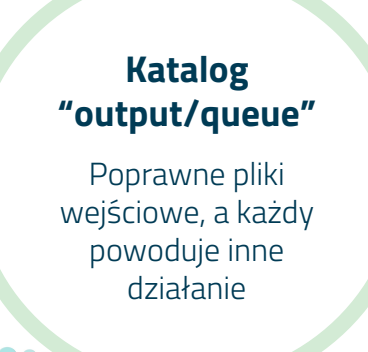




**Twój program**


skompilowany z

- fprofile-arcs
- ftest-coverage



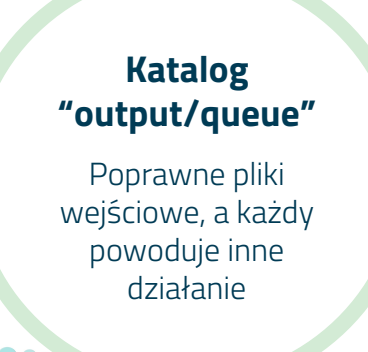

**Katalog  
"output/queue"**

Poprawne pliki  
wejściowe, a każdy  
powoduje inne  
działanie




**Katalog  
"output/queue"**

Poprawne pliki  
wejściowe, a każdy  
powoduje inne  
działanie



**Katalog  
"output/queue"**

Poprawne pliki  
wejściowe, a każdy  
powoduje inne  
działanie



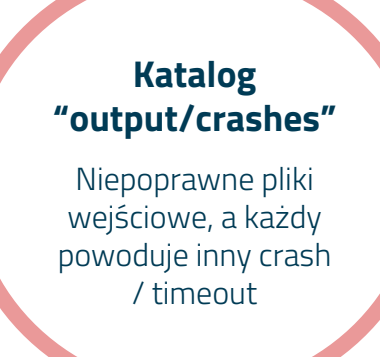
# afl-cov

GitHub



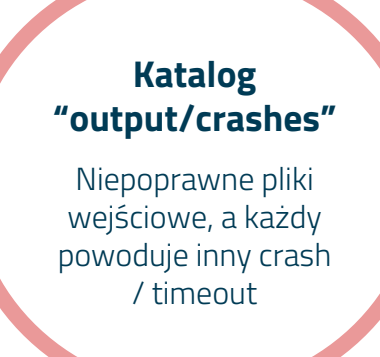
# afl-cov

GitHub



**Katalog  
"output/crashes"**

Niepoprawne pliki  
wejściowe, a każdy  
powoduje inny crash  
/ timeout

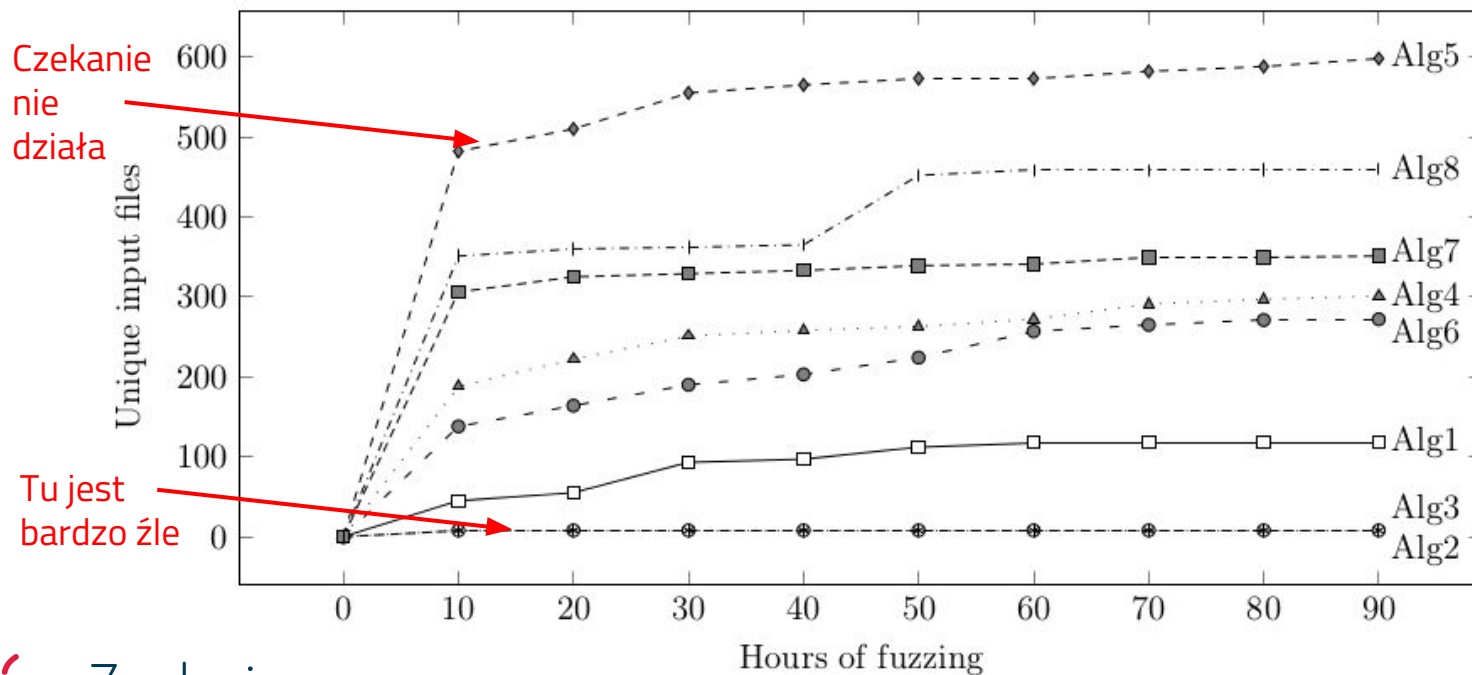


**Katalog  
"output/crashes"**

Niepoprawne pliki  
wejściowe, a każdy  
powoduje inny crash  
/ timeout

Test: coverage.total		Lines:	369	84	110
Date: 2011-06-04		Functions:	54	110	85.71
Filename	Line Coverage %	Functions %			
1	0.0%	0.0%	0.0%	0.0%	0.0%
2	0.0%	0.0%	0.0%	0.0%	0.0%
3	0.0%	0.0%	0.0%	0.0%	0.0%
4	75.0%	50.0%	50.0%	0.0%	0.0%
5	100.0%	100.0%	100.0%	0.0%	0.0%
6	85.0%	17.1%	22.2%	0.0%	0.0%
7	100.0%	21.0%	85.0%	0.0%	0.0%
8	100.0%	51.0%	85.0%	0.0%	0.0%
9	100.0%	24.0%	25.0%	0.0%	0.0%
10	0.0%	0.0%	0.0%	0.0%	0.0%
11	0.0%	0.0%	0.0%	0.0%	0.0%
12	0.0%	0.0%	0.0%	0.0%	0.0%
13	0.0%	0.0%	0.0%	0.0%	0.0%
14	0.0%	0.0%	0.0%	0.0%	0.0%
15	0.0%	0.0%	0.0%	0.0%	0.0%
16	0.0%	0.0%	0.0%	0.0%	0.0%
17	0.0%	0.0%	0.0%	0.0%	0.0%
18	0.0%	0.0%	0.0%	0.0%	0.0%
19	0.0%	0.0%	0.0%	0.0%	0.0%
20	0.0%	0.0%	0.0%	0.0%	0.0%
21	0.0%	0.0%	0.0%	0.0%	0.0%
22	0.0%	0.0%	0.0%	0.0%	0.0%
23	0.0%	0.0%	0.0%	0.0%	0.0%
24	0.0%	0.0%	0.0%	0.0%	0.0%
25	0.0%	0.0%	0.0%	0.0%	0.0%
26	0.0%	0.0%	0.0%	0.0%	0.0%
27	0.0%	0.0%	0.0%	0.0%	0.0%
28	0.0%	0.0%	0.0%	0.0%	0.0%
29	0.0%	0.0%	0.0%	0.0%	0.0%
30	0.0%	0.0%	0.0%	0.0%	0.0%
31	0.0%	0.0%	0.0%	0.0%	0.0%
32	0.0%	0.0%	0.0%	0.0%	0.0%
33	0.0%	0.0%	0.0%	0.0%	0.0%
34	0.0%	0.0%	0.0%	0.0%	0.0%
35	0.0%	0.0%	0.0%	0.0%	0.0%
36	0.0%	0.0%	0.0%	0.0%	0.0%
37	0.0%	0.0%	0.0%	0.0%	0.0%
38	0.0%	0.0%	0.0%	0.0%	0.0%
39	0.0%	0.0%	0.0%	0.0%	0.0%
40	0.0%	0.0%	0.0%	0.0%	0.0%
41	0.0%	0.0%	0.0%	0.0%	0.0%
42	0.0%	0.0%	0.0%	0.0%	0.0%
43	0.0%	0.0%	0.0%	0.0%	0.0%
44	0.0%	0.0%	0.0%	0.0%	0.0%
45	0.0%	0.0%	0.0%	0.0%	0.0%
46	0.0%	0.0%	0.0%	0.0%	0.0%
47	0.0%	0.0%	0.0%	0.0%	0.0%
48	0.0%	0.0%	0.0%	0.0%	0.0%
49	0.0%	0.0%	0.0%	0.0%	0.0%
50	0.0%	0.0%	0.0%	0.0%	0.0%
51	0.0%	0.0%	0.0%	0.0%	0.0%
52	0.0%	0.0%	0.0%	0.0%	0.0%
53	0.0%	0.0%	0.0%	0.0%	0.0%
54	0.0%	0.0%	0.0%	0.0%	0.0%
55	0.0%	0.0%	0.0%	0.0%	0.0%
56	0.0%	0.0%	0.0%	0.0%	0.0%
57	0.0%	0.0%	0.0%	0.0%	0.0%
58	0.0%	0.0%	0.0%	0.0%	0.0%
59	0.0%	0.0%	0.0%	0.0%	0.0%
60	0.0%	0.0%	0.0%	0.0%	0.0%
61	0.0%	0.0%	0.0%	0.0%	0.0%
62	0.0%	0.0%	0.0%	0.0%	0.0%
63	0.0%	0.0%	0.0%	0.0%	0.0%
64	0.0%	0.0%	0.0%	0.0%	0.0%
65	0.0%	0.0%	0.0%	0.0%	0.0%
66	0.0%	0.0%	0.0%	0.0%	0.0%
67	0.0%	0.0%	0.0%	0.0%	0.0%
68	0.0%	0.0%	0.0%	0.0%	0.0%
69	0.0%	0.0%	0.0%	0.0%	0.0%
70	0.0%	0.0%	0.0%	0.0%	0.0%
71	0.0%	0.0%	0.0%	0.0%	0.0%
72	0.0%	0.0%	0.0%	0.0%	0.0%
73	0.0%	0.0%	0.0%	0.0%	0.0%
74	0.0%	0.0%	0.0%	0.0%	0.0%
75	0.0%	0.0%	0.0%	0.0%	0.0%
76	0.0%	0.0%	0.0%	0.0%	0.0%
77	0.0%	0.0%	0.0%	0.0%	0.0%
78	0.0%	0.0%	0.0%	0.0%	0.0%
79	0.0%	0.0%	0.0%	0.0%	0.0%
80	0.0%	0.0%	0.0%	0.0%	0.0%
81	0.0%	0.0%	0.0%	0.0%	0.0%
82	0.0%	0.0%	0.0%	0.0%	0.0%
83	0.0%	0.0%	0.0%	0.0%	0.0%
84	0.0%	0.0%	0.0%	0.0%	0.0%
85	0.0%	0.0%	0.0%	0.0%	0.0%
86	0.0%	0.0%	0.0%	0.0%	0.0%
87	0.0%	0.0%	0.0%	0.0%	0.0%
88	0.0%	0.0%	0.0%	0.0%	0.0%
89	0.0%	0.0%	0.0%	0.0%	0.0%
90	0.0%	0.0%	0.0%	0.0%	0.0%
91	0.0%	0.0%	0.0%	0.0%	0.0%
92	0.0%	0.0%	0.0%	0.0%	0.0%
93	0.0%	0.0%	0.0%	0.0%	0.0%
94	0.0%	0.0%	0.0%	0.0%	0.0%
95	0.0%	0.0%	0.0%	0.0%	0.0%
96	0.0%	0.0%	0.0%	0.0%	0.0%
97	0.0%	0.0%	0.0%	0.0%	0.0%
98	0.0%	0.0%	0.0%	0.0%	0.0%
99	0.0%	0.0%	0.0%	0.0%	0.0%
100	0.0%	0.0%	0.0%	0.0%	0.0%
101	0.0%	0.0%	0.0%	0.0%	0.0%
102	0.0%	0.0%	0.0%	0.0%	0.0%
103	0.0%	0.0%	0.0%	0.0%	0.0%
104	0.0%	0.0%	0.0%	0.0%	0.0%
105	0.0%	0.0%	0.0%	0.0%	0.0%
106	0.0%	0.0%	0.0%	0.0%	0.0%
107	0.0%	0.0%	0.0%	0.0%	0.0%
108	0.0%	0.0%	0.0%	0.0%	0.0%
109	0.0%	0.0%	0.0%	0.0%	0.0%
110	0.0%	0.0%	0.0%	0.0%	0.0%
111	0.0%	0.0%	0.0%	0.0%	0.0%
112	0.0%	0.0%	0.0%	0.0%	0.0%
113	0.0%	0.0%	0.0%	0.0%	0.0%
114	0.0%	0.0%	0.0%	0.0%	0.0%
115	0.0%	0.0%	0.0%	0.0%	0.0%
116	0.0%	0.0%	0.0%	0.0%	0.0%
117	0.0%	0.0%	0.0%	0.0%	0.0%
118	0.0%	0.0%	0.0%	0.0%	0.0%
119	0.0%	0.0%	0.0%	0.0%	0.0%
120	0.0%	0.0%	0.0%	0.0%	0.0%
121	0.0%	0.0%	0.0%	0.0%	0.0%
122	0.0%	0.0%	0.0%	0.0%	0.0%
123	0.0%	0.0%	0.0%	0.0%	0.0%
124	0.0%	0.0%	0.0%	0.0%	0.0%
125	0.0%	0.0%	0.0%	0.0%	0.0%
126	0.0%	0.0%	0.0%	0.0%	0.0%
127	0.0%	0.0%	0.0%	0.0%	0.0%
128	0.0%	0.0%	0.0%	0.0%	0.0%
129	0.0%	0.0%	0.0%	0.0%	0.0%
130	0.0%	0.0%	0.0%	0.0%	0.0%
131	0.0%	0.0%	0.0%	0.0%	0.0%
132	0.0%	0.0%	0.0%	0.0%	0.0%
133	0.0%	0.0%	0.0%	0.0%	0.0%
134	0.0%	0.0%	0.0%	0.0%	0.0%
135	0.0%	0.0%	0.0%	0.0%	0.0%
136	0.0%	0.0%	0.0%	0.0%	0.0%
137	0.0%	0.0%	0.0%	0.0%	0.0%
138	0.0%	0.0%	0.0%	0.0%	0.0%
139	0.0%	0.0%	0.0%	0.0%	0.0%
140	0.0%	0.0%	0.0%	0.0%	0.0%
141	0.0%	0.0%	0.0%	0.0%	0.0%
142	0.0%	0.0%	0.0%	0.0%	0.0%
143	0.0%	0.0%	0.0%	0.0%	0.0%
144	0.0%	0.0%	0.0%	0.0%	0.0%
145	0.0%	0.0%	0.0%	0.0%	0.0%
146	0.0%	0.0%	0.0%	0.0%	0.0%
147	0.0%	0.0%	0.0%	0.0%	0.0%
148	0.0%	0.0%	0.0%	0.0%	0.0%
149	0.0%	0.0%	0.0%	0.0%	0.0%
150	0.0%	0.0%	0.0%	0.0%	0.0%
151	0.0%	0.0%	0.0%	0.0%	0.0%
152	0.0%	0.0%	0.0%	0.0%	0.0%
153	0.0%	0.0%	0.0%	0.0%	0.0%
154	0.0%	0.0%	0.0%	0.0%	0.0%
155	0.0%	0.0%	0.0%	0.0%	0.0%
156	0.0%	0.0%	0.0%	0.0%	0.0%
157	0.0%	0.0%	0.0%	0.0%	0.0%
158	0.0%	0.0%	0.0%	0.0%	0.0%
159	0.0%	0.0%	0.0%	0.0%	0.0%
160	0.0%	0.0%	0.0%	0.0%	0.0%
161	0.0%	0.0%	0.0%	0.0%	0.0%
162	0.0%	0.0%	0.0%	0.0%	0.0%
163	0.0%	0.0%	0.0%	0.0%	0.0%
164	0.0%	0.0%	0.0%	0.0%	0.0%
165	0.0%	0.0%	0.0%	0.0%	0.0%
166	0.0%	0.0%	0.0%	0.0%	0.0%
167	0.0%	0.0%	0.0%	0.0%	0.0%
168	0.0%	0.0%	0.0%	0.0%	0.0%
169	0.0%	0.0%	0.0%	0.0%	0.0%
170	0.0%	0.0%	0.0%	0.0%	0.0%
171	0.0%	0.0%	0.0%	0.0%	0.0%
172	0.0%	0.0%	0.0%	0.0%	0.0%
173	0.0%	0.0%	0.0%	0.0%	0.0%
174	0.0%	0.0%	0.0%	0.0%	0.0%
175	0.0%	0.0%	0.0%	0.0%	0.0%
176	0.0%	0.0%	0.0%	0.0%	0.0%
177	0.0%	0.0%	0.0%	0.0%	0.0%
178	0.0%	0.0%	0.0%	0.0%	0.0%
179	0.0%	0.0%	0.0%	0.0%	0.0%
180	0.0%	0.0%	0.0%	0.0%	0.0%
181	0.0%	0.0%	0.0%	0.0%	0.0%
182	0.0%	0.0%	0.0%	0.0%	0.0%
183	0.0%	0.0%	0.0%	0.0%	0.0%
184	0.0%	0.0%	0.0%	0.0%	0.0%
185	0.0%	0.0%	0.0%	0.0%	0.0%
186	0.0%	0.0%	0.0%	0.0%	0.0%
187	0.0%	0.0%	0.0%	0.0%	0.0%
188	0.0%	0.0%	0.0%	0.0%	0.0%
189	0.0%	0.0%	0.0%	0.0%	0.0%
190	0.0%	0.0%	0.0%	0.0%	0.0%
191	0.0%	0.0%	0.0%	0.0%	0.0%
192	0.0%	0.0%	0.0%	0.0%	0.0%
193	0.0%	0.0%	0.0%	0.0%	0.0%
194	0.0%	0.0%	0.0%	0.0%	0.0%
195	0.0%	0.0%	0.0%	0.0%	0.0%
196	0.0%	0.0%	0.0%	0.0%	0.0%
197	0.0%	0.0%	0.0%	0.0%	0.0%
198	0.0%	0.0%	0.0%	0.0%	0.0%
199	0.0%	0.0%	0.0%	0.0%	0.0%
200	0.0%	0.0%	0.0%	0.0%	0.0%
201	0.0%	0.0%	0.0%	0.0%	0.0%
202	0.0%	0.0%	0.0%	0.0%	0.0%
203	0.0%	0.0%	0.0%	0.0%	0.0%
204	0.0%	0.0%	0.0%	0.0%	0.0%
205	0.0%	0.0%	0.0%	0.0%	0.0%
206	0.0%	0.0%	0.0%	0.0%	0.0%
207	0.0%	0.0%	0.0%	0.0%	0.0%
208	0.0%	0.0%	0.0%	0.0%	0.0%
209	0.0%	0.0%	0.0%	0.0%	0.0%
210	0.0%	0.0%	0.0%	0.0%	0.0%
211	0.0%	0.0%	0.0%	0.0%	0.0%
212	0.0%	0.0%	0.0%	0.0%	0.0%
213	0.0%	0.0%	0.0%	0.0%	0.0%
214	0.0%	0.0%	0.0%	0.0%	0.0%
215	0.0%	0.0%	0.0%	0.0%	0.0%
216	0.0%	0.0%	0.0%	0.0%	0.0%
217	0.0%	0.0%	0.0%	0.0%	0.0%
218	0.0%	0.0%	0.0%	0.0%	0.0%
219	0.0%	0.0%	0.0%	0.0%	0.0%
220	0.0%	0.0%	0.0%	0.0%	0.0%
221	0.0%	0.0%	0.0%	0.0%	0.0%
222	0.0%	0.0%	0.0%	0.0%	0.0%
223	0.0%	0.0%	0.0%	0.0%	0.0%
224	0.0%	0.0%	0.0%	0.0%	0.0%
225	0.0%	0.0%	0.0%	0.0%	0.0%
226	0.0%	0.0%	0.0%	0.0%	0.0%
227	0.0%	0.0%	0.0%	0.0%	0.0%
228	0.0%	0.0%	0.0%	0.0%	0.0%
229	0.0%	0.0%	0.0%	0.0%	0.0%
230	0.0%	0.0%	0.0%	0.0%	0.0%
231	0.0%	0.0%	0.0%	0.0%	0.0%
232	0.0%	0.0%	0.0%	0.0%	0.0%
233	0.0%	0.0%	0.0%	0.0%	0.0%
234	0.0%	0.0%	0.0%	0.0%	0.0%
235	0.0%	0.0%	0.0%	0.0%	0.0%
236	0.0%	0.0%	0.0%	0.0%	0.0%
237	0.0%	0.0%	0.0%	0.0%	0.0%

# Pliki *output/queue*



Znaleziono  
błędy: 0

—□— Algorithm1 —○— Algorithm2 —\*— Algorithm3 —▲— Algorithm4  
—◆— Algorithm5 —●— Algorithm6 —■— Algorithm7 —+— Algorithm8

# Fuzzing równoległy

## Tryb master/master

Każdy proces (rdzeń) wykonywał zarówno deterministyczne jak i niedeterministyczne mutacje

## 1-12x szybciej

Przy dużych plikach wąskim gardłem był dostęp do pamięci i nie zawsze było 12x szybciej przy 12 rdzeniach

## Zwiększyłem granularność

Deterministyczne strategie są wykonywane w całości. Podzieliłem operacje na części, żeby mogły być wykonywana równoległe

## Bez rozpraszania

Miałem do dyspozycji tyle maszyn co algorytmów, więc nie było sensu rozpraszać

## Znalazłem problem ...

Synchronizacja w trybie master/master powoduje, że czasami deterministyczne mutacje dla niektórych plików nie są aplikowane

## ... to go naprawiłem

Dodałem dodatkową synchronizację, która była wymagana tylko przy znalezieniu nowego pliku, ale działa tylko lokalnie (bez rozpraszania)





*Rock-solid reliability. It's hard to compete with brute force if your approach is brittle and fails unexpectedly. Automated testing is attractive because it's simple to use and scalable; anything that goes against these principles is an unwelcome trade-off and means that your tool will be used less often and with less consistent results.*

Michał Zalewski, AFL Historical notes

# Zmiany w mutacjach



## Słownik

AFL pozwala zdefiniować listę ciągów, które powinny się pojawić podczas fuzzingu. Napisałem taką listę.

## Niedopatrzenie w AFL

Implementacja AFL nie doklejała słów ze słownika na koniec pliku. Zgłosiłem problem i został naprawiony od wersji 2.40b.

## Pominięcie dużych, mniej ważnych plików

Deterministyczny fuzzing potrafił trwać godzinami dla dużych plików (>50kB).

Dodałem heurystykę omijającą takie pliki, o ile nie są "ważne".

# Zmiany w testowanej aplikacji

## Zmiany w konfiguracji

W szczególności przestawienie stałych odpowiedzialnych za przetwarzanie dużych plików na małe wartości

## Wyłączenie sum kontrolnych

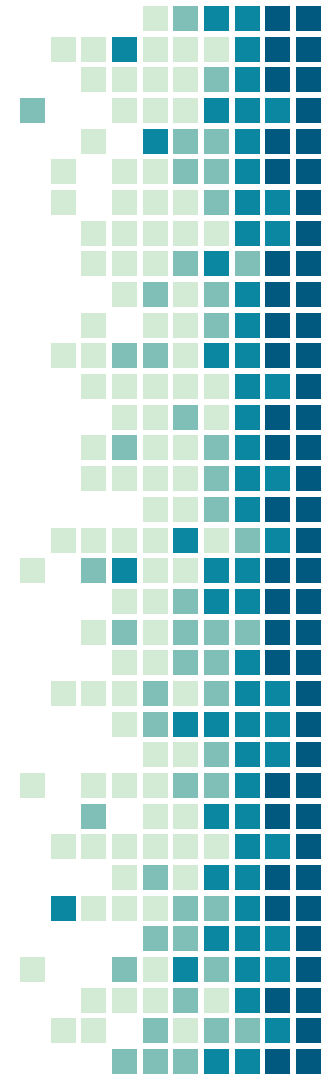
Nie było szans, żeby wylosować 64-bitową sumę kontrolną

## Dostosowanie frameworku testowego

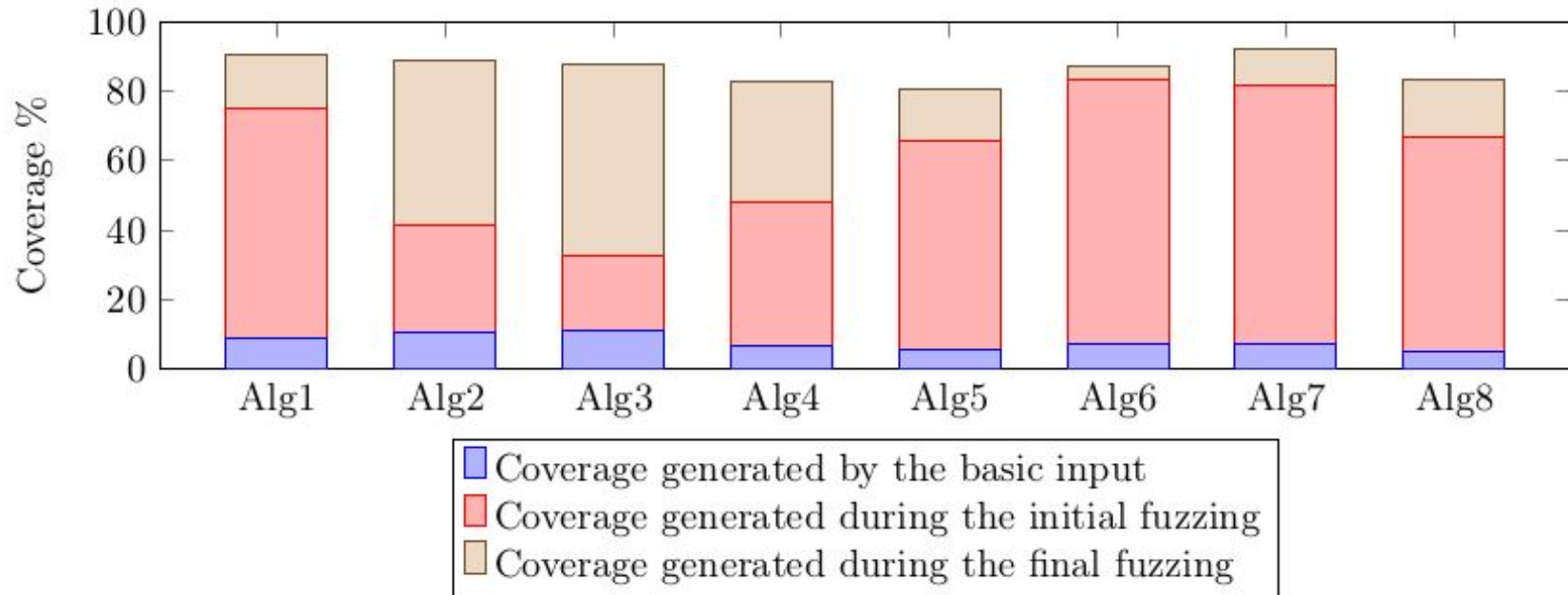
W szczególności napisanie programu konwertującego plik wygenerowany przez AFL na dwa pliki wymagane przez framework

## Włączenie debug asercji

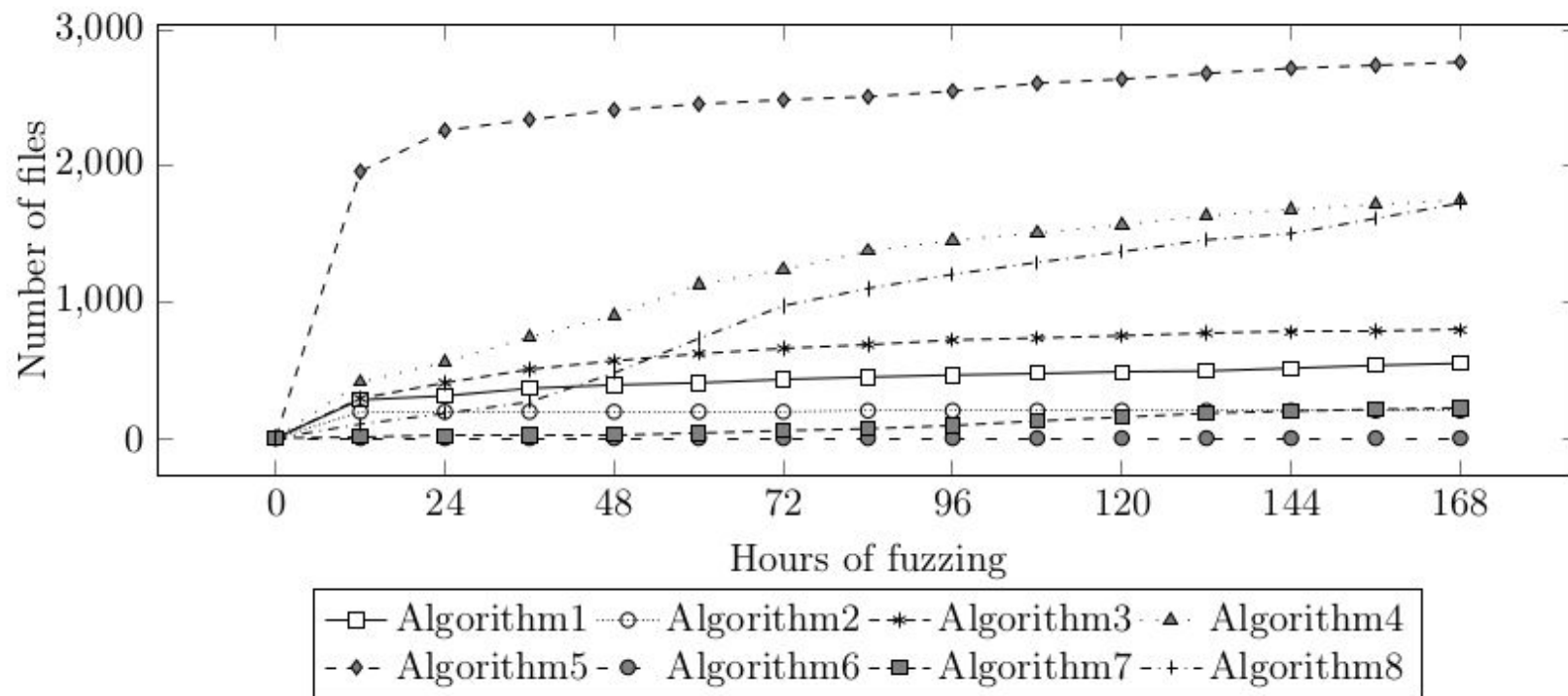
Zwiększyło szanse na wykrycie błędu przez asercje



# Rezultaty: pokrycie kodu

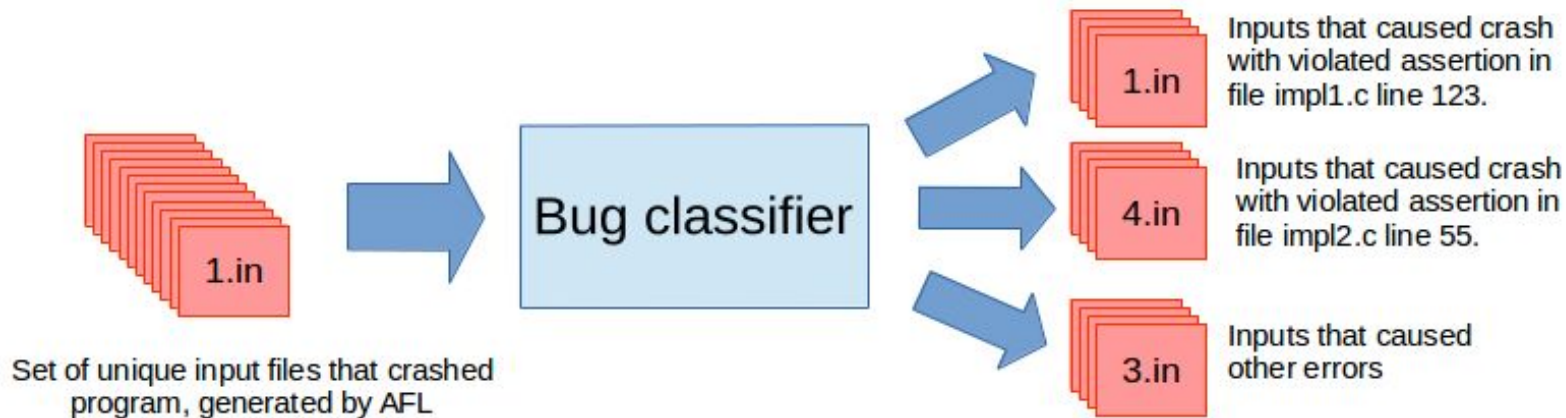


# Rezultaty: znalezienie błędów





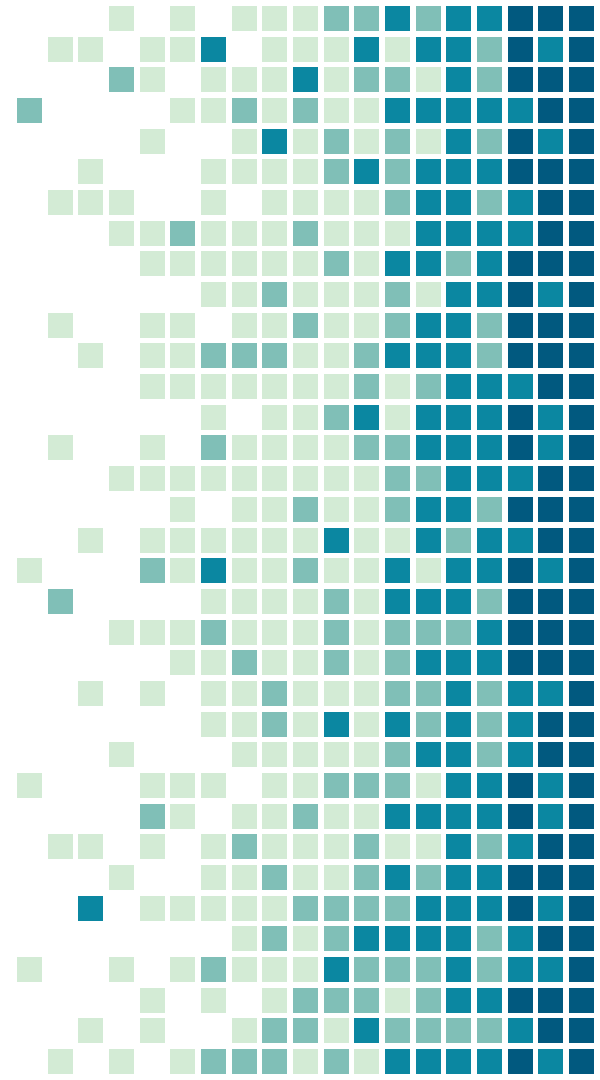
# Rezultaty: znalezienie błędy



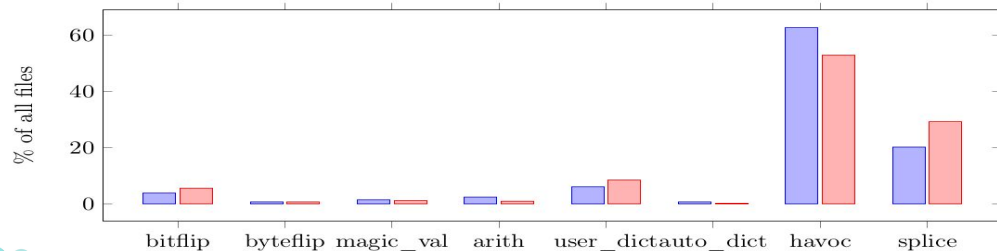
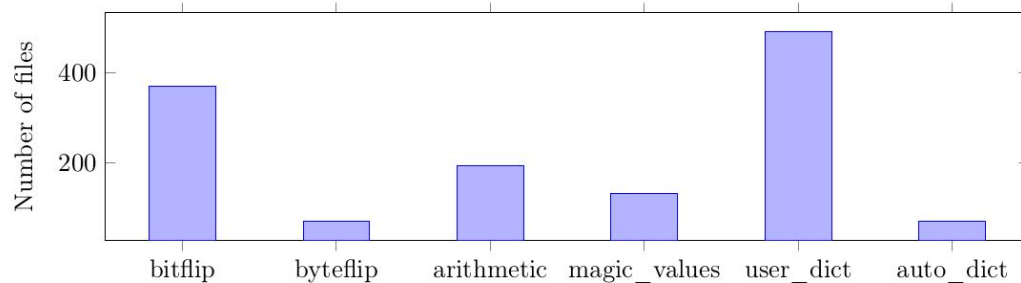
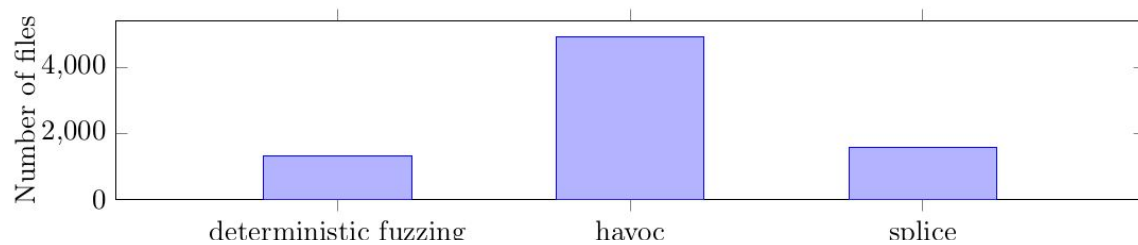


# Znalezione błędy: 2

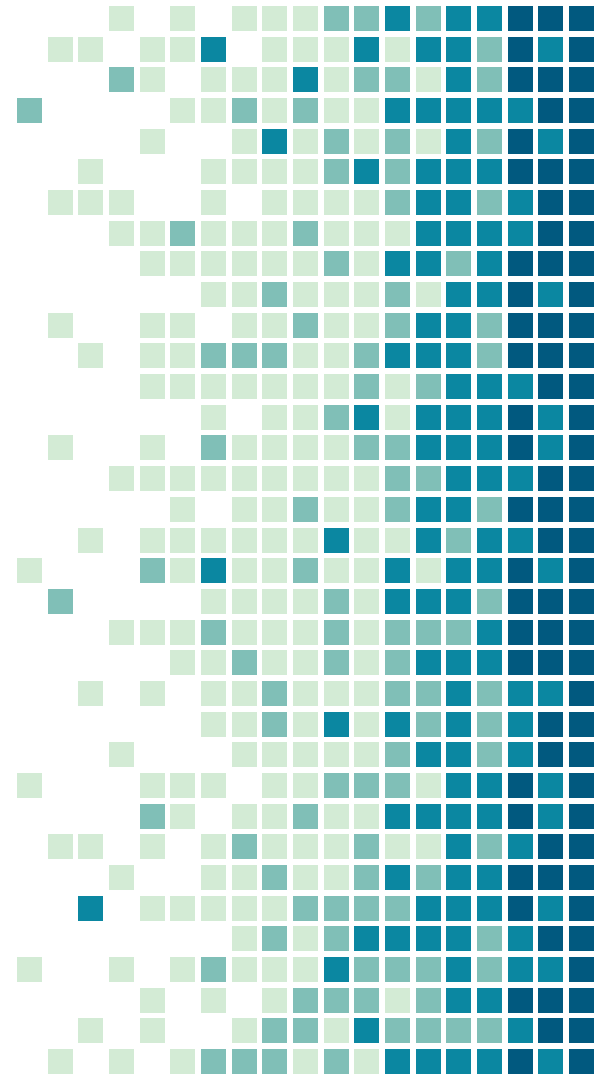
+trochę za mocnych debug asercji



# Które strategie działają?



Experiment unique input files Experiment unique crash files



# Dodatkowe materiały o fuzzingu

## Gorąco polecam

- <https://github.com/secfigo/Awesome-Fuzzing>
  - Zbiór linków, praktycznie wszystko czego można potrzebować
- <http://lcamtuf.coredump.cx/afl/>
  - Większość detali AFL jest opisana razem z przemyśleniami M. Zalewskiego
- CppCon 2017: Kostya Serebryany "Fuzz or lose..."
  - Świetna prezentacja, całkowicie inna niż ta (dotyczy libFuzzera i OSS-Fuzz)
- <http://9livesdata.com/fuzzing-how-to-find-bugs-automagically-using-afl/>
  - Skrót tego o czym dziś mówiłem w postaci artykułu

## Nie polecam tych książek

- *Fuzzing: Brute Force Vulnerability Discovery*
- *Fuzzing for Software Security Testing and Quality Assurance*
- *Open source fuzzing tools*
  - Są stare (2007-2008)
  - Uważam, że niewiele z nich wyciągnąłem

# Dziękuję za uwagę!

## Czas na pytania!

Kontakt:

 [jackowski@9livesdata.com](mailto:jackowski@9livesdata.com)

 [linkedin.com/in/ajackowski](https://www.linkedin.com/in/ajackowski)

