Зборник радова Факултета техничких наука, Нови Сад

UDK: 007:004.056
DOI: https://doi.org/10.24867/20BE07Trajkovic

GRYPE СКЕНЕР ЗА ТЕСТИРАЊЕ РАЊИВОСТИ У КОНТЕЈНЕРИМА

GRYPE VULNERABILITY SCANNER FOR CONTAINERS

Анђела Трајковић, *Факултет техничких наука, Нови Сад*

Област– ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У овом раду је описана важност континуалног обезбеђивања безбедности у контејнерима и смањењу ризика од нежељених напада апликације. Акценат је дат на скенер Grype, који испитује рањивост у контејнерима.*

Кључне речи: *информациона безбедност, рањивост, контејнер, Grype*

Abstract – *In this paper is described the importance of continuously providing security in containers and reducing the risk of unwanted application attacks. Emphasis is placed on the Grype scanner, which examines vulnerabilities in containers.*

Keywords: *information security, vulnerability, container, Grype*

1. УВОД

Прелаз на аутоматизовани начин рада је у сталном расту, што проузрокује повећањем потребе да се изградња софтвера оријентише клијентима. Уједно корисничка повратна информација постаје главни критеријум квалитета софтвера. Један од могућих концепата за побољшање квалитета софтвера је *Development and Operations (DevOps)* [1], који се заснива на примени континуалне интеграције и континуалне испоруке софтвера.

Лакоћа коришћења софтвера и прилагођавање ширем аудиторијуму има за последицу повећања броја корисника. Повећање броја корисника апликације у једном тренутку може премашити хардверске могућности машине, на којој је покренута апликација, и захтевати скалирање. Скалирање може бити хоризонтално и вертикално. Због економских аспеката и тежње да се опслужи што већи број корисника, најоптималније решење постају удаљени сервери.

Процес коришћења удаљених сервера назива се виртуелизација, чија је примена отежана до појаве контејнера. Контејнери својом ефикасношћу доприносе лакшем преносу података и програма. Надмашују све методе развоја софтвера и отварају пут континуалног развоја и испоруке.

Контејнер је ефикасан у погледу преноса података, али је ефикасност безбедне имплементације софтвера контејнерима значајно опала.

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

Уграђивањем безбедности од самог почетка развоја софтвера смањује се простор за безбедоносне нападе. Касније, повезивање софтвера са алатима за скенирање, праћење безбедоносних аспеката и ажурирање истих омогућава лакше одржавање безбедности у апликацији.
У општем случају мете су софтвери који управљају осетљивим подацима. Осетљиви подаци су углавном фокусирани на нечији идентитет или новчане трансакције. Како би очували податке, приликом изградње софтвера неопходно је укључити и информациону безбедност [2].
Постоје разнолики алати, софтвери и организације које прате безбедносне пропусте и развој безбедносних механизама. Сходно томе у овом раду биће детаљно приказана најпопуларнија организација *OWASP* (*Open Web Application Security Project*) *TOP TEN* [3], рањивостима и скенер рањивости *Grype* [4].

2. ВИРТУЕЛИЗАЦИЈА

Данас се све више пословних активности аутоматизује, чиме се појављује и већи број осетљивих информација које је потребно ускладиштити. Порастом броја осетљивих информација потреба за сигурним рачунарским системима постаје све очигледнија [5]. Поред неопходне безбедности о којој ће нешто више бити касније, изазов је пронаћи место где се могу чувати огромне количине података. Имајући у виду да постоје хардверска ограничења за чување података и извршавање програма, као алтернатива се користе удаљени сервери.

Удаљени сервери су заправо удаљене физичке машине чије хардверске капацитете можемо дистрибуирати на више окружења или корисника. Односно, виртуелизација омогућава хостовање апликације услед недостатка ресураса.

Предност виртуелизације је у поједностављењу репликације и скалирању апликација. Постоје две врсте технологија које обезбеђују виртуелизацију сервера, то су виртуелизација на нивоу хардвера и виртуелизација на нивоу оперативних система.

Хардверски ниво визуелизације укључује покретање хипервизора који ствара и виртуелизује ресурсе сервера на више виртуелних машина. Сваки хардвер, виртуелна машина (ВМ), покреће сопствени оперативни систем и апликације. Насупрот томе, виртуелизација оперативног система виртуелизује ресурсе само на нивоу оперативног система; енкапсулира стандардне процесе оперативног система и њихове зависности за креирање „контејнера", којима заједнички управља основно језгро ОС-а.

Најпознатији примери виртуелизације хардвера су Xen, *KVM* и *VMware ESXi*. Код виртуелизације оперативног система су *Linux* (*LXC*), *Docker*, *BSD Jails* і *Solaris Zones*.
Виртуелизација хардвера била је доминантна технологија за примену, паковање и управљање апликацијама све до појаве контејнера, који представљају виртуелизацију оперативног система. Контејнери стичу све већу популарност појавом *Docker*-а, поред тога омогућавају и виртуелизацију на ниским трошковима и побољшаним перформансама у поређењу са виртуелним машинама [6]. Међутим, *Docker* није једино окружење за покретање контејнера, међу популарним се појављују и *Rkt* и *Containerd*.

2.1 Виртуелне машине

Виртуелна машина (ВМ) симулира оперативни систем користећи хипервизор. Хипервизор може да ради као софтвер на физичком хост оперативном систему или као фирмвер на машини без оперативног система. ВМ ради као виртуелни гостујући оперативни систем на хипервизору. Хипервизор омогућује логичку поделу физичке машине на више мањих ВМ, од којих свака има одговарајућу намену. Свака мања ВМ сада садржи наменски гостујући ОС, своје библиотеке и зависности. Иако је то велики помак ка изолацији података на машини, ипак је потребно изоловати мању количина података. Због велике количине ресурса перформансе извршавања и скалирања опадају, као решење се појављују контејнери [7].

2.2 Контејнери

Највећи изазов софтверске индустрије био је изоловање и управљање апликацијама независних од спољашњег окружења и платформе на којој се извршавају.
Проблем изолације велике количине података и хардверске зависности, 2007. године, решиле су компаније *Google*, *IBM*, *OpenVZ* и SGI. Сарађивали су са циљем да направе виртуелно окружење унутар *Linux* кернела без коришћења гостујућег ОС-а, контејнере.
Софтвер у свом животном циклусу мења више окружења или оперативних система, између којих може постојати разлика у конфигурацији. Један такав пример јесте рад у развојном, тестном и продукционом окружењу.
Поред тога, када више апликација раде на истој машини могу користити рачунарске ресурсе из друге апликације, и то ће у већини случајева произвести проблем изолације. У раду са ВМ-ма неопходно је у сваком кораку имплементације реконфигурисати и препаковати апликацију, што је напорно и подложно грешкама.
Контејнери у софтверској индустрији решавају ове проблеме обезбеђујући изолацију између апликација и управљање рачунарским ресурсима. Такође, решавају и међусобне зависности, изолујући извршавање апликације са зависностима.

2.2.1 *Docker* окружење и *Docker* слике

Сви контејнери се заснивају на *Linux*-у. *Linux* због својих добрих перформанси извршавања, поузданости, широке употребе и бесплатног коришћења представља један од најидеалнијих ОС-а за развој *cloud*-а. Најпознатији *Linux* контејнер је *Docker*.
Docker слику одређују пакети, изворни код, зависности и библиотеке које користи. Састоји се од низа корака које ОС извршава како би покренуо апликацију. Кораци су међузависни и заправо представљају постепено упутство за изградњу софтвера од најосновније до најкомплексније конфигурације.

2.2.2 Микросервисне архитектуре

Почетак развоја софтвера обележен је монолитном архитектуром, тј. архитектуром која се развија у једном, а затим користи у другом окружењу. Проблем разлике између окружења решавају контејнери, омогућавајући да софтвер има исту конфигурацију у свим окружењима. Међутим, појављују се проблеми развоја и испоруке готовог производа. Квалитет софтвера опада и непотребни трошкови се повећавају. С тим се појавила потреба за развијањем софтвера у мањим, али употребљивим целинама. Што описује концепт микросервисне архитектуре, уједно и концепт *DevOps-a*.
DevOps има за циљ да смањи разлику између развоја и одржавања софтвера. Аутоматизује свакодневне операције развоја софтвера, континуалном интеграцијом и континуалном испоруком (*CI/CD*). Тиме развој софтвера постаје флексибилан по питању надоградње или мењања постојеће имплементације.
Микросервисна архитектура омогућава брзу, честу и поуздану испоруку великих, сложених апликација. Главни циљ микросервисне архитектуре јесте декомпозиција и слаба међусобна зависност компоненти сложених апликација. Коришћење микросервиса подразумева да се дате апликације шире на неколико услуга или платформи, од којих су многе у јавни облак, тако да је њихово обезбеђење изазов због сложености развоја, тежине праћења, отклањања грешака и ревизије целе апликације у страним окружењима.
Претходно поменута два концепта добијају на популарности појавом контејнера и нивелизују претходне губитке.

3. БЕЗБЕДНОСТ У АПЛИКАЦИЈАМА

Бројне развојне методе заснивају се на додавању безбедности готовом софтверском производу, што резултује лошим перформансама и потешкоћама приликом прилагођавања софтвера ригидним политикама. Након адаптације тих политика, или је отежано коришћење апликације или сигурност није на задовољавајућем нивоу. Како не би губили време на адаптацију безбедоносних механизама на крају, решење постаје развој безбедности као градивног елемента изградње сотвера.
Континуираном имплементацијом и интеграцијом, апликација се свакодневно усавршава. Постаје све већи изазов што раније пронаћи грешке. У раним фазама се укључују алати за заштиту апликација. Безбедносни алати морају радити истом брзином као и развој апликације како би брже пронашли проблеме у коду. Претње постају све сложеније, теже се проналазе и моћније су у потенцијалном наношењу штете. Постоје разне технике за примену безбедности. Неке од њих се

заснивају на имплеметацији безбедносних мера, неке у провери недостатака тих мера, а неке на анализирању рањивих места у имплементираном коду. Користећи контејнерске платформе пожељно је укључити и безбедносне функције и проценити да ли су слике других корисника злонамерне, тј. да ли садрже *backdoor* или *cron* послове [8].
Постоји велики број препорука за заштиту од напада, у овом раду дат је акценат на *OWASP TOP 10 Vulnerabilities*, најпознатијој листи рањивости.

То је листа 10 рангираних рањивости по консензусу стручњака за безбедност из целог света. Ажурира се у складу са напретком и променама на *AppSec* тржишту. Важност листе лежи у практичним информацијама које пружа као контролна листа и интерни стандард за развој апликација. Рангиране рањивости из 2021. године:

1) Компромитована аутентификација (*Broken Access Control*)
2) Криптографске грешке (*Cryptographic Failures*)
3) Инјекција (*Injection*)
4) Несигуран дизајн (*Insecure Design*)
5) Погрешно исконфигурисана безбедност (*Security Misconfiguration*)
6) Рањиве и застареле компоненте (*Vulnerable and Outdated Components*)
7) Грешке при идентификацији и аутентификацији (*Identification and Authentication Failures*)
8) Грешке у софтверу и интегритету података (*Software and Data Integrity Failures*)
9) Безбедносно евидентирање и праћење грешака (*Security Logging and Monitoring Failures*)
10) Фалсификовање захтева на страни сервера (*Server-Side Request Forgery*)

4. РАЊИВОСТИ

У овом поглављу биће приказани недостаци контејнера, јер сваки недостатак у систему чини потенцијално место напада, тј. потенцијалну рањивост система.
Нападач система може бити човек, аутоматизовани бот или унутрашњи напади, који настају због грешака у безбедносној имплементацији. Стога је важно разумети безбедносне циљеве и формулисати безбедносне захтеве, сходно томе и политике контроле приступа.

4.1 Рањивости у контејнерима

Примарни тип безбедносне рањивости у контејнерима су системски позиви који нису свесни именског простора и на тај начин могу довести до случајног цурења информација између контејнера. Сет *API*-ја за *Linux* системске позиве је огроман и њихов процес ревизије још увек траје.
Контејнери обезбеђују слој изолације између апликација у одвојеним контејнерима, чиме се повећава безбедност. Међутим, контејнери и даље морају да комуницирају са другим контејнерима и системима, и стога остају подложни даљинском искоришћавању безбедносних рањивости уметнутих у слике контејнера.

Контејнери се сматрају мање безбедним од ВМ. Ниво изолације је често оно на шта се позива. Механизмом изолације у контејнеру управљају три главне технике: *cgroups, namespace*s и *chroot*. Начин на који су контејнери имплементирани доводи до тога да они директно комуницирају са језгром оперативног система хоста. Тако, компромитовани контејнер има краћи пут да компромитује главни оперативни систем, поред тога има и апликације са којима се налази на главном хосту, за разлику од виртуелних машина. Код виртуелних машина, нападач мора прво да заобиђе виртуелно језгро ОС-а и хипервизор, пре него што стигне до језгра оперативног система хоста.

Разлика у изолационом механизму подразумева да контејнери имају веће поверење рачунарске базе. Дакле, контејнери имају више компоненти које су критичне за њихову безбедност. Ове компоненте чине сам контејнер, ОС или оркестратор контејнера. Рањивост унутар било које од ових компонената може угрозити безбедносна својства целог система контејнера. Поред велике рачунарске базе постоје претње у раду са *Docker* сликама. Један од њих је могућност непознатих, потенцијално малициозних, корисника да окаче слику. Свака окачена слика на *Docker Hub*-у се може користити у било ком пројекту. Још један од могућих потенцијалних напада у *cloud*-у, је јавно доступан код. Корисник куповином производа добија лиценцу за коришћење истог. Трећа страна може увидети и искористити недостатке тог производа. Већа поуздана рачунарска база мотивише повећање потреба за скенерима рањивости у контејнерима [9].

5. ПРЕГЛЕД СКЕНЕРА РАЊИВОСТИ

Узимајући у обзир све факторе који утичу на појаву рањивости, постоји широк спектар скенера који се разликују по томе шта и како скенирају. Нема идеалног скенера који ће обухватити све могуће провере рањивости.

Упоређивани скенери су: *Clair, Dagda , Falco* и *Aqua*

Security. То су популарни скенери *Docker* слика или окружења. Који обезбеђују статичко скенирање користећи познате базе рањивости и имају јавно доступан код. Сви су дизајнирани тако да се могу уградити у *CI/CD* процес. Већина њих се може интегрисати са другим алатима (*Clair, Dagda*).

Dagda користи антивирусни механизам за откривање рањивости. *Falco* скенира претње у *Kubernetes*-у. *Clair* и *Falco* омогућавају писање сопствених правила за скенирање. *Aqua Security* поред излистаних рањивости приказује и препоруке за решавање истих. Идентификоване су битне сличности и разлике функционалности скенера. Како је тема овог рада *Anchore*-ов пројекат *Grype,* наредно поглавље резервисано је за анализу пројекта.

6. GRYPE

Grype је алат за детекцију рањивости у контејнерски базираним апликацијама развијен од стране компаније *Anchore*.

Овај алат се лако може интегрисати у *CI/CD* како би пронашао безбедносно угрожене делове система. Углавном се користи са *Syft*-ом, алатом за генерисање софтверске листе материјала, *SBOM*. Односно листе свих компоненти отвореног кода и компоненти трећих страна присутних у бази кода.

Написан је на *python* програмском језику. Изворни код се налази на *gitHub*-у. Омогућен је и као *Docker* слика, постављена на *DockerHub*-у. Придруживањем *Anchore* заједници, било ко може допринети побољшању перформанси *Grype-a*.

7. РЕЗУЛТАТИ СКЕНИРАЊА *GRYPE-OM*

Након инсталације *Grype*-а, поступак скенирања започиње командом *grype naziv_slike*. Прво се локално ажурира последња верзија базе рањивости и повлачи слика са *Docker Hub-a.* Затим се слика парсира и рашчлањује. Потом индексира директоријум, проналазе се сви остали пакети *(Cataloged packages)* и на крају се преброје и прикажу познате рањивости.
Анализиране су две званичне слике. Прва је *postgres*, а друга *nginx*. Резултат скенирања се састоји из назива рањивости, верзије инсталиране у слици, глобално доступне поправљене верзије, типа пакета, *CVE* рањивости и тежине сваке рањивости.

7.1 Postgres слика

У *Postgres* слици пронађено је 721 пакет и 122 рањивости. Тежина рањивости може бити *negligible, unknown, low, medium, high* и *critical*. Како се *negligible, unknown* и *low* рањивости могу сматрати небитним наспрам осталих, нису разматране. Такође нису разматране ни глобално поправљене рањивости, јер их је потребно само заменити у слици. Рањивости које су имале вредност *(won't fixed*) у колони поправљене верзије су занемарене, из разлога што за њих постоји претходна верзија која није имала рањивости. Преостале су CVE-2021-30560 и CVE-2022-37434 рањивости. Прва је типа *High* и представља рањивост из 2021. године. Појављује се након инсталирања 91.0.4472.164 верзије *Google Chrome*-а. Друга је типа *Critical* и приказује рањивост *zlib* датотеке која омогућује компресију датотека у *gzip* формату.

7.2 Nginx слика

У *Nginx* слици пронађено је 239 пакета и 129 рањивости. Анализа *nginx*-а се базира на истим полазним претпоставкама као у претходној процени. Критичне су CVE-2011-3389, CVE-2021-30560 и CVE-2022-37434 рањивости. Прва је типа *Medium* и то је проблем са *SSL* протоколом у одговарајућим конфигурацијама, омогућава појаву *middleman*-а. Друге две су рањивости које су поменуте у претходном потпоглављу.

8. ЗАКЉУЧАК

Овим радом је објашњен значај виртуелизације, постојање виртуелних машина, али и појава њихове алтернативе, контејнера.

Описан је развој апликације кроз континуалну интеграцију и континуални развој као једно од најбољих решења за развој апликације и сузбијање безбедоносних претњи. Приказани су потенцијални проблеми од имплементације до извршавања контејнера. На крају је објашњен скенер *Grype*.

Употребом скенера и пропратних алата који се повезују са апликацијом, у циљу праћења оптерећења и безбедносних елемената откривање напада је брже.

Неки од поменутих алата се заснивају на скенирању рањивости од окружења до саме *Docker* слике. Једно од могућих унапређења *Grype*-а би било уградња дела који скенира део процеса везаног за рањивост окружења. Истраживањем недостатака овог скенера допринеће његовом развоју. Поред проналажења недостатака скенера, велики допринос за развој безбедности би био решавање постојећих рањивости.

Потенцијални развој скенера би се могао развијати у правцу откривања и решавања рањивости, истраживањем и коришћењем *ML*-а (*machine learning*).

9. ЛИТЕРАТУРА

[1] Ebert, Christof, et al. "DevOps." *Ieee Software* 33.3, 2016, стр. 94-100.

[2] Qadir, Suhail, and S. M. K. Quadri. "Information availability: An insight into the most important attribute of information security." *Journal of Information Security* 7.3, 2016, стр. 185-194.

[3] OWASP TOP TEN, https://owasp.org/www-project-top-ten/ (Последњи приступ август 2022.)

[4] Grype, https://github.com/anchore/grype (Последњи приступ август 2022.)

[5] Kemmerer, Richard A. "Cybersecurity." *25th International Conference on Software Engineering, 2003. Proceedings.*. IEEE, 2003.

[6] Sharma, Prateek, et al. "Containers and virtual machines at scale: A comparative study." *Proceedings of the 17th international middleware conference,* 2016.

[7] Agarwal, Gaurav. *Modern DevOps Practices*, 2021, стр. 28

[8] Abbott, Brendan Michael. *A security evaluation methodology for container images*, 2017

[9] Jagelid, Michelle. "Container Vulnerability Scanners: An Analysis.", 2020, стр. 24

Кратка биографија:

Анђела Трајковић рођена је у Врању 1999. год. Мастер рад на Факултету техничких наука из области Електротехнике и рачунарства – Софтверско инжењерство и информационе технологије одбранила је 2022.год.
контакт: trajkovic.andjela99@gmail.com