

**Sigurnost operacijskih sustava i aplikacija**

# **Dinamička analiza koda: Fuzzing**

Ante Čavar, 23.05.2025

# Pregled predavanja

- Pitanja za ispite
- Motivacija
- Uvod u *fuzzing*
- Vrste *fuzzinga*
- *Fuzzing* alati
- Primjeri *fuzzinga*
- Izazovi *fuzzinga*
- Zaključak
- Literatura

# Pitanja za ispite

- Objasnite razliku između black-box, white-box i grey-box *fuzzing* pristupa.
- Navedite i opišite najmanje tri vrste tehnika generiranja ulaznih podataka u *fuzzingu*.
- Navedite barem 2 uspješna primjera korištenja *fuzzinga* u industriji te ih ukratko opišite
- Opišite glavne izazove moderne *fuzzing* metodologije i kako se adresiraju.
- Navedite najmanje tri popularna *fuzzing* alata te im opišite svrhu tj. domenu u kojoj se koriste

# Motivacija

- **Problem sigurnosti softvera**

- Sigurnosni propusti u softveru uzrokuju milijarde dolara štete godišnje
- Tradicionalno testiranje često propušta rubne slučajeve i neočekivane ulaze
- Ručna provjera koda je spora i podložna ljudskim greškama

- **Ograničenja postojećih pristupa**

- Statička analiza koda ne može otkriti sve vrste ranjivosti
- Ručno penetracijsko testiranje nije skalabilno
- Standardni testovi često pokrivaju samo očekivane putanje izvršavanja

# Motivacija

- **Primjeri skupih sigurnosnih propusta**
  - Equifax breach (2017): preko 147 milijuna korisnika, trošak >\$1.7 milijardi
  - Heartbleed (2014): ranjivost u OpenSSL-u koja je pogodila 2/3 web poslužitelja
  - Log4Shell (2021): kritična ranjivost u široko korištenom logging okviru
- **Financijski i reputacijski rizici**
  - Prekidi poslovanja
  - Gubitak povjerenja klijenata
  - Regulatorne kazne (GDPR, CCPA)

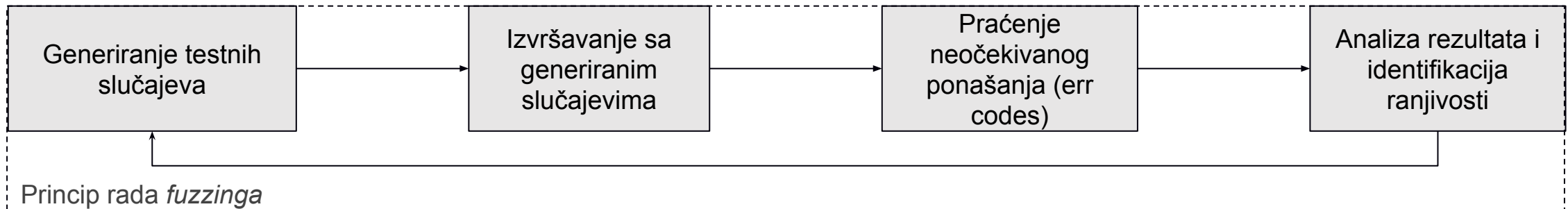
# Uvod u *fuzzing* - osnove

- **Definicija**

- (Automatizirana) tehnika testiranja koja šalje neočekivane ili nepravilne podatke programu
- Cilj je pronaći sigurnosne propuste i ranjivosti koji se ne otkrivaju standardnim testiranjem

- **Temeljna ideja**

- "Zbuniti" program nevaljanim, neočekivanim ili nasumičnim ulazima
- Izazvati rušenja, curenja memorije ili druga neispravna ponašanja
- hackeri već desetljećima koriste *fuzzing* za otkrivanje ranjivosti



# Uvod u fuzzing - povijest

- **Rane tehnike (1990-e)**
  - Jednostavno nasumično generiranje ulaznih podataka
  - Niska uspješnost i efikasnost
  - Ograničena primjena
- **Srednja faza (2000-e)**
  - Razvoj format-aware fuzzinga
  - Fuzzing kao dio sigurnosnih audita
  - Prve primjene u industriji
- **Moderni fuzzing (2010-e do danas)**
  - Coverage-guided fuzzing
  - Integracija s CI/CD pipeline-ima
  - Genetski algoritmi i evolucijski pristup
  - *Manès et al. (2019): "revolucionarni skok u učinkovitosti"*
- **Najnoviji trendovi**
  - Integrirani fuzzing u razvojne alate
  - Specijalizirani fuzzing za različite domene
  - Kombinacija s drugim sigurnosnim tehnikama

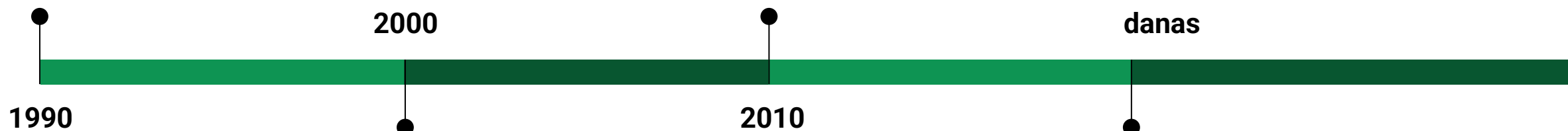
# Uvod u *fuzzing* - povijest

## Rane tehnike

- Jednostavno nasumično generiranje ulaznih podataka
- Niska uspješnost i efikasnost
- Ograničena primjena

## Moderni *fuzzing*

- Coverage-guided *fuzzing*
- Integracija s CI/CD pipeline-ima
- Genetski algoritmi i evolucijski pristup
- *Manès et al. (2019)*: "revolucionarni skok u učinkovitosti"



## Srednja faza

- Razvoj format-aware *fuzzinga*
- *Fuzzing* kao dio sigurnosnih audita
- Prve primjene u industriji

## Najnoviji trendovi

- Integrirani *fuzzing* u razvojne alate
- Specijalizirani *fuzzing* za različite domene
- Kombinacija s drugim sigurnosnim tehnikama



# Vrste *fuzzinga* - tehnike generiranja ulaza

- **Mutacijski *fuzzing***
  - Modificira postojeće validne ulazne podatke
  - Nasumične promjene bitova, bajtova ili blokova podataka
  - Efikasno za testiranje formata datoteka i protokola
  - najčešće korištena tehnika u praksi
- **Generativni *fuzzing***
  - Stvara ulazne podatke od početka prema specifikaciji
  - Zahtijeva model ili specifikaciju formata
  - Bolja pokrivenost kompleksnih formata podataka
- **Gramatički *fuzzing***
  - Koristi formalnu gramatiku za generiranje ulaza
  - Idealan za jezike i strukturirane protokole
  - Primjeri: testiranje parsera, interpretera, kompajlera
- **Hibridni pristupi**
  - Kombinacija različitih tehnika generiranja
  - Prilagodba specifičnim potrebama i ciljanim aplikacijama

# ***Vrste fuzzinga - Crna kutija***

- **Karakteristike**

- Nema pristupa izvornom kodu ili internim strukturama programa
- Tretira program kao "crnu kutiju"
- najjednostavniji ali i najmanje učinkovit pristup

- **Primjene**

- Testiranje vlastitih programa kad izvorni kod nije dostupan
- Penetracijska testiranja vanjskih sustava
- Implementiran u alatima kao što su Radamsa i OWASP ZAP

# ***Vrste fuzzinga - Crna kutija***

- **Prednosti**

- Jednostavna implementacija
- Nema potrebe za poznavanjem implementacije
- Može se primijeniti na bilo koji program
- jedini način testiranja softvera zatvorenog koda (*closed source*)

- **Nedostaci**

- Niska pokrivenost koda
- Neučinkovito pronalaženje dubokih grešaka
- Teško pronalazi ranjivosti koje zahtijevaju specifične ulaze

# Vrste *fuzzinga* - Bijela kutija

- **Karakteristike**

- Potpuni pristup izvornom kodu programa
- Koristi statičku analizu, simboličko izvršavanje i praćenje putanja
- omogućuje dubinsko testiranje kompleksnih uvjeta

- **Primjeri**

- SAGE (Microsoft)
- KLEE
- Mayhem

# Vrste *fuzzinga* - Bijela kutija

- **Prednosti**

- Visoka pokrivenost koda
- Efikasno pronalaženje kompleksnih ranjivosti
- Može ciljano testirati kritične dijelove koda
- Mogućnost zaobilaženja složenih uvjeta

- **Nedostaci**

- Skupo za implementaciju i održavanje
- Zahtijeva specijalizirane alate i znanje
- Problemi sa skalabilnošću kod velikih programa
- "Path explosion" problem

# Vrste *fuzzinga* - Siva kutija

- **Karakteristike**

- Djelomični pristup informacijama o strukturi programa (najčešće dobiveni reverzingom)
- Koristi instrumentaciju za praćenje pokrivenosti koda
- najpopularniji pristup u modernom *fuzzingu*

- **Primjeri alata**

- AFL (American Fuzzy Lop)
- libFuzzer
- honggfuzz

# Vrste *fuzzinga* - Siva kutija

- **Prednosti**

- Bolja pokrivenost koda od black-box pristupa
- Manje resursa od white-box pristupa
- Praktično primjenjiv na veće sustave
- Ravnoteža između efikasnosti i implementacijske složenosti

- **Tehnike**

- *Coverage-guided fuzzing*
- *Feedback-driven fuzzing*
- *Evolutionary fuzzing*

# Vrste *fuzzinga* - usporedba

- Usporedba prema efikasnosti pronalaska ranjivosti

Pristup	Pokrivenost koda	Zahtjevnost implementacije	Brzina izvođenja	Skalabilnost
Crna kutija	Niska	Niska	Visoka	Visoka
Bijela kutija	Vrlo visoka	Vrlo visoka	Niska	Niska
Siva kutija	Srednja	Srednja	Srednja-visoka	Srednja-visoka

- Odabir pristupa prema cilju

- Za nepoznate sustave bez pristupa kodu: crna kutija
- Za kritične komponente gdje je važna visoka pokrivenost: bijela kutija
- Za većinu modernih primjena u industriji: siva kutija



# *Fuzzing* alati - pregled

- **Evolucija *fuzzing* alata**
  - Rani alati: jednostavni generatori nasumičnih ulaza
  - Srednja generacija: format-aware fuzzeri
  - Moderna generacija: inteligentni, coverage-guided fuzzeri
- **Kategorizacija alata po primjeni**
  - Opći fuzzeri (za razne aplikacije)
  - Specijalizirani fuzzeri (za protokole, parsere, itd.)
  - In-process i out-of-process fuzzeri
  - Kontinuirani *fuzzing* sustavi (CI/CD integracija)

# *Fuzzing* alati - otvorenog koda

- **AFL (American Fuzzy Lop)**

- Najpopularniji grey-box fuzzer
- Koristi genetske algoritme za generiranje ulaza
- Instrumentation-guided *fuzzing*
- revolucionirao *fuzzing* tehnologiju
- AFL++ je moderne nadogradnja originalnog AFL-a

- **libFuzzer**

- In-process, coverage-guided fuzzer
- Integriran s LLVM kompajlerom
- Vrlo brz zbog izbjegavanja fork()
- Pogodan za jedinične testove

- **Honggfuzz**

- Podržava više hardverskih feedback kanala
- Efikasan na višejezgrenim sustavima
- Podržava hardware-based feedback (Intel PT)

- **Syzkaller:**

- Specijaliziran za testiranje jezgre OS-a
- Razvijen od Googlea
- Pronašao tisuće kritičnih ranjivosti u Linux kernelu

# Fuzzing alati - AFL

```
american fuzzy lop 1.86b (test)

process timing
  run time : 0 days, 0 hrs, 0 min, 2 sec
  last new path : none seen yet
  last uniq crash : 0 days, 0 hrs, 0 min, 2 sec
  last uniq hang : none seen yet

cycle progress
  now processing : 0 (0.00%)
  paths timed out : 0 (0.00%)

stage progress
  now trying : havoc
  stage execs : 1464/5000 (29.28%)
  total execs : 1697
  exec speed : 626.5/sec

fuzzing strategy yields
  bit flips : 0/16, 1/15, 0/13
  byte flips : 0/2, 0/1, 0/0
  arithmetics : 0/112, 0/25, 0/0
  known ints : 0/10, 0/28, 0/0
  dictionary : 0/0, 0/0, 0/0
  havoc : 0/0, 0/0
  trim : n/a, 0.00%

map coverage
  map density : 2 (0.00%)
  count coverage : 1.00 bits/tuple

findings in depth
  favored paths : 1 (100.00%)
  new edges on : 1 (100.00%)
  total crashes : 39 (1 unique)
  total hangs : 0 (0 unique)

path geometry
  levels : 1
  pending : 1
  pend fav : 1
  own finds : 0
  imported : n/a
  variable : 0

overall results
  cycles done : 0
  total paths : 1
  uniq crashes : 1
  uniq hangs : 0

[cpu: 92%]
```

Primjer fuzziranja programa koristeći AFL

Možemo primjetiti metrike poput:

- vrijeme od pokretanja
- vrijeme od zadnjeg pada sustava
- koliko putova je otkriveno
- koliko unikatnih padova je detektirano
- ukupno padova
- trenutno stanje (koji skup testova/podataka se vrti)

# *Fuzzing* alati - komercijalni i cloud

## • Komercijalni alati

### • Peach Fuzzer

- Profesionalni framework
- Podržava složene protokole i formate
- Koristi se u enterprise okruženjima

### • Defensics (Synopsys):

- Specijaliziran za testiranje mrežnih protokola
- Ekstenzivna biblioteka gotovih testnih slučajeva
- Fokus na sigurnost industrijskih sustava

## • Cloud *fuzzing* platforme

### • Google OSS-Fuzz

- Kontinuirani *fuzzing* projekt za open-source projekte
- otkrio preko 16,000 bugova u brojnim projektima
- Besplatno dostupan za open-source projekte

### • Google ClusterFuzz

- Distribuirana infrastruktura za skalabilni *fuzzing*
- Automatizira cijeli proces pronalaska, reprodukcije i praćenja bugova

### • Microsoft Security Risk Detection:

- Cloud-bazirani white-box *fuzzing* servis
- Kombinira simboličko izvršavanje i *fuzzing*

# Primjeri *fuzzinga* - uspješne primjene

- **Google Project Zero**

- Elite tim sigurnosnih istraživača
- Koriste *fuzzing* za pronalazak zero-day ranjivosti
- Otkrili tisuće sigurnosnih propusta u kritičnom softveru
- jedan od najuspješnijih primjera primjene *fuzzinga*

- **Microsoft Security Development Lifecycle (SDL)**

- *Fuzzing* kao obavezni dio razvoja softvera
- Implementirano za sve Microsoft proizvode
- Značajno smanjenje sigurnosnih incidenata

# Primjeri *fuzzinga* - uspješne primjene

- **Apple Security Bounty program**

- Nagrade za pronalazak ranjivosti
- Fokus na *fuzzing* testiranje
- Poboljšava sigurnost iOS i macOS platformi

- **Uspješni open-source projekti**

- Chrome Browser - ClusterFuzz
- OpenSSL - nakon Heartbleed-a
- Linux kernel - Syzkaller
- Firefox - ContinuousFuzzing program

# Izazovi *fuzzinga* - ograničenja pristupa

- **Logička ograničenja**

- Teško otkrivanje složenih logičkih grešaka
- Nemogućnost verificiranja poslovne logike
- *fuzzing* nije prikladan za pronalazak semantičkih grešaka
- Ograničenja u validaciji ispravnosti funkcioniranja

- **Dubinske barijere**

- checksum provjere i kompleksni preduvjeti značajno otežavaju *fuzzing*
- Format-specifična ograničenja
- Magic bytes i složene strukture formata
- Višefazne autentikacije i autorizacije

# Izazovi *fuzzinga* - ograničenja pristupa

- **Strukturni izazovi**

- Nelinearni programski tokovi
- Velike aplikacije s kompleksnim arhitekturama
- Virtualizirani i skriveni slojevi
- Distribuirane aplikacije i mikroservisi

- **Izazovi praćenja**

- Teško praćenje pokrivenosti koda u nekim okruženjima
- JIT kompilacija i dinamički generirani kod
- Interpretirni jezici
- Hardverske komponente



# Izazovi *fuzzinga* - praktična primjena

- **Organizacijski izazovi**

- Integracija *fuzzinga* u postojeće razvojne procese
- Budžetiranje i resursi za sigurnosno testiranje
- Educiranje razvijača o *fuzzing* tehnikama
- Mjerenje povrata investicije (ROI) za *fuzzing*

- **Vremenski okvir**

- Balansiranje između vremena razvoja i vremena testiranja
- Određivanje optimalnog trajanja *fuzzing* kampanje
- Godefroid (2020) navodi "problem određivanja kada prestati s *fuzzingom*"
- Strategije za brzu identifikaciju visoko-vrijednih ranjivosti

# Izazovi *fuzzinga* - praktična primjena

- **Regulatorni aspekti**

- Usklađenost s industrijskim standardima
- Regulatorni zahtjevi za sigurnosno testiranje
- Dokumentiranje procesa za certifikaciju
- Pravna odgovornost za neotkrivene ranjivosti

- **Zakonska ograničenja**

- Etičke i pravne granice testiranja
- Testiranje sustava trećih strana
- Odgovornost i transparentnost u otkrivanju ranjivosti
- Usklađenost s GDPR-om i drugim propisima o privatnosti

# Zaključak

- Automatizirajući pronalazak ranjivosti, *fuzzing* uvelike pridonosi sigurnosti i stabilnosti programa kojima se svakodnevno služimo
- Još dugo godina će ostati jedna od dominantnih tehnika testiranja programa zbog automatizacije te pokrivenosti svih slučajeva
- Smatram da će u budućnosti *fuzzing* biti više integriran sa DevSecOps procesima te da će se uz razvoj kako strojnog učenja tako i LLM-ova *fuzzing* još više unaprijediti

# Literatura

- Chen, Chen, et al. "A systematic review of fuzzing techniques." *Computers & Security* 75 (2018): 118-137.
- Godefroid, Patrice. "Fuzzing: Hack, art, and science." *Communications of the ACM* 63.2 (2020): 70-76.
- Zhu, Xiaogang, et al. "Fuzzing: a survey for roadmap." *ACM Computing Surveys (CSUR)* 54.11s (2022): 1-36.
- Manès, Valentin J.M., et al. "The art, science, and engineering of fuzzing: A survey." *IEEE Transactions on Software Engineering* 47.11 (2019): 2312-2331.
- Liang, Hongliang, et al. "Fuzzing: State of the art." *IEEE Transactions on Reliability* 67.3 (2018): 1199-1218.

# Dodatna literatura

- Zeller, Andreas and Gopinath, Rahul and Böhme, Marcel and Fraser, Gordon and Holler, Christian (2019) The Fuzzing Book
  - alt: <https://www.fuzzingbook.org/>
- AFL++: <https://github.com/AFLplusplus/AFLplusplus>
- libFuzzer: <https://llvm.org/docs/LibFuzzer.html>

# Hvala!