

Fuzzing

Motivasyon

1990 yılının karanlık ve fırtınalı bir gecesinde Bart Miller (Wisc. Üniversitesi) çevirmeli bir ağ üzerinden çalışmak için oturum açtı.

Fırtınadan dolayı çok fazla gürültü bulunuyordu Bu gürültü nedeniyle uygulamaları sürekli çöküyordu

Bu durum ona bu fikri verdi...



Motivasyon



Fikir: “fuzz” değerlerini ver – İşletim sistemi ve yardımcı programlara saf rastgele değerler ver, ör., `/dev/random`’den gelen gürültü değerleri

Bozulmasını izle!

Hizmetlerin %25-33’ünün çökmesine neden olur

Hatalardan bazıları, yaygın güvenlik açıklarından yararlananlarla aynıdır (özellikle arabellek aşımaları)

Fuzzing

Yazılım geliştiricileri hata yapar

Hatalar -> buglar -> güvenlik zafiyetleri

Fuzzing, bir programa hatalı biçimlendirilmiş/rastgele girdi göndermeyi ve onu çökertmeyi veya bir güvenlik açığı bulmayı ummayı amaçlamaktadır.

Programları korumak için hataları bulmakta kullanışlıdır

Ama aynı zamanda sistemlere sızmak için hataları bulmak için de

Apple, Firefox gibi birçok tanınmış şirkette/üründe (güvenlik) sorunları bulmak için kullanılır

User Testing vs Fuzzing

Kötü şeylerin olmasını bekleyerek programı birden fazla **normal** girdi ile çalıştırın

Amaç: **Normal** kullanıcıların hatalarla karşılaşmasını önlemek

Kötü şeylerin olmasını bekleyerek programı birden fazla **anormal** giriş ile çalıştırın

Hedef: **Saldırganların istismar** edilebilir hatalarla karşılaşmasını önlemek

Excel File Edit View Insert Format Tools Data Window Help

Home Layout Tables Charts SmartArt Formulas

Insert Chart

Column Line Pie Bar Area Scatter Other Line

A12

1 2 3 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

lasdjflfajsd

Save the changes you made to

Cancel Save

Chart Styles

Run Saved Query...
New Database Query...
Import Text File...
Import from FileMaker Pro...
Import from FileMaker Server...
Edit Query...
Data Range Properties
Parameters...

Sort...
Filter
Clear Filters
Advanced Filter...
Form...
Subtotals...
Validation...
Data Table...
Text to Columns...
Consolidate...
Group and Outline
PivotTable...
Table Tools
Get External Data
Refresh Data

Sheet1

Ready

Sum = 0

Fuzzing

Otomatik olarak **rastgele** (hatalı biçimlendirilmiş) test senaryoları oluşturun

Uygulamayı hatalara karşı **izleyin**

Girdilerin doğası farklı olabilir

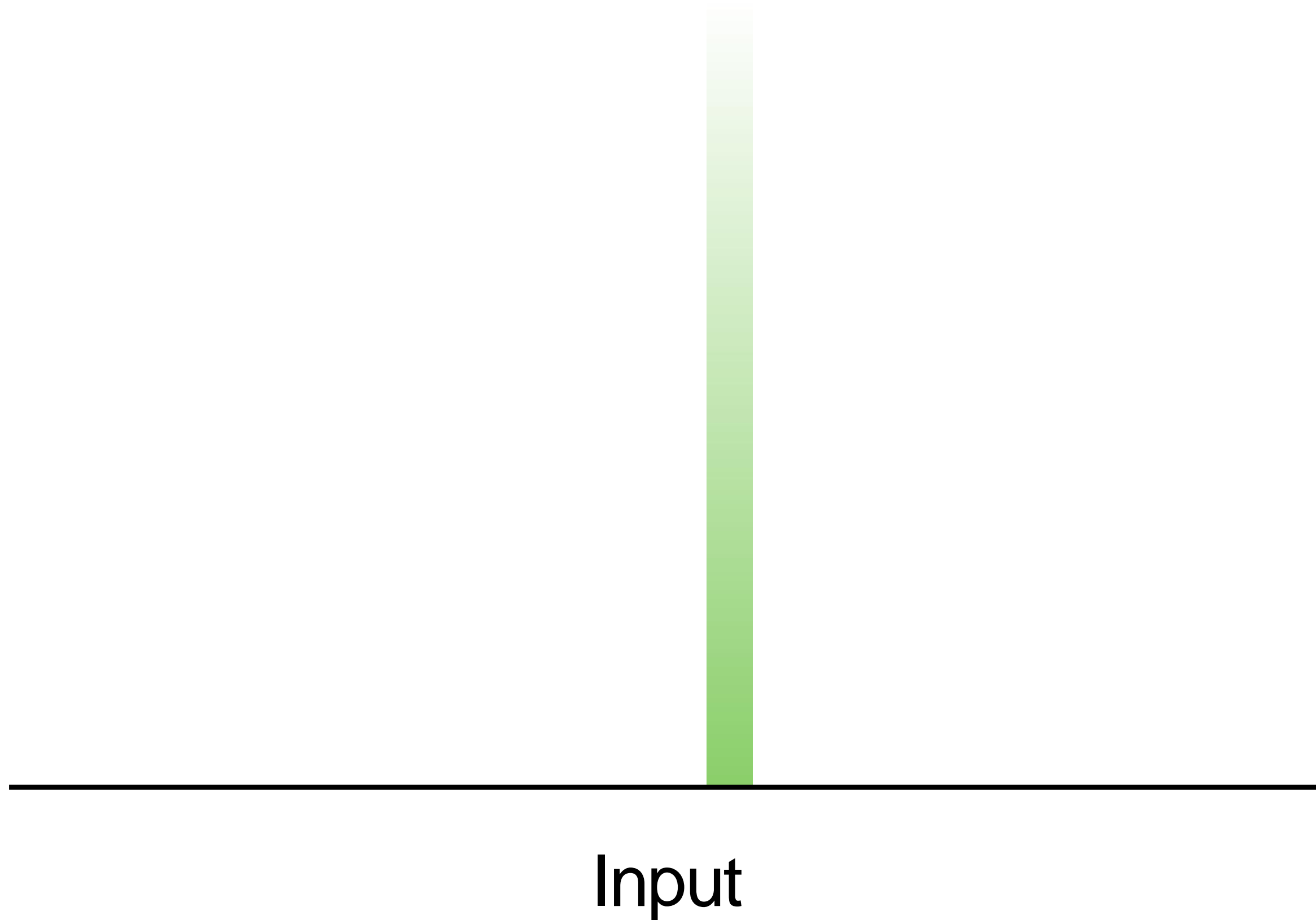
Dosyalar: .pdf, .png, .wav, .mpg, .json, etc

Network based: http, SOAP, SNMP

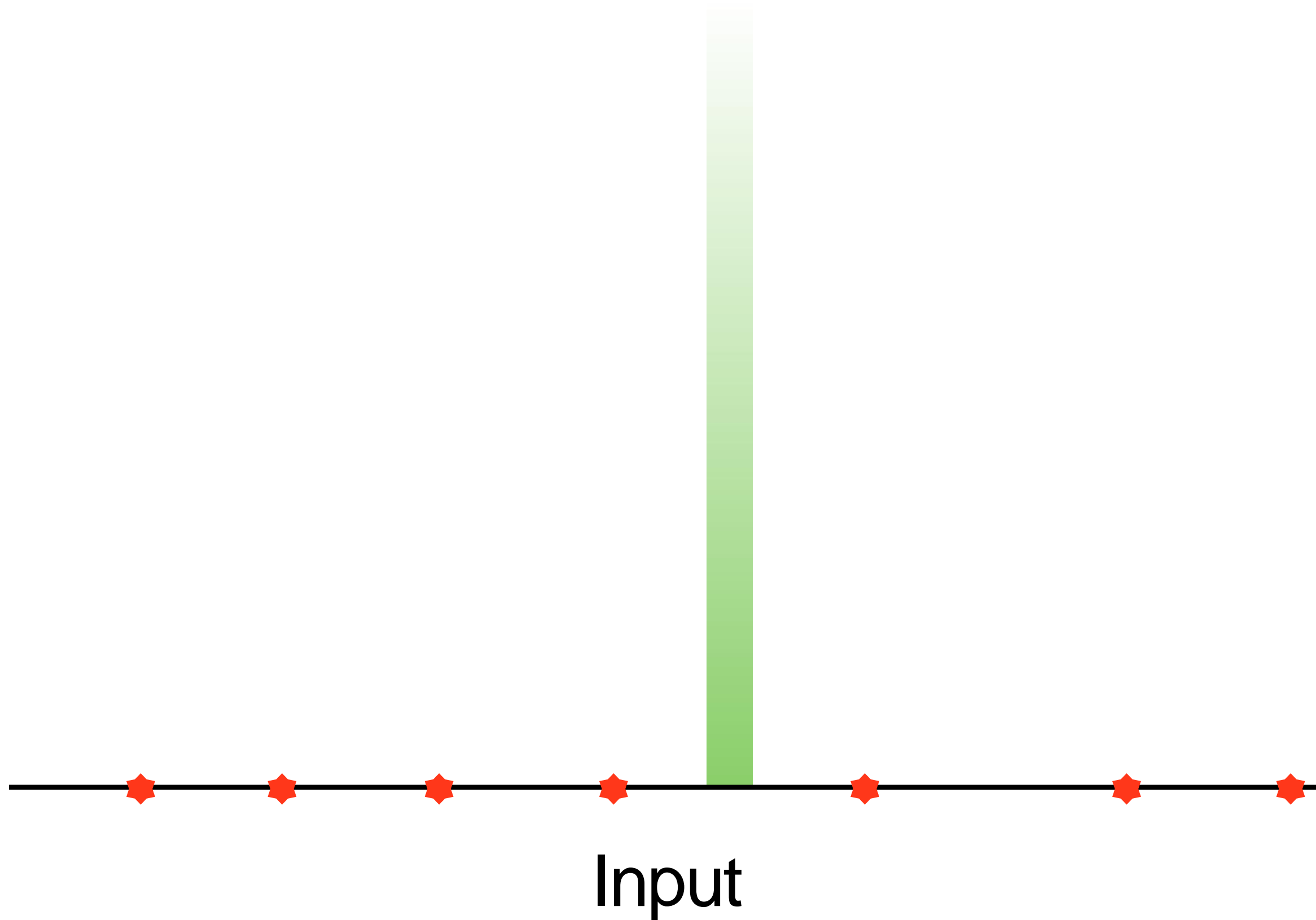
Hata Bulma Olasılığı?

Input

Hata Bulma Olasılığı?



Hata Bulma Olasılığı?



Fuzzing Nasıl Yapılır?

Çok uzun girdiler, çok kısa girdiler, boş girdiler, ...

Tam sayıların minimum/maksimum değerleri, sıfır ve negatif değerler

Hataları tetiklemesi muhtemel özel değerler, karakterler veya anahtar kelimeler; örneğin boş değerler, yeni satırlar veya dosya sonu karakterleri

String ifade formatlayıcısı %s %x %n

Noktalı virgül, eğik çizgi ve ters eğik çizgi, tırnak işaretleri, yazdırılamayan karakterler,...

Uygulamaya özel anahtar kelimeler, örneğin halt, return, DROP TABLES,...

Şablon Tabanlı Fuzzing

Mutasyona dayalı olarak da bilinir

Karmaşık girdi gerektiren programlar rastgele girdiyle düzgün bir şekilde test edilemez

Ör: JavaScript yorumlayıcısının test girdileri olarak geçerli JavaScript programlarına ihtiyacı var

Rastgele girdiler yerine mevcut, geçerli girdileri **değiştirin (mutate)**

Mutasyonlar rastgele olabilir veya sezgisel yöntemler tarafından yönlendirilebilir

Kapsam Tabanlı Fuzzing

Fuzzer örneğin yeni bir dalı kapsayan geçerli, yararlı bir girdi oluşturursa bunu daha sonraki mutasyonlar için bir başlangıç noktası olarak kullanın.

White-box fuzzing olarak da bilinir

Grammar Tabanlı Test

Karmaşık **metin** girdileri nasıl oluşturulur?

Klasik uygulama: Derleyicilerin test edilmesi

Günümüzde: Web uygulamaları, yorumlayıcılar, ayrıştırıcılar, ...XML, JSON vb. kullanan her şey.

Microsoft'un **SAGE** aracının (Godefroid ve diğerleri, 2008a) şirketin milyonlarca dolar değerindeki yazılım hatalarından kurtardığı bildiriliyor.

Context-free Grammars

Finite set of *terminals*

Finite set of *non-terminals*

Finite set of *rules*

Each *rule*: *Non-terminal* \rightarrow List of *terminals* and *non-terminals*

Starting rule

Örnek Grammar

expr . expr op expr

expr . (expr)

expr . - expr

expr . id

op . +

op . -

op . *

op . /

op . ↑

Türetme

Üretim yeniden yazma kuralı olarak yorumlanabilir

Sol taraftaki terminal olmayan, sağ taraftaki dize ile değiştirilir.

$E \rightarrow -E \rightarrow -(E) \rightarrow -(id)$

Değiştirme serisi = türetme

Her türetme adımında iki seçenek bulunur:

- Hangi terminal olmayanın değiştirileceği
- Değiştirilen terminal olmayan için hangi alternatif kullanılacak

Grammar Tabanlı Test

Test üretme \rightarrow grammar'e göre yeniden yazma

Başlangıç sembolüyle başla

Sağdaki terminal olmayan bir kuralı, soldaki terminal olmayan kuralla değiştirin

Yalnızca terminaller kalana kadar tekrarlayın

Sonuç test girdisidir

expr . expr op expr

expr . (expr)

expr . - expr

expr . id

op . +

op . -

op . *

op . /

op . ↑

$E \rightarrow - E \rightarrow - (E) \rightarrow - (E + E) \rightarrow - (E + id) \rightarrow - (id + id)$

Problemler

Context-free grammars → yineleme

Yineleme → **Sonsuz** sayıda olası girdi

Sonraki kuralla değiştirilecek kural hangisi?

Grammar Annotation

Yineleme derinliğini sınırla

Tekrar sayısını sınırla

Parsa ağacı derinliğini sınırla

```
{count 3} zeros;  
      zeros ::= '0';  
      zeros ::= '0' zeros;
```

Kombinasyonel Problem

Ör, VoIP software testi

Caller, VoIP server, client

CallerOS: Windows, Mac

ServerOS: Linux, Sun, Windows

CalleeOS: Windows, Mac

Grammar ile Aynı Problem

Grammerleri test etmek için covering array kullanmak mümkündür

```
Call ::= CallerOS ServerOS CalleeOS;  
CallerOS ::= 'Mac';  
CallerOS ::= 'Win';  
ServerOS ::= 'Lin';  
ServerOS ::= 'Sun';  
ServerOS ::= 'Win';  
CalleeOS ::= 'Mac';  
CalleeOS ::= 'Win';
```

Araçlar

LibFuzzer (<https://llvm.org/docs/LibFuzzer.html>),

AFL (<https://github.com/google/AFL>)

AFL++ (<https://github.com/AFLplusplus>)

Spike (<https://www.kali.org/tools/spike/>)

Peach (<https://wiki.mozilla.org/Security/Fuzzing/Peach>)

OneFuzz (<https://github.com/microsoft/onefuzz>)

Honggfuzz (<https://github.com/google/honggfuzz>)

AFL (American Fuzzy Lop)++

Geri bildirime dayalı fuzzer

Dosyayı değiştir – Kapsamı artırıyor mu? Sakla; yoksa sil

american fuzzy lop 0.47b (readpng)		
process timing		overall results
run time : 0 days, 0 hrs, 4 min, 43 sec		cycles done : 0
last new path : 0 days, 0 hrs, 0 min, 26 sec		total paths : 195
last uniq crash : none seen yet		uniq crashes : 0
last uniq hang : 0 days, 0 hrs, 1 min, 51 sec		uniq hangs : 1
cycle progress	map coverage	
now processing : 38 (19.49%)	map density : 1217 (7.43%)	
paths timed out : 0 (0.00%)	count coverage : 2.55 bits/tuple	
stage progress	findings in depth	
now trying : interest 32/8	avored paths : 128 (65.64%)	
stage execs : 0/9990 (0.00%)	new edges on : 85 (43.59%)	
total execs : 654k	total crashes : 0 (0 unique)	
exec speed : 2306/sec	total hangs : 1 (1 unique)	
fuzzing strategy yields	path geometry	
bit flips : 88/14.4k, 6/14.4k, 6/14.4k	levels : 3	
byte flips : 0/1804, 0/1786, 1/1750	pending : 178	
arithmetics : 31/126k, 3/45.6k, 1/17.8k	pend fav : 114	
known ints : 1/15.8k, 4/65.8k, 6/78.2k	imported : 0	
havoc : 34/254k, 0/0	variable : 0	
trim : 2876 B/931 (61.45% gain)	latent : 0	

american fuzzy lop ++2.65d (libpng_harness) [explore] {0}		
process timing		overall results
run time : 0 days, 0 hrs, 0 min, 43 sec		cycles done : 15
last new path : 0 days, 0 hrs, 0 min, 1 sec		total paths : 703
last uniq crash : none seen yet		uniq crashes : 0
last uniq hang : none seen yet		uniq hangs : 0
cycle progress	map coverage	
now processing : 261*1 (37.1%)	map density : 5.78% / 13.98%	
paths timed out : 0 (0.00%)	count coverage : 3.30 bits/tuple	
stage progress	findings in depth	
now trying : splice 14	avored paths : 114 (16.22%)	
stage execs : 31/32 (96.88%)	new edges on : 167 (23.76%)	
total execs : 2.55M	total crashes : 0 (0 unique)	
exec speed : 61.2k/sec	total tmouts : 0 (0 unique)	
fuzzing strategy yields	path geometry	
bit flips : n/a, n/a, n/a	levels : 11	
byte flips : n/a, n/a, n/a	pending : 121	
arithmetics : n/a, n/a, n/a	pend fav : 0	
known ints : n/a, n/a, n/a	own finds : 699	
dictionary : n/a, n/a, n/a	imported : n/a	
havoc/splice : 506/1.05M, 193/1.44M	stability : 99.88%	
py/custom : 0/0, 0/0		
trim : 19.25%/53.2k, n/a		
[cpu000: 12%]		

Özet

Fuzzing gibi basit test teknikleri çok etkili olabilir

Uygulanması nispeten kolay

Maliyetli olabilir – ne zaman durulacak? – ve kapsam açısından değerlendirmek zor

Pek çok araç var örneğin:

Muhteşem kaynak: The Fuzzing Book (<https://www.fuzzingbook.org/>)