Зборник радова факултета техничких наука, Нови Сад

UDK: 007:004.056
DOI: https://doi.org/10.24867/21BE23Sipovac

ФАЗ ТЕСТИРАЊЕ ВЕБ АПЛИКАЦИЈА

FUZZ TESTING OF WEB APPLICATIONS

Жељана Шиповац, *Факултет техничких наука, Нови Сад*

Област – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *Тема рада јесте примјена фаз тестирања на веб апликацијама. У раду су анализирани алати за фаз тестирање, као и рањиве апликације са веба коришћене за потребе извођења фаз тестова, и демонстрације рада алата. Урађени су фаз тестови над овим апликацијама и анализирани су резултати тестирања.*

Кључне речи: *фаз тестирање, тестирање, безбедност*

Abstract – *The topic of the paper is the application of fuzz testing on web applications. Tools for fuzz testing are analysed, as well as vulnerable applications from the web used for the purposes of performing fuzz tests, and demonstrations of ways to use fuzzing tools. Fuzz tests were performed on these applications, using fuzz testing tools, and the test results were analysed.*

Keywords: *fuzz testing, testing, security*

1. УВОД

Тестирање софтвера је метод провјеравања да ли стварни софтверски производ одговара очекиваним захтјевима и потврђивање да софтверски производ нема дефекте. То укључује извршавање софтверских/системских компоненти коришћењем ручних или аутоматизованих алата за процјену једног или више својстава од интереса.

Сврха тестирања софтвера јесте да се идентификују грешке, празнине или недостајући захтјеви у супротности са стварним захтјевима [1].

Постоје двије главне стратегије тестирања у тестирању софтвера: позитивно тестирање и негативно тестирање.

Позитивно тестирање утврђује да апликација ради како се очекује.

Негативно тестирање осигурава да апликација може елегантно да рукује неважећим уносом или неочекиваним понашањем корисника. Сврха негативног тестирања је да спријечи пад софтверске апликације због негативних улаза и да побољша квалитет и стабилност [2, 3].

Извођењем само позитивног тестирања можемо само да се увјеримо да наш систем ради у нормалним условима. Како би се направио систем који ради без грешака, морамо да будемо сигурни да може да се носи са неочекиваним условима [3].

Управо овим се бави негативно тестирање. Једна од врста негативног тестирања софтвера је и фаз тестирање које ће бити тема овог рада.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Горан Сладић, ред. проф.

2. ФАЗ ТЕСТИРАЊЕ

Тестирање је саставни дио процеса развоја софтвера. Мотиви тестирања софтвера су, између осталог, потврда да имплементација одговара спецификацији, потврда робусности и отклањање програмских грешака. Поред правилног функционисања софтвера, потребно је обезбиједити и одређени степен безбједности за кориснике и системе који користе софтверски производ. Отклањање оваквих сигурносних пропуста у софтверу кроз тестирање и током развоја софтвера стога није практично [4].

2.1. *Дефиниција фаз тестирања*

Фаз тестирање или *fuzzing* је аутоматизована метода тестирања софтвера која убризгава неважеће, деформисане или неочекиване улазе у систем да би се открили софтверски недостаци и рањивости. Алат за фаз тестирање убризгава ове улазе у систем, а затим прати изузетке као што су падови или цурење информација. Једноставније речено, фаз тестирање уводи неочекиване улазе у систем и посматра да ли систем има било какве негативне реакције на улазе које указују на недостатке или проблеме у безбједности, перформансама или квалитету [5].

2.2. Развој фаз тестирања

Термин "*fuzz*" првобитно је креирао професор Бартон Милер 80-их година. Професор Милер је био са даљине пријављен у *Unix* систем преко *dial-up* мрежне везе током олује, што је изазивало много сметњи на *dial-up* линку и узроковало пад апликација које су користиле податке ван линије [6].

Професор Милер је након тога додијелио пројекат својој класи на Универзитету у Висконсину, у ком су студенти требали да развију основни фазер који ради преко командне линије, за тестирање поузданости Unix програма бомбардовањем са насумичним подацима и праћењем евентуалних падова програма. Професор Милер и његови ученици су открили улазе који изазивају пад програма за више од четвртине најчешће коришћених *Unix* услужних програма које

су испитали [5, 7]. Оригинални *fuzzing* пројекат наставио је да даје доприносе 1995, 2000, 2006. и недавно 2020. године.

2.3. Предности фаз тестирања

Фаз тестирање нуди широк спектар предности у пољу безбједности и квалитета софтверског производа [7, 8, 9]:

- Фаз тестирање даје добру општу слику квалитета циљног система и софтвера. Користећи фаз тестирање, може се лако процијенити робусност и безбједносни ризик система и софтвера који се тестирају.
- Фаз тестирање је примарна техника коју злонамјерни хакери користе да пронађу рањивости софтвера.
- Фаз тестирање има ниске трошкове и за трошкове и за вријеме. Када се фазер покрене, може сам да почне да тражи грешке, без ручне/људске интервенције, и може да настави то да ради колико год је потребно.
- Фаз тестирање помаже у откривању грешака које не би биле откривене конвенционалним методама тестирања или ручним ревизијама.
- Сваки проналазак води до више проналазака.
- Фаз тестирање штити од неочекиваних ивичних случајева.
- Смањује трошкове развоја производа

3. АЛАТИ ЗА ФАЗ ТЕСТИРАЊЕ - ФАЗЕРИ

Фаз тестирање је, као што је већ наведено, метода аутоматског тестирања која убацује или инјектује невалидне, неочекиване, или малициозне податке у систем како би открила грешке и рањивости система. Алати који изводе фаз тестирање, или фазери инјектују ове податке у систем и након тога посматрају понашање система и прате да ли ће доћи до неких изузетака од нормалног рада система, на примјер заустављања рада или цурења информација. У овом поглављу, биће наведени фазери који су коришћени за фаз тестирање веб апликација.

3.1. *Owasp ZAP*

OWASP Zed Attack Proxy (ZAP) је један од најпопуларнијих бесплатних алата за тестирање безбиједности софтвера, и активно је одржаван од стране интернационалног тима волонтера. Користи се за аутоматско откривање безбиједносних рањивости у веб апликацијама. Такође може да се користи и за мануелно тестирање безбједности софтвера [10].

ZAP се одржава под окриљем *Open Web Application Security Project (OWASP). ZAP* је дизајниран посебно за тестирање веб апликација и истовремено је флексибилан и проширив.

У својој основи, *ZAP* је оно што је познато као „прокси човјек у средини“. Стоји између претраживача који користи тестер и веб апликације, тако да може пресрести и прегледати поруке послате између претраживача и веб апликације, измијенити садржај ако је потребно, а затим прослиједити те пакете на одредиште.

За област истраживања овог рада, и у нападима изведеним зарад испитивања безбједности веб апликација који ће бити објашњени у наредним поглављима, највише ће се користити „*fuzzer*“ опција коју нуди *OWASP ZAP*. Ова опција омогућава извршавање фаз тестова над пресретнутим захтјевима.

3.2. *FFUF*

FFUF, или “*Fuzz Faster you Fool*“ је алат за тестирање отвореног кода, који је намијењен откривању скривених елемената и садржаја веб апликација, или веб сервера [11]. Овај алат је намијењен брзом фаз тестирању, и написан је у *Go* језику. Ово је један од новијих, и до сада се показао као најбржи алат за фаз тестирање, отвореног кода.

FFUF је апликација вођена командном линијом, која се покреће у *Linux* терминалу или *Windows* командној линији, што значи да не садржи интерактивни GUI, већ се умјесто тога покреће помоћу команда унесених преко командне линије.

Неки од примјера коришћења овог алата су:

- Откривање фајлова
- Фаз тестирање *GET* захтјева
- Фаз тестирање *POST* захтјева

3.3. *Burp Suite*

Burp Suite је интегрисана платформа/графички алат за извођење безбједносног тестирања веб апликација. Његови различити алати раде заједно како би подржали цијели процес тестирања, од почетног мапирања и анализе површине напада апликације, до проналажења и искоришћавања безбједносних пропуста.
Алат је написан на Јави и развијен од стране *PortSwigger Web Security*. Поред основних функционалности, као што су прокси сервер, скенер и уљез, алат садржи и напредније опције као што су паук, репетитор, декодер, екстендер и секвенцер [12].
Битан дио алата, који је неопходан за извођење фаз тестирања је прокси за пресретање . *Burp Suite* садржи прокси за пресретање који омогућава кориснику да види и измијени садржај захтјева и одговора док су у транзиту [13].
Burp Suite долази са интегрисаним *HTML Fuzzer*-ом, познатијим као *Burp Intruder*. Овај дио *Burp Suite*–а нам даје могућност за fuzz тестирање са високим степеном прилагођавања. Након што пресретнемо захтјев, користећи прокси, Burp Suite нам нуди опцију слања тог захтјева ка Intruder-у.

4. ВЕБ АПЛИКАЦИЈЕ СА РАЊИВОСТИМА

У овом поглављу биће описано пар рањивих веб апликација које ће бити коришћене за демонстрирање начина рада алата за фаз тестирање, и над којима ће бити извођени фаз напади. Апликације које ће бити описане направљене су тако да посједују одређене рањивости, те служе програмерима и људима задуженим за безбиједност софтвера да побољшавају своје вјештине у домену безбиједности.

4.1. *OWASP Mutillidae II*

OWASP Mutillidae II је бесплатна веб апликација отвореног кода, намјерно направљена као рањива веб апликација. *Mutillidae* се може инсталирати на *Linux* и *Windows*. Садржи десетине рањивости и наговјештаја за помоћ кориснику. Ово је окружење намијењено за извођење сигурносних напада.
Mutillidae је коришћен у дипломским курсевима безбедности, корпоративним веб курсевима обуке и као циљ за процјену за софтвер за процјене рањивости [14].

4.2. *bWAPP*

bWAPP, или *buggy web application*, је бесплатна и намјерно направљена као рањива веб апликација отвореног кода. Помаже заљубљеницима у безбиједност, програмерима и студентима да открију и спријече пропусте на вебу.

bWAPP садржи преко 100 рањивости. Покрива све главне познате веб несигурности, укључујући све ризике из OWASP top 10 пројекта. *bWAPP* је намијењен само за тестирање безбедности веб апликација, и за образовне сврхе [15].

4.3. *DAMN VULNERABLE WEB APPLICATION (DVWA)*

Damn Vulnerable Web Application (DVWA) је *PHP/ MySQL* веб апликација која је циљано направљена рањива. Њен главни циљ јесте да буде помоћ професионалцима за безбједност да тестирају своје вјештине и алате у легалном окружењу, и да помогне веб програмерима да боље разумију процесе обезбјеђења веб апликација. Овај софтвер има и документоване и недокументоване рањивости с намјером да корисници покушају да открију што више проблема [16].

5. ИЗВОЂЕЊЕ ФАЗ ТЕСТИРАЊА

У раду је описано више изведених фаз тестова. Изведени су тестови са циљем откривања различитих врста рањивости: *SQL Injection*-а, *XSS Injection*-а, откривања осјетљивих информација, сривених фајлова, информација о корисницима...

Напади са циљем откривања *SQL Injection*-а су изведени над апликацијама *bWAPP* и *OWASP Mutillidae II*. Коришћена су три различита алата *OWASP ZAP*, *FFUF* и *Burp suite*. Ови тестови су имали доста сличан ток, креће се од пресретања захтјева коришћењем проксија, затим се за алате *OWASP ZAP* и *Burp suite* захтјев даље шаље ка дијелу алата задуженом за фаз тестирање, док се код алата *FFUF* користе команде командне линије. У сва три случаја потребно је обезбиједити улазне листе за фаз тестирање (у овом случају су то листе прилагођене за *SQL Injection*), и означити дијелове захтјева који ће бити мета фаз тестирања. Након тога могуће је покренути напад.

У свим тестирањима са циљем откривања *SQL Injection*-а детектована је рањивост на *SQL Injection,* а у неким је тест успио и да открије податке о разним табелама унутар базе података, као и податке о корисницима.

Изведени су и тестови са циљем откривања скривених фајлова. За ове тестове коришћен је *FFUF* алат који омогућава фаз тестирање у ову сврху. Тестови су изведени над апликацијама *bWAPP* и *DVWA*. За извршавање овог типа напада, прво је потребно открити ког типа су фајлови од којих је сачињена апликација. Да бисмо то открили, потребно је креирати фајл који садржи листу могућих екстензија. Потом се користи сљедећа команда, гдје након ознаке *–u* слиједи путања до почетног фајла апликације (*index*), а након ознаке *–w* слиједи путања до улазне листе могућих екстензија:
„ffuf –u http://localhost:8080/bwapp/indexFUZZ –w extension-discovery.txt“. Циљ команде је откривање екстензије улазног (*index*) фајла.
У оба случаја откривено је да је фајл типа „*.php*“.
За други дио напада, тражење конкретних назива фајлова, користи се SecLists листа намијењена за „*directory-discovery*“. Овај пут траже se сви фајлови чији називи се налазе унутар улазне листе, а затим прецизирамо да се траже фајлови са есктензијом „*.php*“. Формирана команда ће бити:
“ffuf -u http://localhost:8080/bwapp/FUZZ -w .../directory-discovery-seclists.txt -e .php”
Следећи корак је покретање команде и анализирање резултата.

У апликацији bWAPP је на овај начин успјешно откривен скривени фајл намијењен управо овој сврси. Откривени су и бројни други скривени фајлови (*secret.php*, *passwords*, *documents*...). У апликацији *DVWA* су такође откривени скривени фајлови. У оба случаја су као улазне листе коришћене листе са пројекта *SecLists* [17].

Следећи тестови који су извршени су били у циљу откривања осјетљивих информација. Први корак напада је било пресретање захтјева. Након тога се захтјев шаље дијелу алата за фаз тестирање. За први случај се тестирала форма за пријаву апликације *bWAPP*,и користио се тип напада „cluster bomb“ алата *Burp Suite*, који се односи на два улаза. Унешене су листе за фаз тестирање оба улаза (у нашем случају корисничко име и лозинку), и напад је покренут.

Други напад је изведен помоћу алата *OWASP ZAP* над апликацијом *Mutillidae*. Мета напада је био дио захтјева за пријаву корисника на систем, који се односи на лозинку, а као улазна листа коришћена је листа најчешћих лозинки са интернета. Први напад је успио да открије креденцијале корисника, док су другим, у апликацији *Mutillidae*, такође откривени креденцијали корисника, као и неки други подаци о њему, на примјер потпис (*eng. signature*).

Последњи у низу тестова који су изведени су тестови на *XSS* рањивост. Поново су коришћени алати *OWASP ZAP* и *Burp Suite*, а за улазне листе су коришћене листе прилагођене за *XSS* нападе. Први корак напада је био пресретање захтјева коришћењем проксија. Затим се захтјев шаље дијелу алата за фаз тестирање, означе се дијелови захтјева који су мета напада и обезбиједи путања до улазних листи. У апликацијама *Mutillidae* и *bWapp* је откривена ова рањивост, и извршен је послати малициозни *JavaScript* код.

Из резултата тестирања закључено је да је за откривање скривених фајлова и фолдера погодно користити алате вођене командном линијом, као што је *FFUF*, док смо за рањивости као што је *SQL injection* и *XSS injection* више повратних информација добили од алата као што су *OWASP ZAP* и *Burp suite*.

6. ЗАКЉУЧАК

У овом раду је описано фаз тестирање и његови бенефити када је ријеч о квалитету израде софтвера.

У раду су приказане теоријске основе фаз тестирања, као и његов развој кроз вријеме. Описани су и неки од алата који се користе за фаз тестирање, и рањиве апликације доступне на вебу које су од помоћи за вјежбање вјештина фаз тестирања и откривања рањивости.

Такође, у раду су описани начини за откривање неких од рањивости веб апликација помоћу фаз тестирања, као и начини генерисања улазних листи за поменуте тестове. Откривене су рањивости као што је *SQL Injection*, *XSS Injection*, цурење информација и откривање скривених фајлова.

7. ЛИТЕРАТУРА

[1] *"What is Software Testing? Definition", Thomas Hamilton*

https://www.guru99.com/software-testing-introduction-importance.html

[2] *"Negative Testing"*

https://smartbear.com/learn/automated-testing/negative-testing/

[3] "*What is Negative Testing? Test cases With Example"* , Thomas Hamilton

[4] *"DETEKCIJA SIGURNOSNIH PROPUSTA FAZ TESTIRANJEM",* Aleksandar Nikolić, Goran Sladić, Branko Milosavljević, Zora Konjović

https://infom.fon.bg.ac.rs/index.php/infom/article/view/1551/1523

[5] *"Fuzz testing", Synopsys*

https://www.synopsys.com/glossary/what-is-fuzz-testing.html

[6] *"Fuzzing info – the art of unexpected input engineering"*

https://fuzzinginfo.wordpress.com/history/

[7] *"Fuzz testing"*

https://fuchsia.dev/fuchsia-src/contribute/testing/fuzz_testing

[8] *"Code intelligence, What Is Fuzz Testing?"*

https://www.code-intelligence.com/what-is-fuzz-testing#FuzzTestingDefinition

[9] *"Fuzzing – what is it, and why bother?"*

https://cytal.co.uk/fuzzing-what-is-it-and-why-bother/)

[10] https://github.com/zaproxy/zaproxy

[11] *"Everything you need to know about FFUF"*

https://codingo.io/tools/ffuf/bounty/2020/09/17/everything-you-need-to-know-about-ffuf.html

[12] *"Web Security Testing with Burp Suite", Dr. Sunny Wear*

https://www.pluralsight.com/paths/web-security-testing-with-burp-suite

[13] *"What is Burp Suite?"*

https://www.geeksforgeeks.org/what-is-burp-suite/

[14] https://github.com/webpwnized/mutillidae

[15] "*bWapp*"

http://www.itsecgames.com

[16] https://github.com/digininja/DVWA

[17] https://github.com/danielmiessler/SecLists

Кратка биографија:

Жељана Шиповац рођена је 23.09.1997. године у Невесињу, БиХ. Завршила је основне академске студије 2020. године на Факултету техничких наука. Уписала је мастер студије исте године, студијски програм Електронско пословање.

контакт: sipovac.zeljana@gmail.com