Proaktivno obvladovanje tveganj v informacijskih sistemih

Andrej Dobrovoljc

doktorska disertacija

predana

Fakulteti za računalništvo in informatiko

kot del izpolnjevanja pogojev za pridobitev naziva

doktor znanosti

s področja

računalništva in informatike

Ljubljana, 2018

Naložba v vašo prihodnost
Operacijo delno financira Evropska unija
Evropski socialni sklad

Proaktivno obvladovanje tveganj v informacijskih sistemih

Andrej Dobrovoljc

doktorska disertacija
predana
Fakulteti za računalništvo in informatiko
kot del izpolnjevanja pogojev za pridobitev naziva
doktor znanosti
s področja
računalništva in informatike

Ljubljana, 2018

IZJAVA

Izjavljam, da sem avtor dela in da slednje ne vsebuje materiala, ki bi ga kdorkoli predhodno že objavil ali oddal v obravnavo za pridobitev naziva na univerzi ali na drugem visokošolskem zavodu, razen v primerih, kjer so navedeni viri.

— Andrej Dobrovoljc —

september 2018

Oddajo so odobrili

dr. Denis Trček

redni profesor za računalništvo in informatiko

MENTOR IN ČLAN OCENJEVALNE KOMISIJE

dr. Borut Likar

redni profesor za management

SOMENTOR IN ČLAN OCENJEVALNE KOMISIJE

Univerza na Primorskem

dr. Marko Bajec

redni profesor za računalništvo in informatiko

PREDSEDNIK OCENJEVALNE KOMISIJE

dr. Matjaž Gams

redni profesor za računalništvo in informatiko

ZUNANJI ČLAN OCENJEVALNE KOMISIJE

Institut Jožef Stefan

PREDHODNA OBJAVA

Izjavljam, da so bili rezultati obravnavane raziskave predhodno objavljeni/sprejeti za objavo v recenzirani reviji ali javno predstavljeni v naslednjih primerih:

[1] DOBROVOLJC, Andrej, TRČEK, Denis, LIKAR, Borut. Predicting Exploitations of Information Systems Vulnerabilities Through Attackers´ Characteristics. *IEEE Access*, 2017. doi: 10.1109/ACCESS.2017.2769063

Potrjujem, da sem pridobil pisna dovoljenja vseh lastnikov avtorskih pravic, ki mi dovoljujejo vključitev zgoraj navedenega materiala v pričujočo disertacijo. Potrjujem, da zgoraj navedeni material opisuje rezultate raziskav, izvedenih v času mojega podiplomskega študija na Univerzi v Ljubljani.

Staršema Darinki in Alojzu ter vzorniku, pokojnemu tastu Janezu Kramarju.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*
Andrej Dobrovoljc
Proaktivno obvladovanje tveganj v informacijskih sistemih

POVZETEK

Obvladovanje varnostnih tveganj je eden večjih izzivov v sodobnih informacijskih sistemih. Grožnje pogosto prihajajo preko svetovnega spleta in jih je težko predvideti. Napadalci so tako lahko vedno korak pred nami in ukrepanje zgolj na osnovi znanih incidentov ni zadostno. S stalnim aktivnim odkrivanjem in odstranjevanjem ranljivosti v programski opremi lahko dosežemo precej višjo raven varnosti. Kadar je v sistemu prisotno večje število ranljivosti, se moramo odločati, kateri bomo dali prednost pri odstranjevanju. S proaktivnim pristopom, kjer predvidevamo, katere ranljivosti bodo v praksi bolj verjetno izkoriščene, lahko zagotovimo najvišjo raven varnosti. Najpogosteje uporabljena metoda določanja prioritet, ki temelji na oceni CVSS (Common Vulnerability Scoring System), je pogosto tarča kritik zaradi slabe učinkovitosti. Zgolj na osnovi ocene CVSS namreč ne moremo sklepati o verjetnosti izkoriščanja. Eden ključnih izzivov na tem področju je torej prepoznati indikatorje izkoriščanja. Ker je izkoriščanje ranljivosti v osnovi človeška grožnja, je pri predvidevanju izkoristljivih ranljivosti smiselno upoštevati značilnosti tipičnih napadalcev. Opredelili smo več metod določanja prioritet, ki to upoštevajo. Učinkovitost metod želimo med seboj primerjati glede na uspešnost pri omejevanju tveganja. V ta namen smo razvili model vrednotenja, ki omogoča takšne primerjave. Predlagane metode določanja prioritet, ki upoštevajo človeške grožnje, smo primerjali z najbolj priljubljenimi obstoječimi metodami. Ob tem smo uporabili podatke o ranljivostih iz javno dostopnih podatkovnih zbirk. Eksperimentalni rezultati kažejo, da so metode določanja prioritet, ki upoštevajo značilnosti napadalcev, v splošnem učinkovitejše od obstoječih metod. Učinkovitost se je potrdila tudi na nekaterih realnih primerih informacijskih sistemov v praksi.

Ključne besede: tveganje, grožnja, ranljivost, napadalec, kvantitativna ocena, metoda določanja prioritet

University *of Ljubljana*
Faculty *of Computer and Information Science*
Andrej Dobrovoljc
Proactive risk management in information systems

ABSTRACT

Managing security risks is one of the major challenges in modern information systems. Threats often come via the World Wide Web and are therefore difficult to predict. Thus, attackers can always be a step ahead of us and reactive approach based on known security incidents is not sufficient. A much higher security level can be achieved by active detection and neutralization of software vulnerabilities. When a large number of vulnerabilities are present in the system, they have to be prioritized for removal according to their severity. With a proactive approach, where we foresee which vulnerabilities will be more likely exploited in practice, the highest level of security can be assured. A widely used prioritization policy based upon a CVSS (Common Vulnerability Scoring System) score is frequently criticised for bad effectiveness. The main reason is that the CVSS score alone is not a good predictor of vulnerability exploitation in the wild. One of the key challenges in this area is therefore to identify the indicators of exploitation. Since the exploitation of vulnerability is basically a human threat, it is reasonable to take into account the characteristics of typical attackers. We propose several methods for setting priorities that take this into account. Methods have to be compared according to their effectiveness in risk mitigation. To this end, we have developed a valuation model that allows such comparisons. Proposed methods, which take into account human threats, were compared with the most popular existing methods. In the experiment we used vulnerability data from publicly available databases. Experimental results show that methods which take into account the characteristics of attackers are generally more effective than existing methods. The effectiveness was also confirmed in some real cases of information systems in practice.

Key words: risk, threat, vulnerability, threat agent, quantitative assessment, prioritization policy

ZAHVALA

Iskreno se zahvaljujem vsem, ki ste me na različne načine podpirali pri tej zahtevni nalogi. Najprej se zahvaljujem mentorju prof. dr. Denisu Trčku in somentorju prof. dr. Borutu Likarju za nasvete, usmeritve in zaupanje.

Posebej bi se rad zahvalil dr. Borutu Lužarju, ki me je v najtežjih trenutkih znal vzpodbuditi in najti v mojih idejah dovolj pozitivnih stvari, da sem vztrajal in dokončal delo. Za pomoč pri statističnih izračunih se zahvaljujem dr. Janezu Povhu. Zahvaljujem se tudi nekdanjim sodelavcem dr. Jeleni Klisara, dr. Jožetu Bučarju in Darku Zeleniki za poglobljene diskusije in vzpodbudo.

Največja zahvala za podporo in hkrati opravičilo gresta moji družini. Marjetka, Andreja, Vid in Sara, veliko ste mi pomagali s tem, ko ste pokazali razumevanje za mojo odsotnost, ko bi raje videli, da sem z vami.

— Andrej Dobrovoljc, Ljubljana, september 2018.

KAZALO

I

Uvod

1.1 Varnostna tveganja v informacijskih sistemih

Povezanost informacijskega sistema (IS) s svetovnim spletom je danes nekaj samoumevnega. Razvile so se povsem nove informacijske rešitve, ki smo si jih včasih težko predstavljali. Ob tem so nam krepko spremenile način življenja (npr. nakupovanje, opravljanje bančnih storitev, komuniciranje ipd.). Vse širše se uveljavlja tudi koncept interneta stvari (angl. Internet of Things). Kljub številnim koristim omenjenih informacijskih rešitev se vse bolj zavedamo občutljivosti slednjih na kakršne koli motnje ali nepredvidljiv način njihovega delovanja [1, 2]. Za primer vzemimo nekaj zanimivih naslovov iz znane slovenske revije s področja računalništva:[1]

- *Luknja v TeamViewerju omogoča oddaljen prevzem nadzora*,
- *Ranljivost v estonski osebni izkaznici*,
- *Spletne ranljivosti v pomivalnih strojih*,
- *Prastara ranljivost v modernih vozilih*,
- *Milijoni Intelovih procesorjev ranljivi zaradi skrivne kode*,
- *V pametni televizor lahko vdremo tudi po zraku*,
- *Pametna plišasta igrača otroške glasove poslala v splet*,
- *Hekerski napad preko akvarija*,
- *Srčni spodbujevalniki so bolj ranljivi, kot smo mislili*,
- *Varnostne vrzeli so velik posel*.

Tveganje pomeni verjetnost, da se bo zaradi določenih zunanjih ali notranjih dejavnikov zgodila neka škoda oziroma kakršen koli drug negativen izid. Ob zagonu inovativnih informacijskih rešitev ter z njimi povezanih storitev ponudniki tako pogosto preveč razmišljajo le o njihovi odmevnosti in o hitri rasti števila uporabnikov. Za priljubljene produkte namreč lahko zanimanje pokažejo tudi posamezniki, ki želijo produkt zlorabiti. V informacijski rešitvi morda prepoznajo sredstvo za izpolnitev svojih zlobnih ciljev. Tveganja so torej močno povezana s stalnimi spremembami v okviru IS in v uporabniškem okolju. Potrebno jih je obvladovati ali še bolje predvidevati.

[1] http://www.monitor.si/novice/

1.2 Proaktivno obvladovanje varnostnih tveganj

Tveganja pri IS v osnovi povezujemo z ranljivostmi [1]. Vsak IS je ranljiv, in sicer na več različnih načinov. Pomembno področje predstavljajo ranljivosti programske opreme (angl. Software Vulnerability). Kako pomembno je to področje, se lahko prepričamo ob pogledu na sezname odkritih ranljivosti. Po podatkih podatkovne zbirke *National Vulnerability Database* (NVD),[2] ki beleži ranljivosti vseh pomembnih programskih produktov na trgu, je bilo od leta 1997 odkritih že več kot 100.000 ranljivosti. Ker je ista ranljivost lahko prisotna v več produktih, se je organizacija *Secunia Research* odločila, da šteje vsak pojav ranljivosti posebej [3]. Po njihovih podatkih je bilo samo v letu 2016 na novo odkritih 17.147 ranljivosti, kar predstavlja 33-odstotno rast v trendu zadnjih petih let. Te visoke številke dokazujejo, da je odkrivanje ranljivosti postalo zelo razširjena dejavnost. Z njo se ukvarjajo številni posamezniki z zelo različnimi interesi, saj je to znanje po nekaterih raziskavah zelo cenjeno [4–6]. Vse skupaj se posledično odraža v vedno novih grožnjah, kar za IS pogosto predstavlja težko predvidljivo tveganje.

Obvladovanje tveganj je pomembno v celotnem življenjskem ciklu IS in v vsakem obdobju se jih lotevamo na različne načine. Najučinkovitejše je preventivno delovanje. Na stopnji razvoja programske opreme na primer preprečujemo nastajanje ranljivosti. Precej bolj kritično obdobje nastopi, ko IS predamo v redno uporabo. Takrat se moramo ukvarjati z odstranjevanjem oz. nevtralizacijo odkritih ranljivosti. Vseh ranljivosti žal nikoli ne moremo preprečiti že na stopnji razvoja.

Kljub vsemu je uspeh pri obvladovanju tveganj v vseh obdobjih odvisen predvsem od naše aktivnosti oz. pasivnosti. Kadar po izvedenem napadu zgolj odpravljamo škodo in iščemo zaščito za naprej, govorimo o *pasivnem* pristopu. *Aktivni* pristop povezujemo na primer z iskanjem in odstranjevanjem ranljivosti v IS, še preden se je zgodil napad. Zavedati se moramo, da je zagotavljanje varnosti stalno premikajoči se cilj (angl. Moving Target). Pravi uspeh si torej lahko obetamo z uporabo *proaktivnega* pristopa. Predvidevati moramo korake napadalcev [7]. Ena izmed možnosti je predvidevanje ranljivosti, ki jih bodo napadalci izkoriščali.

[2] https://nvd.nist.gov/

1.3 Motivacija

Varovanje IS pred vedno novimi grožnjami za organizacije predstavlja velik strošek. Da bi zagotovili varnost in hkrati ohranili stroške varovanja v omejenem obsegu, so podjetja prisiljena obvladovati tveganja z uporabo različnih metod. Ker vsaka ranljivost v rokah vsiljivcev predstavlja drugačno raven grožnje, je treba uporabiti ustrezno metodo določanja prioritet [8, 9].

Naloga skrbnika IS je, da najprej odstrani tiste ranljivosti, ki predstavljajo največje varnostno tveganje. Ker se ranljivosti med seboj močno razlikujejo, je zelo pomembno, da jih lahko med seboj primerjamo. Za verodostojno primerjavo resnosti groženj, ki jih predstavljajo posamezne ranljivosti, so zaželene kvantitativne ocene, opredeljene na standardiziranem postopku ocenjevanja, kar pa je težko doseči.

V letu 2005 je organizacija *National Infrastructure Advisory Council* (NIAC) predstavila odprti standard z imenom *Common Vulnerability Scoring System* (CVSS). Ta omogoča kvantitativno ocenjevanje resnosti ranljivosti [10]. Njegov osnovni namen je omogočiti izmenjavo informacij o ranljivostih med različnimi udeleženci na IT področju, tj. skrbniki IS, ponudniki programske opreme, raziskovalci idr.

Standard CVSS je danes široko sprejet. Postal je sestavni del avtomatiziranih orodij za odkrivanje ranljivosti, ki so zasnovana na protokolu SCAP [3] [11, 12]. Ocene resnosti ranljivosti na osnovi standarda CVSS se pogosto uporabljajo za določanje prioritet pri njihovem odstranjevanju. Kljub temu obstajajo resni dvomi o primernosti uporabe ocen CVSS v ta namen [13]. Raziskave namreč dokazujejo, da visoka ocena CVSS še ne pomeni, da bo nekoč v resnici prišlo do izkoriščanja ranljivosti [14]. Nevarnost, ki je predstavljena z oceno CVSS, se ne ujema s tveganjem v realnem okolju. Še več, nekateri celo trdijo, da je učinkovitost takšne metode primerljiva s pristopom, kjer ranljivosti za odstranitev izbiramo povsem naključno [12].

Kakor koli že, lastniki IS potrebujejo jasen odgovor na vprašanje, katera metoda razvrščanja ranljivosti po prioriteti je najučinkovitejša [15]. Prav zato potrebujemo orodje, ki bi omogočalo takšne primerjave. Ker v literaturi nismo zaznali rešitev tega izziva, smo se odločili, da v ta namen razvijemo svoj model.

Obstaja veliko različnih metod določanja prioritet. V primeru proaktivnega pristopa potrebujemo metodo, ki bo sposobna predvidevati, katere ranljivosti bodo v realnem okolju napadalci najverjetneje izkoriščali. Da bi izboljšali možnosti predvide-

[3] https://scap.nist.gov/

vanja, je ključnega pomena, da prepoznamo lastnosti v okviru IS ter njegovi okolici, ki bi lahko predstavljale indikatorje izkoriščanja ranljivosti [16]. Na primer, znano je, da vključitev lastnosti napadalcev v oceno tveganja izboljša njeno točnost [17]. Pričakujemo torej, da z upoštevanjem lastnosti napadalcev lahko izboljšamo metodo določanja prioritet.

1.4 Prispevki k znanosti

V disertaciji predstavimo naslednja prispevka k znanosti.

Razvijemo model za merjenje učinkovitosti metod pri odstranjevanju ranljivosti. Predstavljen je model za ugotavljanje učinkovitosti različnih metod pri odstranjevanju ranljivosti. Pri informacijskih sistemih, ki so že v redni uporabi, lahko omejujemo tveganja z različnimi metodami določanja prioritet ranljivostim. Za izvajanje primerjav med različnimi metodami potrebujemo model, ki omogoča takšne primerjave. V ta namen opredelimo merski sestav, ki meri izpostavljenost izkoristljivih ranljivosti informacijskega sistema grožnjam, kar predstavlja osnovo za ocenjevanje učinkovitosti.

Razvijemo metode določanja prioritet ranljivostim z upoštevanjem lastnosti napadalcev. Predstavimo več metod obravnavanja ranljivosti, ki temeljijo na preverjanju zmožnosti napadalcev. V ta namen opredelimo funkcijo izkoriščanja ranljivosti, ki sledi definicijam grožnje v literaturi. Metode razvijemo nad opisi ranljivosti, ki jih opredeljuje standard CVSS. Prednost dajemo ranljivostim, ki jih lahko izkoristi več napadalcev. Primerjave z več obstoječimi metodami kažejo, da so metode z upoštevanjem lastnosti napadalcev učinkovitejše. Njihova prednost se kaže v boljšem predvidevanju izkoristljivih ranljivosti. Predstavimo tudi nov sistem ocenjevanja ranljivosti ABVS, ki natančneje predstavlja raven tveganja posamezne ranljivosti za informacijski sistem kot ocena CVSS.

1.5 Pregled vsebine

Disertacija obsega sedem poglavij in dodatek.

- V 2. poglavju opredelimo osnovne koncepte, kot so ranljivost, grožnja in tveganje. Pri vsakem konceptu podamo njegove osnovne lastnosti, ki so v nadaljevanju pomembne pri predstavitvi metod obvladovanja tveganj.

- Poglavje 3 podaja pregled obstoječih metod za obvladovanje tveganj v IS s poudarkom na programski opremi. Ločeno predstavimo metode, ki se uporabljajo v času razvoja programske opreme, in metode, ki so uporabne v času uporabe IS. Poglavje zaključimo s pregledom odprtih problemov.

- V 4. poglavju predstavimo model ocenjevanja resnosti ranljivosti glede na lastnosti tipičnih napadalcev. V ta namen opredelimo funkcijo izkoriščanja ranljivosti, ki je odvisna od lastnosti napadalca in ranljivosti. Za vsako komponento grožnje posebej opredelimo pogoje, ki jih mora izpolniti akter.

- V 5. poglavju predstavimo model merjenja učinkovitosti metod za določanje prioritet ranljivostim pri odstranjevanju iz IS. V ta namen opredelimo kvantitativno metriko za merjenje izpostavljenosti IS izkoristljivim ranljivostim. Zatem predstavimo simulacijsko okolje za izvedbo eksperimenta, ovrednotimo predlagani model in predstavimo rezultate.

- V 6. poglavju predlagano metodo določanja prioritet dodatno izboljšamo z uporabo naključno ustvarjenih napadalcev. V ta namen razvijemo dodatno simulacijsko okolje. Opredelimo tudi merski sestav za oceno učinkovitosti metod pri zmanjševanju tveganja. Metode nato ovrednotimo in primerjamo ter na osnovi rezultatov predlagamo nov sistem ocenjevanja ranljivosti.

- V 7. poglavju predlagane metode določanja prioritet ovrednotimo na realnih primerih iz prakse. Pridobljene rezultate nato primerjamo s sorodnimi raziskavami in ocenimo uporabnost metod za praktične namene.

- Poglavje 8 povzema predstavljeno vsebino, glavne rezultate in ugotovitve ter navaja raziskovalne izzive za prihodnost.

- V dodatku A podajamo primer statistične analize rezultatov.

2

Sestavine tveganja

2.1 Uvod

Temeljni cilj znanosti je razviti modele, s katerimi lahko izvajamo natančne napovedi. Pri računalniški varnosti žal še vedno velja, da je povsem zanesljivo le to, da bo sistem nekoč utrpel neko škodo, če se bo le soočil z dovolj motiviranimi in usposobljenimi napadalci. Razlog tiči v dejstvu, da je zagotavljanje varnosti močno prepleteno s človeškim dejavnikom [18]. Prav zato moramo na tem področju razvijati modele, ki upoštevajo človeški dejavnik ter posledično prispevajo k boljšemu obvladovanju tveganj.

V tem poglavju predstavimo ključne pojme in definicije, ki se nanašajo na področje obvladovanja varnostnih tveganj v IS. V uvodnem delu podajamo podroben pregled področja ranljivosti v programski opremi. Nato pojasnimo življenjski cikel ranljivosti ter opišemo vloge in interese posameznikov v procesu njihovega odkrivanja, razkrivanja in nevtralizacije. V nadaljevanju sledi opis razpoložljivih podatkovnih virov o razkritih ranljivostih. V drugem sklopu poglavja predstavimo pojem grožnje ter podamo nekaj smernic za njihovo razumevanje in obvladovanje. Ob tem se nekoliko podrobneje usmerimo na opis groženj, ki so povezane z izkoriščanjem ranljivosti v programski opremi. Poglavje zaključimo s predstavitvijo različnih definicij tveganja in osnovnimi pristopi pri njihovem obvladovanju.

2.2 Ranljivosti

2.2.1 Definicija

Ranljivosti povezujemo z vsemi komponentami, ki sestavljajo IS. Standard NIST SP 800-30 jih opredeljuje na naslednji način:

Definicija 2.1 (Opredelitev ranljivosti po standardu NIST): Ranljivost je pomanjkljivost ali slabost sistema v postopkih varovanja, zasnovi, izvedbi ali notranjem nadzoru (sprožena pomotoma ali namerno izkoriščena), ki se odrazi v kršitvi politike varovanja sistema.

V okviru naše obravnave se bomo omejili le na ranljivosti v programski opremi, saj je področje preširoko in preveč raznoliko, da bi bilo možno in smiselno naenkrat zajeti vse vrste.

2.2.2 *Življenjski cikel ranljivosti*

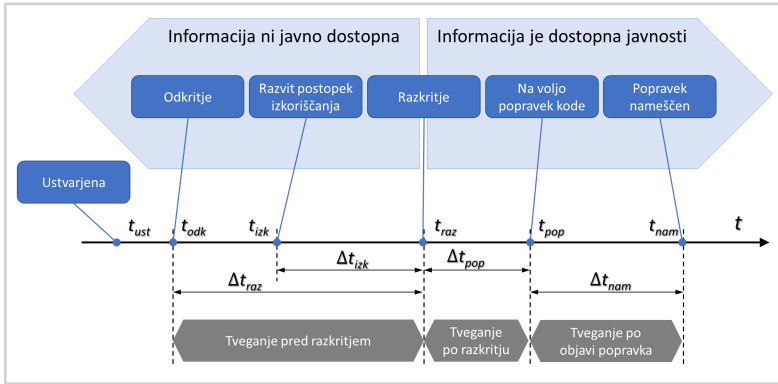
Ranljivosti so praviloma posledica neustrezne kakovosti programske opreme. Stroga pravila zagotavljanja kakovosti v procesu razvoja žal ne jamčijo produktov brez ranljivosti. Pomanjkanje ustreznih praks ob prehodu na nove tehnologije je lahko eden izmed pomembnih dejavnikov tveganja. Na število odkritih ranljivosti vplivajo tudi organizacijski in družbeni dejavniki v okolju, ki so jim podvrženi razvijalci programske opreme [19]. Pomembno je postalo razmišljanje o tem, česa programska oprema ne sme delati, in ne zgolj, katere funkcionalnosti bo imela [20].

Takanen s sodelavci [21] podaja osnovni pregled vseh akterjev, ki sodelujejo v procesih preprečevanja in odpravljanja ranljivosti v programski opremi z namenom zagotavljanja boljše kakovosti. Razvršča jih v več skupin in v njih prepozna 3 osnovne vloge: zagotavljanje kakovosti, odobritev produktov in obvladovanje ter rokovanje z ranljivostmi. Preprečevanje in odpravljanje ranljivosti je postal standardni del življenjskega cikla razvoja programske opreme.

Z vidika zagotavljanja varnosti IS je zelo pomembno, da ranljivosti odkrijemo čim prej in jih odpravimo. Arbaugh s sodelavci [22] predlaga model življenjskega cikla ranljivosti (angl. Vulnerability Lifecycle), v katerem loči naslednja stanja: rojstvo, odkritje, pojav postopka izkoriščanja, razkritje, razpoložljivost popravka in nevtralizacija. Ob tem poudarja, da vse ranljivosti ne zavzamejo nujno vseh stanj in v istem vrstnem redu, kot so našteta (npr. možnost namernega ne-razkritja avtorju znane ranljivosti). Z boljšim poznavanjem zakonitosti ranljivosti ter posledično razvojem odgovornejšega ravnanja z njimi je bil življenjski cikel kasneje nadgrajen [23]. Ključne faze novega modela so prikazane na sliki 2.1.

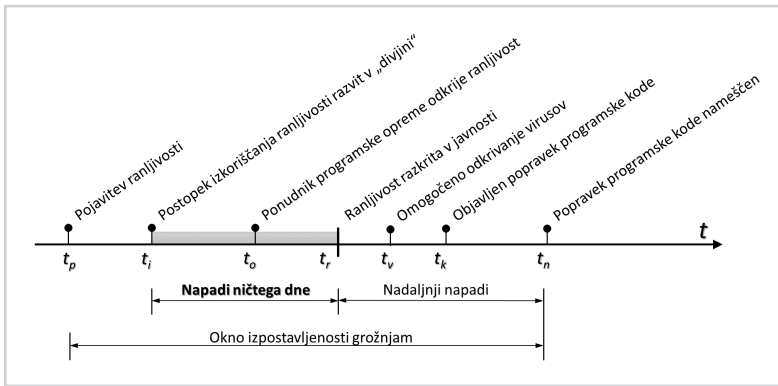
V zvezi s fazami življenjskega cikla nas zanimajo predvsem najnevarnejša obdobja. Označena so na sliki 2.2. Najbolj nevarno je obdobje od trenutka, ko uspe napadalec za neznano ranljivost v tajnosti razviti postopek izkoriščanja (angl. Exploit), do trenutka javne objave informacij o njenem obstoju. V tem času je IS izpostavljen nemotenemu izkoriščanju ranljivosti, saj se grožnje niti ne zavedamo niti nismo zaščiteni pred njo. V literaturi se je takšnih ranljivosti prijelo poimenovanje ranljivosti ničtega dne (angl. 0-Day Vulnerability ali Zero-Day Vulnerability). Ob javnem razkritju ranljivosti avtor programske opreme namreč nima na voljo niti dneva, da bi zakrpal vrzel v produktu.

Z vidika zagotavljanja varnosti pa je v praksi pomembno celotno obdobje izpostavljenosti sistema grožnjam (angl. Window Of Exposure). Slednje traja od nastanka



Slika 2.1

Ključna stanja v življenjskem obdobju ranljivosti. Skozi celotno obdobje se spreminja raven tveganja [4].



Slika 2.2

Okno izpostavljenosti ranljivega informacijskega sistema grožnjam (angl. Window of Exposure). Najbolj kritično je obdobje od trenutka, ko napadalcu uspe razviti postopek izkoriščanja ranljivosti, do trenutka javne objave obstoja ranljivosti [24].

ranljivosti do njene nevtralizacije. Analiza [24] kaže, da lahko nekateri napadi neprekinjeno ostajajo neopaženi tudi več mesecev ali celo let, preden jih nekdo odkrije.

Naloga skrbnika IS je, da s primernimi metodami obdobje izpostavljenosti čim bolj skrajša in s tem izboljša varnost. To je tudi glavni cilj proaktivnega pristopa pri obvladovanju tveganj v IS.

2.2.3 *Razkrivanje ranljivosti*

Preventivni pristop v smislu preprečevanja nastajanja novih ranljivosti je dolgoročno najučinkovitejši. Za dvig kakovosti programske opreme je Schechter [25] predlagal

vzpostavitev trga ranljivosti kot oblike zunanjega testiranja. Ponudniki programske opreme bi se v časovno omejenem obdobju zavezali, da bodo nagradili vsakogar, ki bo našel ranljivost v njihovem produktu. Za najbolj učinkovito in pred napadi varno izvedbo takšnega trga je Ozment [26] predlagal princip dražbe.

Pritisk na dobavitelje programske opreme, da bi zagotavljali varne produkte, poteka tudi z javnim razkrivanjem novoodkritih ranljivosti. Uveljavilo se je načelo odgovornega razkrivanja (angl. Responsible Disclosure), ki ščiti uporabnike. Obseg razkrite informacije mora biti toliko omejen, da ne omogoča direktnega izkoriščanja ranljivosti. Analize različnih oblik razkrivanja (delnega ali popolnega) so predstavljene v [27, 28]. Razkritje ranljivosti vpliva namreč tako na ponudnike kot na napadalce. Raziskave so potrdile pozitivne učinke razkrivanja ranljivosti ob predpostavki, da se izoblikuje beli trg (angl. white market) ranljivosti kot močna konkurenca črnemu trgu (angl. black market).

Vzpostavljanje zakonitega trgovanja z ranljivostmi je bilo povezano z vzpostavljanjem zaupanja med zainteresiranimi raziskovalci belega trga in ponudniki programske opreme. Informacije o ranljivostih, predvsem neznanih, ki omogočajo takojšnje izkoriščanje (angl. 0-Day Exploit Vulnerability), so postale dragoceno blago, ki pa z razkritjem v trenutku izgubi vso svojo vrednost. Miller [5] trgovanje s takšnimi ranljivostmi pomenljivo imenuje trg ničtega dne (angl. 0-day market). Opis njegove lastne uspešne prodaje ranljivosti enemu izmed ponudnikov programske opreme je formalni dokaz delovanja tega trga. Danes veliko pomembnih ponudnikov programske opreme odkupuje ranljivosti preko programov nagrajevanja odgovornega razkrivanja (angl. Bug Bounty programs; Mozilla, Facebook, PayPal). Kljub vsemu se moramo zavedati, da trg še ni dokončno oblikovan. Pojavljajo se novi zainteresirani kupci, katerih namere niso vedno povsem jasne. Med njimi najdemo tudi vladne organizacije in različna privatna podjetja [6].

Čas od javnega razkritja ranljivosti do njene zlorabe je lahko zelo kratek. Razvoj postopka izkoriščanja lahko traja le nekaj dni, zato je ta interval tako zelo pomemben [29]. Varnostna tveganja so torej odvisna od hitrosti objavljanja popravkov na strani ponudnikov programske opreme (angl. Patch) kot tudi od hitrosti razvoja postopkov izkoriščanja na strani napadalcev [30]. V primeru učinkovitega delovanja belega trga lahko ponudnik programske opreme pridobi informacije o ranljivosti pred potencialnimi vsiljivci in izdela popravke še pred njeno javno objavo. Frei s sodelavci [4] zato predlaga metriko "0-day patch", ki predstavlja delež ranljivosti v celotni množici ran-

ljivosti izbranega ponudnika, za katere je bil popravek programske kode na voljo že v trenutku javne objave ranljivosti. Z uporabo te metrike lahko preverjamo odnos ponudnikov programske opreme do zagotavljanja varnosti v svojih produktih oziroma njihovo lastno varnostno učinkovitost.

2.2.4 Odstranjevanje ranljivosti

Frei s sodelavci [23] ugotavlja, da ponudniki programske opreme skrbno čuvajo podrobnosti o delovanju svojega internega procesa odpravljanja ranljivosti (angl. Vulnerability Handling). Kljub vsemu ugotavlja, da proces odgovornega razkrivanja danes že zelo dobro deluje. Ponudniki veliko popravkov objavijo sočasno z javnim razkritjem ranljivosti. Med popravki prednjačijo tiste ranljivosti, ki predstavljajo večje tveganje za IS. To je dokaz, da imajo ponudniki v svojem procesu uveljavljen sistem prioritet.

Po drugi strani analiza kaže, da so hekerji pri pripravi rešitev za zlorabo ranljivosti hitrejši od ponudnikov pri pripravi popravkov. Poleg tega izpostavlja problem (ne)rednega nameščanja popravkov s strani IT osebja v podjetjih ter na osebnih računalnikih običajnih uporabnikov, kar lahko predstavlja še veliko večje tveganje [31].

2.2.5 Podatki o ranljivostih

S širšim družbenim razumevanjem ranljivosti ter z njimi povezanimi grožnjami je v zadnjem desetletju popolnoma zaživelo trgovanje z ranljivostmi. To se je zgodilo z razrešitvijo nekaterih etičnih in pravnih vprašanj razkrivanja ter s podporo gibanja za vzpostavitev belega trga [32]. Dodatno podporo procesu razkrivanja nudi več organizacij, in sicer z evidentiranjem vseh odkritih ranljivosti v programski opremi ter z njihovo objavo v javno dostopnih podatkovnih zbirkah. Teh je sicer več (npr. CVE, NVD, OSVDB, Bugtraq, ISS X-Force in druge), a na tem mestu se bomo omejili samo na ključne.

Common Vulnerabilities and Exposures (CVE)

Osnovo predstavlja iniciativa *Common Vulnerabilities and Exposures* (CVE), ki je nastala pod okriljem organizacije MITRE.[1] Ko nekdo odkrije v programu nekaj, za kar verjame, da je ranljivost, lahko pošlje informacijo strokovnjakom CVE. Če se izkaže, da gre za novo ranljivost, ta dobi svojo identifikacijsko številko, osnovni opis in kronološke podatke o odkritju. Temu sledi javna objava v podatkovni zbirki odkritih

[1] https://cve.mitre.org/

ranljivosti CVE. Če za odkrito ranljivost v trenutku objave obstaja tudi že postopek za njeno nevtralizacijo, je dodana tudi informacija o tem, sicer pa jo objavijo, takoj ko je na voljo. Podatkovna zbirka CVE predstavlja temelj za večino metod kvantitativnega ocenjevanja tveganj v IS.

National Vulnerability Database (NVD)

Postopoma so se pojavile potrebe po natančnejšem opisovanju ranljivosti in njihovi medsebojni primerjavi. Pomembni sta postali predvsem vprašanji, katera ranljivost postane v rokah napadalcev večja grožnja ter kako ranljivosti kvantitativno oceniti na enoten in nepristranski način. V letu 2005 je pod okriljem organizacije *National Infrastructure Advisory Council* (NIAC) nastal nov standard z imenom *Common Vulnerability Scoring System* (CVSS), ki ponuja takšno rešitev [10]. Danes razvoj standarda poteka pod okriljem organizacije FIRST (*Forum for Incident Response and Security Teams*).[2]

CVSS standard je logično nadaljevanje projekta CVE. Vsaka ranljivost, ki je potrjena s strani CVE, se v nadaljevanju opiše s podatki, ki so zahtevani po standardu CVSS, in zapiše v podatkovno zbirko *National Vulnerability Databse* (NVD). [3] NVD je javno dostopna podatkovna zbirka ranljivosti, ki deluje pod okriljem organizacije *National Institute of Standards and Technology* (NIST). Standard CVSS je danes široko sprejet v svetu in je sestavni del mnogo orodij za obvladovanje tveganj [11, 12]. Podrobneje ga predstavljamo v poglavju 3.3.2.

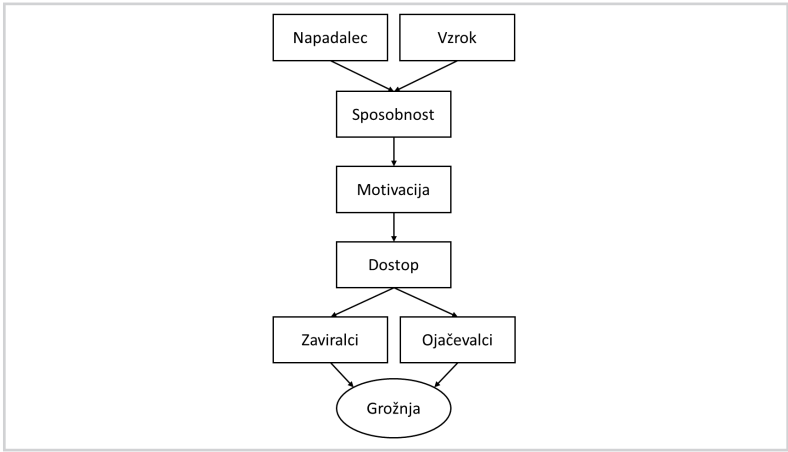
2.3 Grožnje

2.3.1 Definicija

Področje groženj je izjemno široko, zato v tem podpoglavju podajamo samo nekaj najpomembnejših informacij, ki so potrebne za celovito razumevanje proaktivnega pristopa pri ocenjevanju tveganj v IS.

Da bi natančno razumeli, kaj je grožnja, moramo najprej prepoznati posamezne elemente, ki jo sestavljajo. V pojmovniku publikacije *Internet Request For Comments (RFC)* najdemo naslednje definicije grožnje [33]:

[2] https://www.first.org/
[3] https://nvd.nist.gov/vuln



Slika 2.3

Grožnjo sestavlja več komponent. Med ključnimi so sposobnost, motivacija in priložnost za dostop do sistema. Razumevanje grožnje lahko izboljšamo z upoštevanjem dodatnih lastnosti in njihovih medsebojnih odvisnosti [34].

Definicija 2.2 (Opredelitve grožnje po RFC): Grožnja je kršitev varnosti, ki ob primernih okoliščinah, zmožnosti, dejanju ali dogodku lahko povzroči škodo. To pomeni, da je grožnja potencialna nevarnost, ki bi lahko izkoristila ranljivost. Grožnja je lahko namerna (npr. premišljena aktivnost s strani človeka) ali slučajna (npr. možnost nepravilnega delovanja sistema ali višja sila, kot so potres, požar ipd.).

V nadaljevanju se osredotočamo samo na namerne grožnje, saj so te danes najpogostejše in težko predvidljive. Človek, ki premišljeno ogroža IS (angl. Threat Agent), mora imeti ustrezne lastnosti. Poleg tega mora poiskati ali počakati na primerne okoliščine, da bi bil pri napadu lahko uspešen. Med ključnimi lastnostmi lahko izpostavimo sposobnost, motivacijo, vire in dostop do sistema, a so to le nekatere izmed mnogih, ki so potrebne za končni uspeh. Evans s sodelavci [34] na primer opredeljuje grožnjo kot kombinacijo napadalca, cilja napada, tipa napada in učinka. Slika 2.3 po drugi strani poleg ključnih izpostavlja tudi nekatere druge elemente. Pomembno je, da grožnjo razumemo in da jo znamo čim nazorneje opisati [33].

Naloga skrbnika IS je, da pravočasno prepozna možne grožnje in se nanje ustrezno pripravi. Neprestano se pojavljajo nove, zato moramo znanje in zavedanje groženj stalno dopolnjevati. Pri prepoznavanju in oceni groženj si lahko pomagamo z različnimi metodami in orodji. V grobem jih razdelimo v naslednje skupine:

- knjižnice in taksonomije znanih groženj [35],
- poznavanje splošnih vzorcev obnašanja napadalcev (npr. zakonitosti črnega trga),
- modeliranje možnih bodočih groženj,
- prepoznava groženj z uporabo inteligence groženj (angl. Threat Intelligence).

2.3.2 *Modeliranje groženj*

Grožnje predstavljajo premikajočo se tarčo. Napadalec vedno poizkuša biti korak pred žrtvijo, zato si ne moremo pomagati samo s preteklimi izkušnjami. Vsaj do neke mere pa lahko naredimo grožnje predvidljive, če si pomagamo z modeliranjem. Modeliranje groženj je proces popisovanja možnih napadalcev, njihovih napadov ter ocena možnih učinkov na IS. Pri tem je najučinkoviteje, da razmišljamo kot napadalec [20]. Primer modeliranja groženj so drevesa groženj in napadov (angl. Attack Trees, Threat Trees). Glavne koristi modeliranja groženj lahko strnemo v naslednjih točkah [36]:

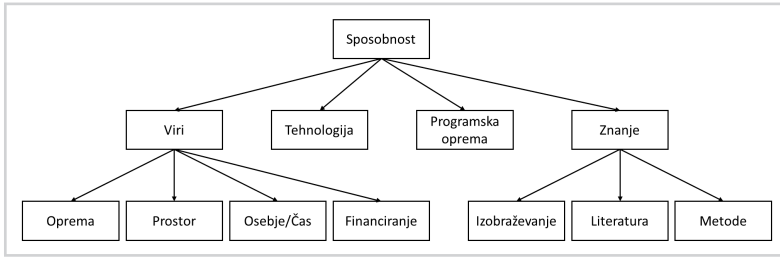
- sistematično prepoznamo in izberemo prioritetne grožnje,
- pomagamo si pri učinkovitem zmanjševanju tveganj,
- prepoznavamo povsem nove možne grožnje,
- prepoznamo ranljivosti v poslovni logiki in druge kritične ranljivosti, ki so vezane na jedro poslovanja.

Na slikah 2.4 in 2.5 so prikazane nekatere komponente groženj, ki so nam lahko v pomoč pri prepoznavanju in opredeljevanju groženj.

2.3.3 *Črni trg*

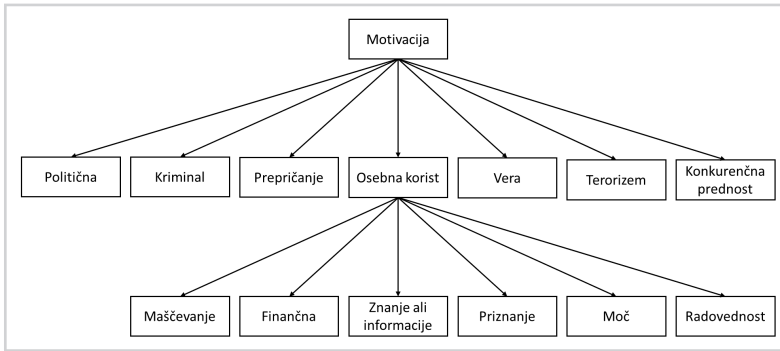
Pri razumevanju groženj so nam v pomoč tudi informacije o tem, kako deluje črni trg ranljivosti. Model obstoječega črnega in belega trga s predstavitvijo vseh ključnih akterjev in procesov v celotnem življenjskem ciklu ranljivosti podaja Frei s sodelavci [23]. Osnovno delovanje obeh trgov je ponazorjeno in opisano na sliki 2.6.

Nejasno ostaja obnašanje udeležencev črnega trga v odnosu do belega trga. Zanimivo je predvsem vprašanje, ali posamezniki tipično prodajajo ranljivosti le na črnem trgu ali praviloma iščejo najugodnejšo ponudbo in s tem sočasno delujejo na obeh trgih.



Slika 2.4

Sposobnost napadalca je odvisna od več komponent. Z obsežnejšimi viri, tehnologijo, opremo in znanjem kot napadalec razpolaga, večjo grožnjo predstavlja (povzeto po [33]).

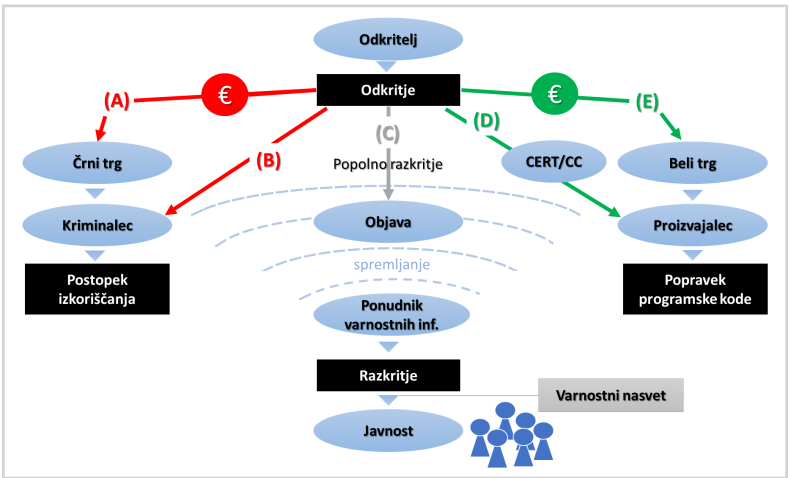


Slika 2.5

Viri motivacije za napad na IS so lahko zelo različni in večplastni. Pomembno je, da prepoznamo vse tiste, ki so dovolj verjetni, da se pravočasno pripravimo nanje (povzeto po [33]).

Tako kot večina današnje ekonomije tudi črni trg posluje preko interneta, preko vsem dostopnih spletnih strani. Deluje nestalno, se pogosto seli in se občasno oglašuje [37]. Vprašanje je, kako lahko proaktivno delujemo proti črnemu trgu. Radianti s sodelavci [38] je s simulacijo na dinamičnem modelu preverila možnosti uničenja črnega trga ranljivosti in ugotovila, da bi bilo to možno doseči z ostrejšimi kaznimi. Ni znano, v kolikšni meri kazni v resnici vplivajo na trgovanje, saj ni verodostojnih podatkov o obsegu tovrstnih transakcij.

Črni trg se vse bolj profesionalizira. Pojavljajo se povsem novi poslovni modeli trgovanja z vedno bolj dovršenimi produkti za izkoriščanje, s tem pa nove oblike groženj, ki si jih doslej nismo niti predstavljali. Danes na primer napadalcem za določene tipe napadov (npr. razširjanje škodljive kode) ni več potrebno iskati ranljivosti. Na črnem trgu lahko kupijo že izdelan programski paket za izkoriščanje, ga namestijo na svoj strežnik in že izvajajo napade (angl. Exploit-as-a-Service) [39].



Slika 2.6

Motivacija za odkrivanje ranljivosti je zelo različna. A in B možnosti sta povezani z zlonamernimi nameni, zaslužkom ali političnimi motivi. Razlog za pot C je lahko slava, priznanje, radovednost, izziv ipd. D predstavlja nesebičen, dobrodelen odnos do skupnosti. C, D in E predstavljajo pritisk na neodzivne proizvajalce programske opreme (povzeto po [23]).

2.3.4 *Napredne vztrajne grožnje*

Omenimo še zelo pomembno skupino groženj z originalnim poimenovanjem *Advanced Persistent Threats* (APT). Usmerjene so v informacijsko premoženje visoke vrednosti. Gre za najnevarnejše grožnje, ki danes obstajajo. Med osnovnimi cilji groženj APT so kraja intelektualne lastnine, dostop do občutljivih podatkov, dostop do strateških informacij ipd. APT grožnje izvajajo visoko usposobljeni, motivirani in dobro plačani posamezniki. Pri teh grožnjah gre tipično za izkoriščanje ranljivosti ničtega dne, da žrtev napad zelo težko zazna. Izkoriščanje praviloma traja daljše obdobje (več mesecev ali celo let) z nizko stopnjo intenzivnosti [40].

2.3.5 *Inteligenca groženj*

Sodobne tehnologije obetajo nove pristope pri zaznavanju groženj. Povsem novo je področje vzpostavljanja inteligence groženj (angl. Threat Intelligence). Danes obstajajo tehnologije, ki omogočajo zbiranje, obdelavo in izmenjavo velikih količin podatkov, kar lahko s pridom uporabimo pri zagotavljanju večje varnosti (npr. Big Data).

Cilj prizadevanj organizacij, ki so dejavne na področju varnosti, je avtomatizirati varnost. Ena izmed priložnosti za izboljšanje varnosti IS je izmenjava informacij o spletnih grožnjah med zainteresiranimi podjetji, ki si med seboj zaupajo. Pod okriljem

organizacije MITRE je nastalo več standardov, ki podpirajo takšno izmenjavo (STIX, TAXII in drugi) [41].

2.4 Tveganje

2.4.1 Definicija

Ena najsplošnejših definicij tveganja se glasi: "Tveganje je nezaželena podskupina izidov v nizu negotovih rezultatov" (Cornelius Keating). Raziskave kažejo, da enotna opredelitev pojma "tveganje", ki bi veljala za vsa področja hkrati, ne obstaja. Samo na področju informacijskih sistemov obstaja preko 200 različnih metod in standardov, ki obravnavajo to področje [42]. Grožnje v IS so tako številne, da je realno nemogoče aktivno delovati na vseh možnih področjih. V nadaljevanju zato navajamo le nekaj definicij, ki so pomembne za našo raziskavo.

Skupni imenovalec metod za obvladovanje tveganj v IS je zagotavljanje varnosti informacijskega premoženja (angl. assets). Slednje obsega strojno ter programsko opremo, mrežno infrastrukturo in tudi ljudi, ki izvajajo svoje vloge v okviru IS. Vsako izmed komponent IS moramo z vidika morebitnega nastanka škode oceniti v zvezi z zaupnostjo (angl. Confidentiality), razpoložljivostjo (angl. Availability) in celovitostjo (angl. Integrity).

Pomembni mednarodni varnostni standardi za področje informacijskih tehnologij opredeljujejo tveganje na naslednje načine:

Definicija 2.3 (Tveganje po standardu ISO/IEC 13335 [42]): Tveganje je potencial, da bo dana grožnja izkoristila ranljivosti sredstva ali skupine sredstev in tako povzročila organizaciji škodo.

Definicija 2.4 (Tveganje in upravljanje s tveganji po standardih NIST [43]): Tveganje je mera ogroženosti entitete s strani potencialne okoliščine ali dogodka in je tipično funkcija (i) škodljivega učinka, ki bi nastal ob tem, ter (ii) verjetnosti pojavitve tega dogodka.

Upravljanje s tveganji vključuje analize groženj in ranljivosti ter načrtovanje in vzpostavljanje varnostnih ukrepov.

Standard ISO/IEC 27001 posebej ne navaja definicije pojma "tveganje", ampak opredeljuje postopek prepoznave tveganj.

Definicija 2.5 (Postopek prepoznave tveganj po ISO/IEC 27001 standardu [42]):

1. Prepoznava premoženja v okviru IS.
2. Prepoznava groženj za posamezno premoženje.
3. Prepoznava ranljivosti, ki bi jih lahko izkoristile grožnje.
4. Prepoznava učinkov na zaupnost, celovitost in razpoložljivost premoženja.

V vseh omenjenih definicijah se prepletajo naslednje komponente: grožnja (angl. *Threat*), ranljivost (angl. *Vulnerability*) in učinek (angl. *Impact*). Njihove medsebojne relacije so prikazane na sliki 2.7, enostavno pa jih povzema osnovna opredelitev tveganj v IS, ki je podana v [33].

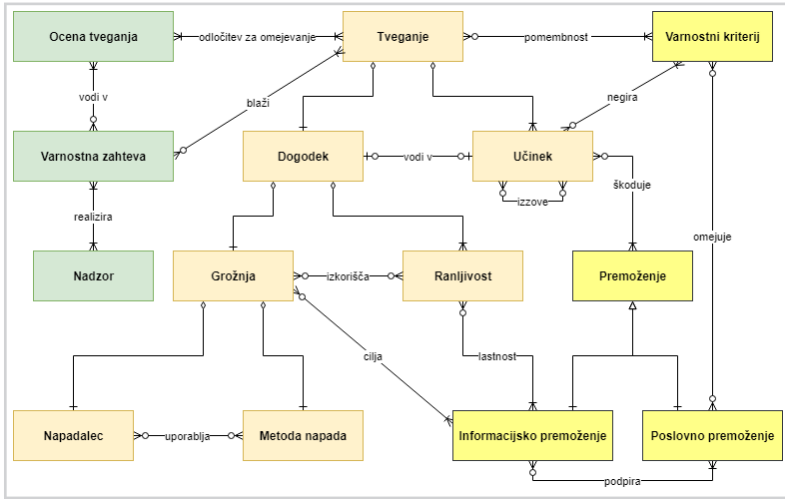
Definicija 2.6 (Osnovna opredelitev tveganja v IS): Tveganje je verjetnost, da bo napadalec izkoristil ranljivost v informacijskem sistemu in s tem povzročil škodo na informacijskem premoženju.

$$Tveganje = grožnja \times ranljivost \times učinek$$

Pojma ranljivost in grožnja sta zelo široka, zato večkrat prihaja do napačnih tolmačenj, razumevanja ali celo do njunega zamenjevanja. Ranljivost je v prvi vrsti lastnost sistema. Točneje povedano je to kakršna koli pomanjkljivost na izpostavljenem delu sistema (angl. *Attack Surface*), ki jo lahko izkoristijo posamezniki z zelo različnimi interesi. Nekaterih odkritih ranljivosti včasih tudi ni možno izkoristiti ali ni pravega interesa za njihovo izkoriščanje. Za dejansko izkoriščanje je namreč potrebno razviti postopek izkoriščanja (angl. *Exploit*), ki pogosto ni preprost [44].

Grožnjo po drugi strani povezujemo z napadalcem sistema (angl. *Attacker*), ki je zmožen izkoristiti eno ali več ranljivosti sistema. Predpogoj za uspešno izvedbo napada je obstoj postopka izkoriščanja, ki ga je možno izvesti preko ranljive komponente v sistemu (angl. *Attack Vector*). Ob tem je nepomembno, ali je napadalec postopek izkoriščanja razvil sam ali ga je od nekod pridobil [45].

Odkrivanje ranljivosti in ravnanje z njimi je poseben proces, v katerem sodelujejo



Slika 2.7

Predstavitev področja varnostnih tveganj v IS z UML razrednim diagramom (angl. Information System Security Risk Management domain model, ISSRM) [42].

posamezniki in skupine z nasprotujočimi si interesi. Za razumevanje procesa moramo dobro poznati tako zakonitosti ranljivosti kot tudi obnašanje vpletenih akterjev. V nadaljevanju zato podrobno predstavljamo obe področji.

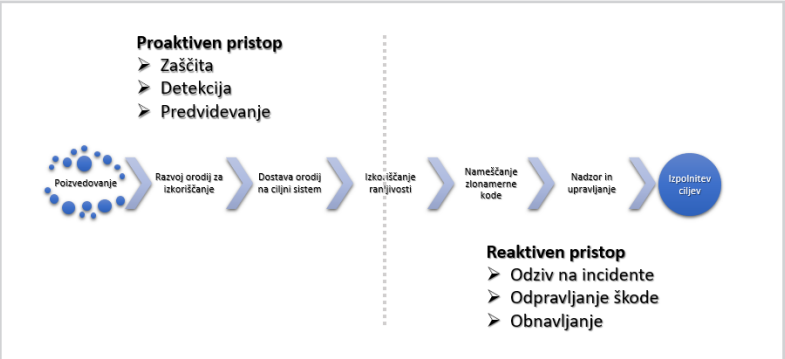
2.4.2 Ocena tveganj

Ocenjevanje tveganj poteka v več korakih. Skladno z definicijo tveganja 2.6 je potrebno izvesti naslednje korake:

1. prepoznati ključno premoženje v okviru IS,
2. prepoznati in oceniti grožnje,
3. prepoznati in oceniti ranljivosti,
4. oceniti tveganje.

Ranljivosti so bolj otipljive od groženj, čeprav jih je pogosto zelo težko odkriti. Prisotne so namreč v sistemu, ki se nahaja pred nami, in jih lahko z različnimi postopki odkrijemo ter odstranimo (nevtraliziramo). S tem zagotovimo aktivno varnost sistema.

Grožnje so po drugi strani skrite in o njih lahko le domnevamo. Povsem zgrešeno je razmišljanje, da se vsega o grožnjah lahko naučimo iz preteklih varnostnih incidentov.



Slika 2.8

Ponazoritev nastajanja in stopnjevanja grožnje. Povzeto po Cyber Kill Chain, Lockheed-Martin [46].

Takšen pristop povezujemo z reaktivnim (pasivnim) zagotavljanjem varnosti. Če želimo biti pri varovanju sistema proaktivni oz. korak pred napadalci, moramo grožnje in z njimi povezane aktivnosti predvidevati v čim zgodnejši fazi njihovega razvoja (glej sliko 2.8). Ocena tveganj bo torej točnejša in obvladovanje učinkovitejše, če ob izvedbi postopka proučimo tako obstoječe kot tudi bodoče grožnje.

V osnovi ločimo naslednje pristope obvladovanja tveganj [7]:

- *Reaktivni pristop*: gre za pasiven pristop, kjer se na varnostne incidente odzovemo, ko se pojavijo. Med odpravljanjem posledic odkrivamo, kako so napadalci uspeli prodreti v sistem, in nato vzpostavimo primerne mehanizme, da se to ne bi več ponovilo.

- *Aktivni pristop*: namesto da bi čakali na morebitne napadalce, da nam pokažejo ranljive točke našega IS, jih z ustreznimi metodami in orodji poiščemo sami. Primer takšnega pristopa je uporaba orodij za odkrivanje obstoječih ranljivosti (angl. Vulnerability Scanners).

- *Proaktivni pristop*: z različnimi metodami poizkušamo prehiteti napadalce in predvideti njihove bodoče aktivnosti (npr. načrtovanje vdorov, odkrivanje ranljivosti, razvijanje postopkov izkoriščanja). Poleg odkrivanja ranljivosti je torej pomembno tudi predvidevanje, katere ranljivosti bodo napadalci v praksi izkoriščali.

3

Modeli obvladovanja tveganj

3.1 Uvod

Tradicionalni pristopi pri ocenjevanju tveganj se praviloma naslanjajo na statistične podatke, ki se zbirajo za desetletja nazaj. V primeru IS teh enostavno ni, saj tehnologija zastari, še preden bi bili ustrezni podatki lahko koristni. Posledica tega je, da na IT področju prevladujejo kvalitativne ocene tveganj [1, 47].

Nekateri računalniško podprti kazalniki, ki omogočajo kvantitativne ocene tveganj, že obstajajo [48]. Na podatkih podatkovne zbirke CVE je osnovan kazalnik dnevne izpostavljenosti izbrane ranljivosti nekega produkta (angl. *Daily Vulnerability Exposure*, DVE)[49]. To lahko uporabimo za izračun tveganja produkta v izbranem preteklem obdobju (angl. Reactive Risk Management). Na podatkih iste zbirke je opredeljen indeks ranljivosti (angl. *Vulnerability Index*), ki se lahko uporabi pri sprotnem izračunavanju tveganja (angl. Active Risk Management). Trček [1] je omenjena pristopa nadgradil z modelom na osnovi sistemske dinamike (angl. System Dynamics Model), s katerim je možno predvidevanje odkrivanja ranljivosti v bližnji prihodnosti (angl. Proactive Risk Management). Zaradi vse večje prepletenosti IS in vseprisotnosti informacijskih tehnologij je potreba po proaktivnem obvladovanju nastajajočih tveganj vse večja. Med možnimi pristopi je tudi uporaba tehnik inoviranja. Ključno postaja predvidevanje groženj, ki še ne obstajajo, a se lahko pojavijo v prihodnosti [50].

Ob tem velja poudariti, da lahko proaktivni pristop uporabimo na različnih stopnjah življenjskega cikla IS. V času razvoja programske opreme usmerjamo napore v preprečevanje nastajanja ranljivosti. Zato potrebujemo orodja za predvidevanje ranljivih programskih komponent in skupno oceno vseh ranljivosti v programski opremi. V obdobju, ko so programske komponente že nameščene, proaktivni pristop povezujemo s predvidevanjem izkoriščanja ranljivosti. Ob mnogih ranljivostih v IS moramo pri odstranjevanju dati prednost tistim, ki so nevarnejše. Upoštevati moramo tako učinek izkoriščanja ranljivosti kot tudi verjetnost njenega izkoriščanja.

V nadaljevanju podajamo pregled metod za obvladovanje tveganj v vrstnem redu, kot si sledijo obdobja v razvoju IS.

3.2 Obvladovanje tveganj v času razvoja programske opreme

3.2.1 Vulnerability Discovery Models (VDM)

Modeli za odkrivanje ranljivosti v programski opremi (angl. Vulnerability Discovery Model, VDM) so se izoblikovali, da bi lahko ponudniki pred začetkom trženja produk-

ta ocenili skupno pričakovano število ranljivosti ter njihovo časovno odkrivanje. S tem bi lahko pravočasno rezervirali potrebne vire za njihovo odpravljanje. Ideja za oblikovanje modelov VDM izhaja neposredno iz statističnih modelov SRM (angl. Software Reliability Model) za odkrivanje programskih napak (t. i. bugov). Izkazalo se je, da je narava ranljivosti precej drugačna od programskih napak. Zato modeli, osnovani na SRM, niso primerni za predvidevanje odkrivanja ranljivosti [51, 52].

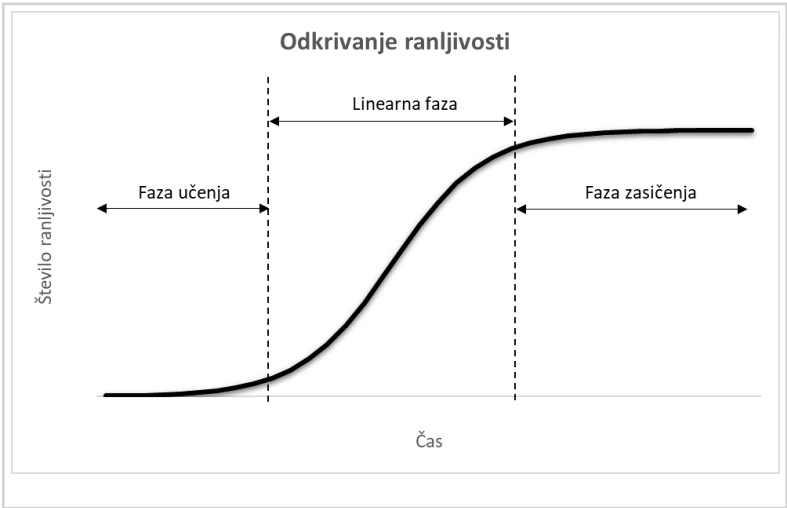
Predlaganih je bilo več modelov za predvidevanje odkrivanja ranljivosti v okviru posamezne aplikacije [53–56]. Med vsemi modeli se pri predvidevanju ranljivosti najbolje odreže AML logistični model [52, 57]. Alhazmi s sodelavci [54] zanj definira pojem gostote ranljivosti (angl. Vulnerability Density), ki predstavlja število ranljivosti na število programskih vrstic. Uporablja se pri oceni pričakovanega skupnega števila vseh ranljivosti v programskem produktu.

Model AML predpostavlja, da je stopnja sprememb v kumulativnem številu odvisna od dveh dejavnikov, in sicer od uvodne privlačnosti produkta ter zasičenja v zaključni fazi (glej sliko 3.1). Predpostavke lahko strnemo v naslednjih točkah [54]:

- ob vstopu produkta na trg zanj še ni veliko zanimanja, kar se kaže tudi v majhnem številu odkritih ranljivostih,
- s popularnostjo izdelka narašča interes za odkrivanje ranljivosti, kar se kaže v večjem številu odkritij,
- ko produkt ni več zanimiv za širši krog uporabnikov, upade tudi interes za odkrivanje ranljivosti, saj je odkritij vse manj,
- število novoodkritih ranljivosti upade tudi zaradi stalnega odpravljanja pomanjkljivosti s strani ponudnika.

Dobre napovedne sposobnosti tega modela potrjuje raziskava odkrivanja ranljivosti na spletnih strežnikih Apache in IIS [58]. Ista raziskava tudi ugotavlja, da je tržni delež produkta eden ključnih dejavnikov, ki spodbuja zainteresirane posameznike k odkrivanju ranljivosti. Večji tržni delež namreč omogoča večji zaslužek zainteresiranim posameznikom, ki delujejo tako v okviru belega kot črnega trga.

Obstoječi VDM modeli so preverjeni na podatkih javno dostopnih podatkovnih zbirk CVE in NVD. Neodvisni preizkusi obstoječih modelov VDM, pregled njihovih pomanjkljivosti in nekateri predlogi za izboljšanje so podani v [59–61].



Slika 3.1
AML časovni model.

3.2.2 Predvidevanje ranljivih programskih komponent

V zaključni fazi razvoja programske opreme je zelo pomembno testiranje in odstranjevanje ranljivosti. Zaradi omejitve razpoložljivih virov moramo določiti prioritete, katere programske komponente potrebujejo večjo pozornost v zvezi z zagotavljanjem varnosti. Potrebujemo torej orodja, s katerimi so možna tovrstna predvidevanja.

Na osnovi podatkov v zbirki ranljivosti NVD je možno z visoko verjetnostjo predvidevati, katere programske komponente bodo ranljive. Osnova za predvidevanja so lastnosti posameznih programskih komponent in podatki o predhodnih različicah. Posamične ranljivosti na starejših različicah programske komponente večinoma ne pomenijo, da bodo ranljive tudi novejše različice. Po drugi strani je podobnost funkcije z drugimi ranljivimi funkcijami dober znak, da bodo ranljivosti prisotne [62].

Predvidevanje ranljivih programskih komponent je možno opraviti tudi z uporabo strojnega učenja. Osnovo takšnega modela predstavlja tekstovno rudarjenje izvorne programske kode. Izračunane frekvence posameznih izrazov v izvorni kodi se v nadaljevanju uporabijo pri ugotavljanju, katere programske komponente so bolj verjetno ranljive [63]. Obstaja še več modelov za predvidevanje ranljivih programskih komponent, ki so osnovani na drugih metrikah programske kode [63].

3.2.3 Common Weakness Enumeration (CWE)

V času razvoja programske opreme lahko naredimo največ za zagotavljanje proaktivne varnosti IS. Ključno je širjenje znanja in razumevanja pomena pomanjkljivosti v programski kodi, saj s tem najučinkoviteje preprečujemo njihovo nastajanje. K temu je velik delež prispeval projekt *Common Weakness Enumeration* (CWE),[1] ki je nastal pod okriljem organizacije MITRE. Njegov osnovi cilj je ustvariti in vzdrževati katalog tipičnih slabosti programske opreme, ki lahko vodijo v ranljivosti. Projekt v osnovi prispeva k boljšemu razumevanju pomanjkljivosti v programski opremi. Sočasno s tem katalog predstavlja osnovo za razvoj avtomatiziranih programskih orodij za prepoznavanje, odpravljanje in preprečevanje teh pomanjkljivosti.

Ker je na stopnji razvoja pomanjkljivosti lahko zelo veliko, jim je treba določiti prioriteto za odstranjevanje. Zato so nad katalogom CWE zgradili sistem *Common Weakness Scoring System* (CWSS)[2], ki omogoča kvantitativno ocenjevanje pomanjkljivosti. CWSS sistem je v osnovi precej podoben sistemu CVSS, ki smo ga na kratko predstavili v poglavju 2.2.5. Razlika med njima je v tem, da se CVSS uporablja za opisovanje že odkritih ranljivosti, CWSS pa za ocenjevanje pomanjkljivosti v programski kodi na stopnji razvoja.

3.3 Obvladovanje tveganj v času uporabe programske opreme

3.3.1 Security Content Automation Protocol (SCAP)

Od trenutka, ko je programska oprema nameščena v IS in predana v uporabo, obvladovanje tveganj postane kritično [14]. Pojavijo se lahko grožnje, ki povzročijo škodo. Varnost lahko zagotavljamo le s stalnim preverjanjem oz. odkrivanjem ranljivosti ter njihovim odstranjevanjem, kar pa je lahko zelo zahtevno in časovno potratno opravilo. Zahvaljujoč podatkom o ranljivostih v obstoječih podatkovnih zbirkah je ta postopek možno avtomatizirati.

Pod okriljem NIST je v letu 2008 nastal protokol SCAP[3] (angl. Security Content Automation Protocol), ki povezuje več standardov. Njegov sestavni del sta tudi standarda CVE in CVSS. Osnovni namen SCAP je povezati različne iniciative ter s tem

[1] http://cwe.mitre.org/
[2] https://cwe.mitre.org/cwss
[3] https://scap.nist.gov/

zagotoviti avtomatizirano upravljanje ranljivosti ter merjenje in ocenjevanje skladnosti informacijskih sistemov, ki so že v redni uporabi.

Med različnimi orodji, ki so se pojavila kot rezultat te iniciative, so tudi skenerji ranljivosti (angl. Vulnerability Scanners). Uporabljajo se tako za prepoznavanje ranljivosti v IS kot tudi za podajanje ocen njihove resnosti grožnje. Ocene se praviloma uporabljajo tudi pri določanju vrstnega reda pri nevtralizaciji ranljivosti. Osnova za delovanje skenerjev ranljivosti so javno dostopni podatki o ranljivostih v zbirkah CVE in NVD.

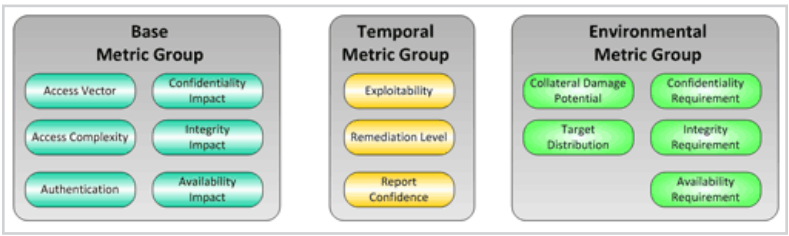
Skenerji ranljivosti na osnovi protokola SCAP so danes nepogrešljiva orodja za obvladovanje tveganj v večini organizacij. Predstavljajo proaktiven pristop pri zagotavljanju varnosti IS, saj omogočajo prepoznavo in nevtralizacijo ranljivosti, preden pride do njihovega izkoriščanja [7, 12].

3.3.2 *Common Vulnerability Scoring System (CVSS)*

Upravitelj IS lahko z uporabo avtomatiziranega skenerja dokaj hitro odkrije množico ranljivosti. V drugem koraku jih je nato treba odstraniti, in sicer v takšnem vrstnem redu, da čim hitreje zmanjša tveganje izkoriščanja. Ranljivostim je torej treba dodeliti ocene, ki predstavljajo raven njihove grožnje. V praksi se je pojavilo več načinov kvalitativnega in kvantitativnega ocenjevanja. Med vsemi se je najbolj uveljavil odprti standard CVSS. Tega smo na kratko predstavili že v podpoglavju 2.2.5, ko smo opisali javno dostopno podatkovno zbirko o ranljivostih NVD. Omenili smo ga tudi v podpoglavju 3.3.1, in sicer kot sestavni del protokola SCAP oz. orodij za avtomatizirano upravljanje ranljivosti, ki so zasnovana na tem protokolu [11, 12].

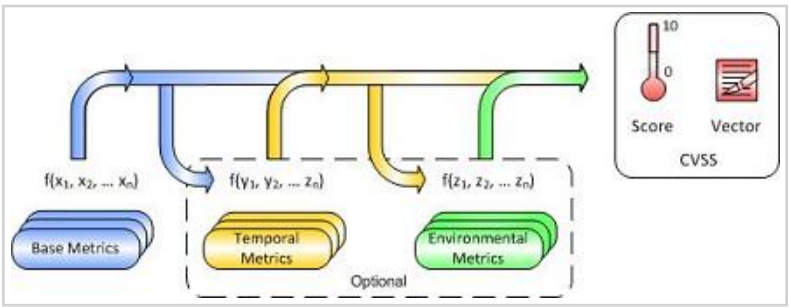
CVSS opredeljuje enoten postopek ocenjevanja ranljivosti z opisnimi (kvalitativnimi) atributi. Ti so razdeljeni v tri skupine: osnovni (*Base*), časovni (*Temporal*) in okoljski atributi (*Environmental*) (glej sliko 3.2). Pri vsaki skupini atributov so opredeljene enačbe za izračun kvantitativne ocene resnosti ranljivosti (angl. Severity Score). To pomeni, da ima vsaka ranljivost lahko tri ocene: CVSS Base, CVSS Temporal in CVSS Environmental Score. Ocene so opredeljene na intervalu od 0 do 10, pri čemer višja ocena pomeni resnejšo ranljivost. CVSS Base ocena se izračuna na osnovi enačb v definiciji 3.1. Spremenljivke v enačbah predstavljajo atribute ranljivosti in imajo kvantitativne vrednosti. Postopek izračuna ostalih ocen CVSS je prikazan na sliki 3.3. Na spletni strani NVD je na voljo tudi spletni kalkulator. [4]

[4] https://nvd.nist.gov/vuln-metrics/cvss/v2-calculator



Slika 3.2

Shema atributov po standardu CVSSv2. Atributi so razdeljeni v tri skupine: osnovni (Base), časovni (Temporal) in okoljski (Environmental)[64].



Slika 3.3

Postopek izračuna ocen CVSS. Ocena CVSS Base vedno obstaja in je na voljo v zbirki NVD. CVSS Temporal se izračuna na osnovi ocene CVSS Base, CVSS Environmental pa na osnovi ocene CVSS Temporal[64].

Definicija 3.1 (CVSS Base Score):

$$BaseScore = (((0.6 * Impact) + (0.4 * Exploitability) – 1.5) * f(Impact))$$

$$Impact = 10.41 * (1 – (1 – ConfImpact) * (1 – IntegImpact) * (1 – AvailImpact))$$

$$Exploitability = 20 * AccessVector * AccessComplexity * Authentication$$

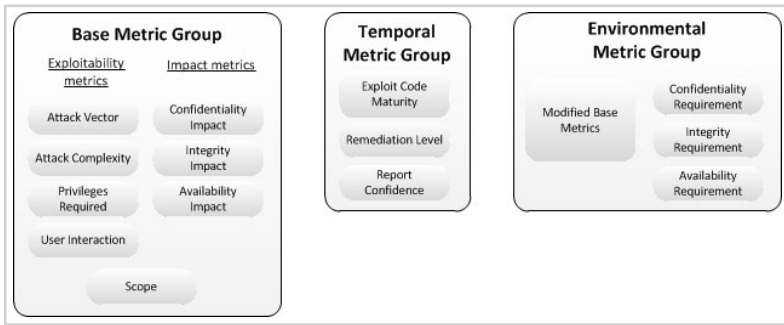
$$f(impact) = 0 \quad if\ Impact = 0, \quad 1.176\ otherwise$$

Vrednosti *CVSS Base* atributov in ocen so javno dostopne v podatkovni zbirki NVD in predstavljajo notranje lastnosti ranljivosti. Vrednosti atributov so predstavljene v obliki vektorja CVSS (primer: `AV:L/AC:M/Au:N/C:C/I:N/A:N`), ki je sestavljen iz niza znakov. Časovni atributi predstavljajo informacije o okoliščinah v zvezi z ranljivostjo, ki se praviloma spreminjajo skozi čas (npr. informacija o obstoju postopkov izkoriščanja, informacija o postopku nevtralizacije). Teh vrednosti v zbirki NVD ni. Lahko jih določijo varnostni analitiki in ponudniki programske opreme, vendar so

Tabela 3.1

Opredelitev kvalitativne ocene ranljivosti na osnovi ocene CVSS Base.

Resnost ranljivosti	*CVSS Base ocena*
LOW	0,0 – 3,9
MEDIUM	4,0 – 6,9
HIGH	7,0 – 10,0



Slika 3.4

Shema atributov po standardu CVSSv3. Atributi so podobno kot pri predhodnih različicah standarda razdeljeni v tri skupine: osnovni (Base), časovni (Temporal) in okoljski (Environmental)[65].

takšni podatki v praksi redki. Okoljski atributi predstavljajo kontekstne informacije o okolju, kjer je nameščena programska oprema z opisano ranljivostjo. Takšne podatke zato lahko zagotovijo samo lastniki IS in niso na voljo v zbirki NVD. Kljub temu da standard CVSS ne zagotavlja kvalitativne ocene ranljivosti, jo je naknadno opredelil NIST in jo objavil v zbirki NVD. Kvalitativna ocena je odvisna od kvantitativne ocene in se določi po tabeli 3.1 [10, 64].

V juniju 2015 je izšla tretja različica standarda CVSS. Shema atributov po tej različici je prikazana na sliki 3.4. Osnovni koncept izračunavanja ocen CVSS se z novo različico ni bistveno spremenil, nekoliko pa so se spremenili atributi. V novi različici standarda je opredeljen tudi postopek določanja kvalitativne ocene CVSS. Kljub novemu standardu se ranljivosti v zbirki NVD še naprej objavljajo tudi v različici CVSSv2, saj je objavljanje podatkov v različici CVSSv3 zaživelo šele nedavno.

CVSS je torej univerzalen sistem za kvantitativno ocenjevanje resnosti groženj, ki jih predstavljajo ranljivosti v programski opremi. Eden njegovih osnovnih namenov uporabe je določanje prioritet pri odstranjevanju ranljivosti glede na tveganje, ki ga predstavljajo za IS [9, 10]. Na njegovi osnovi je bilo kasneje razvitih več različnih

modelov ocenjevanja tveganj [1, 66, 67]. Standard je danes široko sprejet v svetu in v mnogih organizacijah ga uporabljajo kot ključno orodje za obvladovanje tveganj v povezavi s programsko opremo.

3.3.3 *Vulnerability Rating and Scoring System (VRSS)*

Prednost kvalitativne ocene ranljivosti je v tem, da je človeku bližja, saj enostavneje loči nevarne od manj nevarnih ranljivosti. Verjetno je to glavni razlog, da so kvalitativno oceno naknadno opredelili in dodali tudi zbirki NVD. Kljub vsemu nekateri raziskovalci s to oceno, ki se določi na osnovi ocene CVSS Base, niso zadovoljni. Frekvence posameznih kvalitativnih ocen CVSS (LOW, MEDIUM, HIGH) ne sledijo naravni porazdelitvi. Raziskave na podlagi podatkov zbirke NVD ter konkretni podatki na spletni strani NIST dokazujejo, da močno prevladujejo ranljivosti iz skupin HIGH in MEDIUM. [10][5]

Liu s sodelavci [68] zato predlaga izboljšavo sistema za ocenjevanje resnosti ranljivosti. Po njihovem prepričanju mora kakovosten sistem ocene enakomerno razpršiti. Predstavili so merilni sistem *Vulnerability Rating and Scoring System* (VRSS), ki je kombinacija kvalitativnega in kvantitativnega pristopa. Z njim dosežejo boljšo porazdelitev kvalitativnih ocen. Na podlagi podatkov v zbirki NVD so ugotovili, da prevladujejo ranljivosti iz razreda MEDIUM, medtem ko sta razreda HIGH in LOW manj zastopana.

Izračun ocene VRSS poteka v dveh korakih. V prvem ocenimo resnost ranljivosti po ocenjevalni lestvici (LOW, MEDIUM, HIGH). Kvalitativno oceno določimo na osnovi CVSS atributov *Confidentiality Impact*, *Integrity Impact* in *Availability Impact* skladno s tabelo 3.2. V drugem koraku se na osnovi enačb v definiciji 3.2 izračuna kvantitativna VRSS ocena. Vrednost *Impact score* se določi na osnovi *Kvalitativne ocene* ranljivosti v tabeli 3.2. Parameter *Exploitability* se izračuna na osnovi vrednosti CVSS atributov *AccessVector*, *AccessComplexity* in *Authentication* (3.2).

Sistem VRSS z uporabo atributov CVSS Impact doseže naravno porazdelitev frekvenc kvalitativnih ocen VRSS v zbirki NVD. Poleg tega VRSS dosega večjo raznolikost kvantitativnih ocen od sistema CVSS. Z osnovnimi atributi CVSS sta namreč možna 702 različna vektorja CVSS. V primeru sistema CVSS je na osnovi teh vektorjev možnih samo 68 različnih kvantitativnih ocen, v primeru VRSS pa kar 207. Čeprav

[5] https://nvd.nist.gov/general/nvd-dashboard

Tabela 3.2

Preslikovalna tabela atributov CVSS Impact v kvalitativno oceno in VRSS Impact score.

Vrednosti CVSS Impact atributov	*Kvalitativna ocena*	*Impact score*
Vsi atributi imajo vrednost COMPLETE.	HIGH	9
En atribut PARTIAL, dva COMPLETE.	HIGH	8
En atribut NONE, dva COMPLETE.	HIGH	7
En atribut COMPLETE, dva PARTIAL.	HIGH	6
NONE, PARTIAL in COMPLETE.	MEDIUM	5
En atribut COMPLETE, dva NONE.	MEDIUM	4
Vsi atributi imajo vrednost PARTIAL.	MEDIUM	3
En atribut NONE, dva PARTIAL.	MEDIUM	2
En atribut PARTIAL, dva NONE.	LOW	1
Vsi atributi imajo vrednost NONE.	LOW	0

sistem VRSS po teh merilih dosega višjo kakovost, avtorji ne podajajo dokazov, da so ocene VRSS točnejše od ocen CVSS.

Definicija 3.2 (VRSS Score):

$$VRSS\ Score = Impact\ Score + Exploitability\ Score$$

$$Exploitability = 2 \times AccessVector \times AccessComplexity \times Authentication$$

Sistem VRSS je bil kasneje nadgrajen z uporabo vrednosti CWE (Common Weakness Enumeration), ki ranljivosti dodatno loči po tipu. Novi model dosega še večjo raznolikost ocen, a ostaja nejasnost glede točnosti ocen [69].

3.4 Problemi določanja prioritet ranljivostim za odstranjevanje

Razkrite ranljivosti gredo v svojem življenjskem ciklu preko več faz, pri čemer so ključni mejniki naslednji: razkritje, možnost izkoriščanja in možnost odstranitve [30]. Skladno z okoliščinami, ki so značilne za posamezne faze, se ves čas spreminja tudi raven njihove grožnje. Najnevarnejši je čas od trenutka pojavitve delujočega postopka izkoriščanja ranljivosti do trenutka razpoložljivosti postopka za njeno odstranitev [23].

Ranljivosti, za katere na črnem trgu že obstajajo delujoči postopki izkoriščanja, si torej zaslužijo prednostno obravnavo, saj je verjetnost njihovega izkoriščanja zelo velika. Kljub vsemu si tudi te ranljivosti med seboj niso enake in glede na oceno CVSS predstavljajo različno raven grožnje [12].

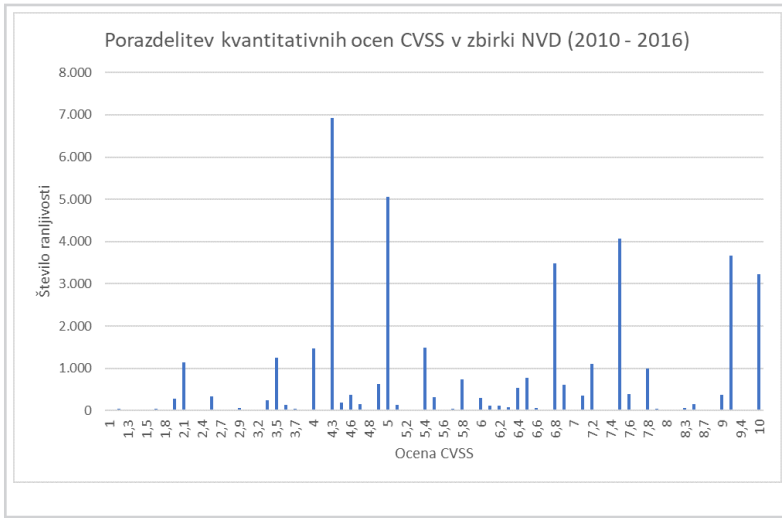
Kaj pa odkrite ranljivosti, za katere še ni dokazov o obstoju delujočih postopkov izkoriščanja? Dvomimo, da si kdo lahko privošči čakanje do trenutka, ko izkoriščanje postane možno. Zato je tudi te ranljivosti treba odpraviti, in sicer v vrstnem redu, ki najučinkoviteje zmanjšuje varnostno tveganje. Posledično to pomeni, da vprašanje o tem, katera metoda razvrščanja ranljivosti po prioriteti je najučinkovitejša, naslavljamo na ravni vseh odkritih ranljivosti v IS.

Pri določanju prioritet ranljivostim za odstranjevanje se najpogosteje uporabljajo ocene CVSS. Žal so raziskave pokazale, da ima sistem CVSS kljub številnim koristim tudi precej pomanjkljivosti. Prva je porazdelitev osnovne kvantitativne ocene CVSS (CVSS Base Score). V prvih letih obstoja zbirke NVD se je izkazalo, da v praksi prevladujejo ranljivosti s samo dvema različnima ocenama (79 %) [10]. Analizo smo ponovili na podatkih zadnjih let (obdobje 2010 – 2016). Rezultati na sliki 3.5 kažejo, da se je porazdeljenost ocen CVSS nekoliko izboljšala. Dve tretjini ranljivosti v NVD ima eno od sedmih različnih ocen CVSS izmed 68 možnih. Po drugi strani je porazdeljenost kvalitativnih ocen CVSS še vedno slaba (slika 3.6). Prevladujejo ranljivosti z ocenami MEDIUM in HIGH, ranljivosti z oceno LOW pa je občutno manj (manj kot 10 %).

Druga pomanjkljivost je slaba razpršenost ocen CVSS. Kakovosten kazalnik naj bi imel čim bolj razpršene vrednosti, v primeru CVSS pa se več vektorjev (naborov atributov) na osnovi predpisanih formul preslika v iste končne ocene CVSS [10]. Ranljivosti različnega tipa in ravni tveganja se med seboj premalo razlikujejo tako z vidika ocen kot tudi z vidika njihovega opisa z atributi (vektorji CVSS). Slednje poizkuša izboljšati nova različica standarda CVSSv3 [65], a še ni na voljo dovolj podatkov, da bi lahko ocenili raven izboljšave. Obstajajo tudi dvomi o točnosti ocen CVSS [13, 68].

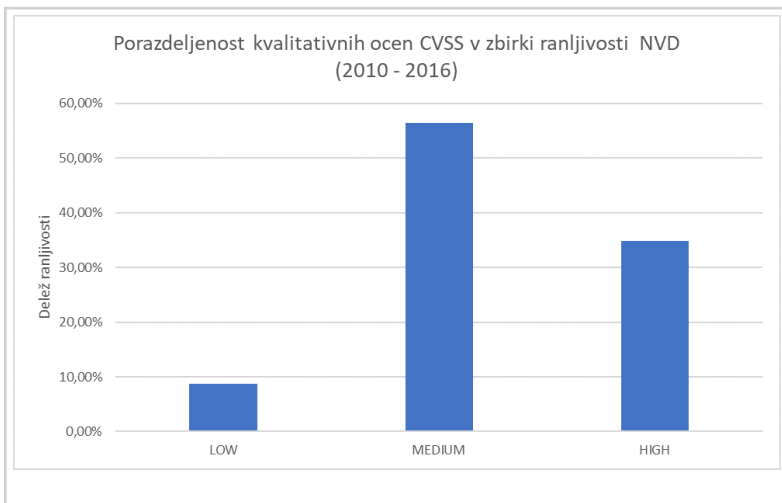
Sistem VRSS, ki predstavlja alternativo sistemu CVSS in prav tako temelji na atributih CVSS, doseže precej višjo raznolikost ocen [68, 69]. Kljub vsemu na podlagi tega še ne moremo sklepati, da je sistem VRSS boljši. Avtorji namreč ne navajajo dokazov, ki bi potrjevali, da so ocene VRSS bolj reprezentativne od ocen CVSS [13].

V literaturi se tako pogosto omenja, da osnovne ocene CVSS, kadar so uporabljene samostojno, niso najprimernejše za namene določanja prioritet pri odstranjevanju ranljivosti [8, 11, 14]. Nekateri učinkovitost sistema CVSS ocenjujejo tako slabo, da ga



Slika 3.5

Porazdelitev kvantitativnih ocen CVSS ranljivosti v podatkovni zbirki NVD v obdobju 2010 – 2016.



Slika 3.6

Porazdelitev kvalitativnih ocen CVSS v podatkovni zbirki NVD v obdobju 2010 – 2016.

enačijo s slučajnim pristopom [12]. Razlogi za takšne zaključke se skrivajo v dejstvu, da številne ranljivosti z visoko stopnjo resnosti nikoli niso izkoriščane v praksi [14].

Ranljivost lahko v različnih okoljih in časovnih obdobjih predstavlja različno raven grožnje. Z uporabo časovnih in okoljskih CVSS atributov lahko oceno ranljivosti prilagodimo trenutnemu stanju. Žal ne obstajajo raziskave, ki bi potrjevale večjo točnost takšnih ocen. Poleg tega tudi ni jasno, v kakšnem obsegu se časovni in okoljski atributi v praksi uporabljajo. Eno redkih priporočil načina uporabe okoljskih atributov je opisano v [70].

3.5 Uporaba predvidevanja pri določanju prioritet

Samo 15 % razkritih in javno objavljenih ranljivosti je v praksi kadar koli izkoriščenih. Poleg tega je raziskava na več zelo razširjenih programskih proizvodih (Microsoft Windows, Microsoft Office, Adobe Reader idr.) pokazala, da je bilo manj kot 35 % njihovih javno objavljenih ranljivosti kadar koli izkoriščenih. Pomemben je tudi podatek, da je med izkoriščenimi ranljivostmi do izkoriščanja v večini primerov prišlo v 58 dneh od njihovega razkritja [14]. Vse to so pomembna dejstva, ki odpirajo vprašanje, kako predvidevati ranljivosti, ki bodo izkoriščene z večjo verjetnostjo.

V ta namen potrebujemo podatke oz. primerno osnovo, na kateri lahko temeljijo predvidevanja. Žal je več študij potrdilo, da zbirka NVD predstavlja dokaj slabo osnovo [9, 71]. Pri evidentiranju podatkov o ranljivostih namreč prihaja do številnih netočnosti. Pripišemo jih lahko predvsem dejstvu, da v procesu odkrivanja in evidentiranja ranljivosti sodeluje zelo veliko posameznikov z zelo različnimi interesi. Osnovne vrste pristranskosti v zbirkah ranljivosti lahko pripišemo [72]:

Pristranskemu izbiranju Raziskovalec v vlogi "belega hekerja" odkriva ranljivosti le na produktu, ki ga dobro pozna. Prav tako se omejuje le na orodja in metode odkrivanja, ki jih dobro pozna. V isto kategorijo sodijo tudi neenotna merila pri izbiranju ranljivosti za javno objavo ter odločitve o tem, kaj je prava ranljivost in kaj ne.

Pristranskemu objavljanju Raziskovalec ali ponudnik programske opreme zaradi lastnih interesov objavi le del odkritih ranljivosti. Prav tako lahko razkrije le del informacij o ranljivosti.

Pristranski abstrakciji Pri večjem številu podobnih ranljivosti istega produkta se v zbir-

ki včasih pojavi en sam vnos, v drugem primeru je za vsako posamezno ranljivost dodan svoj vnos.

Pristranskemu merjenju Kakšno oceno dobi ranljivost, je odvisno od tega, kako je bila analizirana, preverjena in na koncu evidentirana. Na merjenje prav tako vpliva ocena ponudnika, ki jo lahko zaradi lastnih interesov preceni ali podceni.

Vsa omenjena odstopanja preprečujejo izvedbo določenih statističnih analiz, zato moramo biti pri uporabi podatkov previdni. Povsem zgrešeno je na primer med seboj primerjati produkte po številu odkritih ranljivosti in na podlagi tega sklepati o njihovi varnosti. Ko želimo med seboj primerjati učinkovitost metod za določanje prioritet, imajo pristranskosti lahko določen vpliv na merjenje. Omejimo jih lahko tako, da se pri analizah ne spuščamo na raven posameznega programskega produkta ali določenega tipa ranljivosti [14, 72].

Učinkovitost odstranjevanja ranljivosti je v prvi vrsti odvisna od tega, kako hitro metoda za določanje prioritet prepozna najnevarnejše ranljivosti. V primeru IS so najnevarnejše tiste, za katere obstaja visoka verjetnost, da jih bodo napadalci izkoriščali. Izboljšave obstoječih sistemov za ocenjevanje resnosti ranljivosti moramo torej iskati v smeri prepoznavanja indikatorjev izkoriščanja.

Dober indikator, ki bi ga lahko uporabili pri izgradnji sistema za predvidevanje bodočih izkoriščanj, je informacija o obstoju postopka izkoriščanja ranljivosti na črnem trgu [12]. Sistem CVSS v osnovi predvideva opisne atribute, s katerimi opišemo trenutno stanje ranljivosti, kar zadeva obstoj postopkov izkoriščanja. Žal časovni CVSS atributi, ki so namenjeni shranjevanju teh informacij in bi omogočali dobre napovedi, niso na voljo v zbirki NVD [9].

Interesi za zbiranje podatkov o postopkih izkoriščanja očitno niso tako enotni, kot to velja za evidentiranje osnovnih podatkov o ranljivostih. Atributi CVSS Temporal sicer obstajajo, a so proti plačilu na voljo le v omejenem obsegu in v različnih oblikah pri več ponudnikih. Takšni nepopolni podatki niso najprimernejši za končne uporabnike, saj je postopek uporabe zahteven in drag. Potrebo po nujnosti zagotavljanja tovrstnih podatkov vse bolj izražajo številni strokovnjaki za varnost [13].

V tej situaciji je za zagotavljanje proaktivne varnosti potrebno razviti alternativne metode. Najnaravnejši pristop je preprečevati nastajanje ranljivosti v fazi razvoja. To lahko dosežemo s predvidevanjem ranljivih programskih komponent, predvidevanjem skupnega števila ranljivosti v produktu ter drugimi kazalniki kakovosti programske

opreme [62, 73, 74]. Vse te metode nam prav nič ne koristijo na strani programske opreme, ki je že nameščena v IS in o kateri že obstajajo informacije o obstoju ranljivosti.

Avtorji CVSS standarda na potrebe uporabnikov odgovarjajo z novimi različicami standarda. Izboljšati ga poizkušajo predvsem z vedno boljšimi opisi ranljivosti. Vsaka nova izdaja standarda je uvedla nekaj sprememb in dopolnitev znotraj nabora atributov [64, 65]. Žal je uvajanje novih različic standarda dolgotrajen proces. Zadnja različica standarda CVSSv3 je bila predstavljena že v letu 2015, a je zaživela šele konec leta 2017. Poleg tega dodatni atributi v ničemer ne rešujejo ključnega problema, saj na njihovi osnovi ne moremo predvidevati bodočih izkoriščanj. Kako realna je ocena tveganja na osnovi CVSS, tako še naprej ostaja neodgovorjeno vprašanje.

Kljub vsemu obstajajo nekateri dejavniki, ki so povezani z izkoriščanjem ranljivosti in v dosedanjih raziskavah še niso bili preverjeni. Eden izmed možnih pristopov pri oceni tveganja, ki ga predstavlja ranljivost za izbrani IS, je upoštevanje lastnosti potencialnih napadalcev. Pri izkoriščanju vsake ranljivosti IS je v ozadju človek s povsem določenimi nameni in sposobnostmi [75]. Zato domnevamo, da varnost sistema lahko izboljšamo z upoštevanjem lastnosti napadalcev. Končni cilj raziskave je opredeliti metodo za določanje prioritet odstranjevanja ranljivosti, ki bo učinkovitejša od CVSS metode in drugih metod določanja prioritet.

4

Upoštevanje lastnosti napadalcev

4.1 Uvod

V tem poglavju predstavimo model ocenjevanja resnosti ranljivosti glede na lastnosti tipičnih napadalcev, ki se pojavljajo v okoljih sodobnih informacijskih sistemov. Tveganje opredelimo kot funkcijo sposobnosti in motivacije napadalca za izkoriščanje ranljivosti ter priložnosti, ki mu jih ranljivost nudi. V modelu so uporabljeni napadalci knjižnice TAL (angl. Threat Agent Library) [76], ki predstavljajo najtipičnejše skupine napadalcev v sodobnih IS. Opredeljeni so nad osmimi skupinami opisnih atributov. Pri opisih ranljivosti so uporabljeni osnovni atributi sistema CVSS. Osnovni namen modela je ugotoviti, koliko napadalcev knjižice TAL je zmožnih izkoristiti ranljivost. Predvidevamo, da večje število zmožnih napadalcev za izkoriščanje ranljivosti predstavlja večjo grožnjo za IS.

4.2 Vpliv napadalčevih lastnosti na velikost grožnje

Razvrščanje ranljivosti po prioriteti za odstranjevanje mora sovpadati z ravnijo tveganja, ki ga ranljivosti predstavljajo za IS. Zato je na tem mestu ključno vprašanje, ali višja ocena CVSS ranljivosti vedno pomeni tudi višje tveganje. Zanima nas torej, ali obstajajo boljše metode razvrščanja, kot je razvrščanje po oceni CVSS.

Varnostna tveganja so odvisna od več dejavnikov. Opišemo jih lahko z naslednjo funkcijo [7, 33]:

Definicija 4.1 (Tveganje):

$$Tveganje = f(grožnja, ranljivost, učinek)$$

Tveganje obstaja le, če sta prisotna tako ranljivost kot tudi grožnja. Pri obravnavi tveganj v zvezi s programsko opremo si posebno pozornost zaslužijo človeške grožnje, saj naravnih nesreč ne povezujemo z izkoriščanjem ranljivosti. Glavni uporabniki ranljivosti so namreč ljudje, ki si iz različnih razlogov prizadevajo izkoristiti IS ter ga s tem ogrožajo. Čeprav je možno, da bi posameznik nenamerno izkoristil ranljivost v programski opremi in pri tem povzročil škodo v IS, je verjetnost takšnega dogodka zelo majhna. Tudi v literaturi takšnih primerov nismo zaznali, zato jih v nadaljevanju ne obravnavamo.

Odkrivanje ranljivosti in izkoriščanje ranljivosti sta različna procesa, ki nista nujno

povezana. Ranljivost odkrijemo enkrat samkrat, postopkov izkoriščanja pa je lahko več ali pa tudi ni nobenega. Vse je odvisno od interesa napadalcev. V nadaljnji obravnavi se zato osredotočamo le na proces izkoriščanja že odkritih ranljivosti.

Ranljivost je pomanjkljivost sistema, ki jo napadalec lahko izkoristi, da ogrozi IS. Za izvedbo uspešnega napada morajo vendarle biti izpolnjeni nekateri osnovni pogoji. Napadalec mora biti motiviran, obvladati mora metode izkoriščanja (znanje, spretnosti, orodja ipd.) in najti mora primerno priložnost, da ranljivost lahko izkoristi [77]. Grožnjo, ki jo za IS predstavlja napadalec, lahko opišemo z naslednjo funkcijo:

Definicija 4.2 (Grožnja):

$$Grožnja = f(sposobnost, priložnost, motivacija)$$

Zaključimo lahko, da je izkoriščanje ranljivosti odvisno od lastnosti napadalcev, to pa se posledično odraža na velikosti tveganja. Preprosto povedano, pri dveh ranljivostih z enako oceno CVSS predstavlja večje tveganje tista, ki jo je zmožnih uspešno izkoristiti več napadalcev. Dokaz, da ocena CVSS ni najprimernejša za določanje prioritet, je tudi dejstvo, da mnoge ranljivosti z visoko oceno CVSS v praksi nikoli niso izkoriščane. Namesto upoštevanja zgolj ocene CVSS je zato bolje iskati metode za predvidevanje ranljivosti z visoko verjetnostjo izkoriščanja v praksi.

Na osnovi teh dejstev domnevamo, da je z upoštevanjem lastnosti potencialnih napadalcev možno izboljšati učinkovitost metod določanja prioritet za odstranjevanje ranljivosti. Poleg tega domnevamo, da je večja verjetnost, da bo izkoriščana tista ranljivost, ki jo je zmožnih izkoristiti več napadalcev. V zvezi s tem moramo poiskati odgovore na naslednja ključna vprašanja:

1. Kako oceniti vpliv napadalčevih lastnosti na zmožnost izkoriščanja ranljivosti?
2. Kako opredeliti razmerje med lastnostmi napadalca in lastnostmi ranljivosti v zvezi z izkoriščanjem?
3. Kako meriti učinkovitost metod za določanje prioritet odstranjevanja ranljivosti?

Glavni izziv pri prvem vprašanju je opredeliti funkcijo izkoriščanja ranljivosti. Pri drugem vprašanju moramo opredeliti mejne lastnosti, ki jih mora imeti potencialni

napadalec, da je zmožen izkoristiti izbrano ranljivost. Za rešitev takšne naloge potrebujemo primeren opis napadalca. Njegove lastnosti moramo nato primerjati z opisom izbrane ranljivosti in na tej osnovi podati odgovor o njegovi zmožnosti uspešnega izkoriščanja.

Pri vprašanju merjenja učinkovitosti metod je naša ključna naloga opredeliti kazalnike učinkovitosti. Osnovni cilj odstranjevanja ranljivosti je preprečevati napade in posledično omejevati škodo na IS. Da bi bilo tega čim manj, mora učinkovita metoda dajati prednost tistim ranljivostim, za katere obstaja večja verjetnost, da bodo izkoriščane v praksi. Osnova kazalnika za merjenje učinkovitosti in izvajanje primerjav posameznih metod bodo torej izkoristljive ranljivosti oz. morebitne posledice njihovega izkoriščanja.

Predpogoj za nepristransko merjenje je zagotovitev enakovrednih razmer pri vseh meritvah. Ker v realnem okolju to ni izvedljivo, si lahko pomagamo s simulacijo na osnovi agentov. Glavni elementi takšnega okolja so podatki o dejanskih ranljivostih in tipični napadalci ter njihove interakcije. V nadaljevanju moramo zato najprej proučiti lastnosti ranljivosti in napadalcev ter opredeliti način implementacije merilnega modela.

4.3 Opis ranljivosti

Po standardu CVSS je vsaka ranljivost opisana z vektorjem opisnih atributov. Ti podajajo osnovno informacijo o možnostih izkoriščanja, zahtevanih sposobnostih napadalca in o možnih učinkih napada. Podatki o vseh odkritih ranljivostih v programski opremi, ki je prisotna na trgu, se beležijo v podatkovno zbirko NVD (angl. *National Vulnerability Database*). Podrobneje smo jo predstavili že v poglavju 2.2.5.

Kljub temu da so v letu 2017 v zbirki NVD pričeli beležiti podatke o ranljivostih tudi po novi različici standarda CVSSv3, danes velika večina zbranih podatkov, ki jih lahko uporabimo v raziskovalne namene, še vedno pripada različici standarda CVSSv2. Raziskavo bomo zato izvedli nad podatki različice CVSSv2. Opredelitev atributov po tej različici standarda je podana v preglednici 4.1.

4.4 Opis napadalcev

Podobno kot ranljivosti opišemo z atributi, opredeljenimi po standardu CVSS, lahko z opisnimi atributi predstavimo tudi napadalce in njihove lastnosti. V obstoječi literaturi

Tabela 4.1

Opis CVSS osnovnih atributov ranljivosti po CVSSv2 različici standarda [64].

Atribut	Opis in zaloga vrednosti atributa
Access Vector	Način, kako napadalec lahko izkoristi ranljivost. `NETWORK, ADJACENT NETWORK, LOCAL`
Access Complexity	Zahtevnost izvedbe napada po pridobitvi dostopa. `LOW, MEDIUM, HIGH`
Authentication	Število preverjanj pristnosti med izkoriščanjem. `NONE, SINGLE INSTANCE, MULTIPLE INSTANCES`
Confidentiality Impact	Opredeljuje vpliv na zaupnost (razkritje, omejevanje). `NONE, PARTIAL, COMPLETE`
Integrity Impact	Opredeljuje vpliv na celovitost informacij. `NONE, PARTIAL, COMPLETE`
Availability Impact	Opredeljuje vpliv na razpoložljivost virov IS. `NONE, PARTIAL, COMPLETE`

najdemo več modelov, ki grožnje opisujejo na takšen način.

Ruf s sodelavci [78] predlaga tridimenzionalni model, kjer opisni vektor sestavljajo podatki o motivaciji (namerno, nenamerno), lokaciji grožnje (notranja, zunanja) in tipu grožnje (človek, tehnika, višja sila). Metodologija OCTAVE [79], ki se uporablja za obvladovanje varnostnih tveganj, loči štiri standardne kategorije groženj in jih opisuje z naslednjimi atributi: akter, informacijsko premoženje, motivacija (namerno, nenamerno), dostop do sistema (notranji, zunanji) in pričakovani izid (razkritje, uničenje, izguba, prekinitev delovanja, spreminjanje). Nekoliko drugačen nabor atributov predlaga Buckshaw [80]. Napadalce razporeja v devet tipičnih razredov in jih opisuje z naslednjimi lastnostmi: cilj napada, razpoložljivi viri, usposobljenost in tveganje. Po drugi strani Jones s sodelavci [33] v svojem modelu, kjer loči sedem tipičnih skupin napadalcev, kombinira tako kvalitativne kot tudi kvantitativne atribute. Med najbolj zanimivimi izpostavljamo naslednje: velikost skupine napadalcev, obseg financiranja, tehnična usposobljenost, razlog za izbor cilja, raven morale, znanje in dostop do sistema.

Dober kompromis med številom uporabljenih atributov in celovitim opisom ključ-

nih lastnosti napadalcev po naši oceni predstavlja knjižnica TAL (angl. Threat Agent Library) [76]. Nabor osmih atributov zajema tiste lastnosti, ki se najpogosteje pojavljajo v drugih modelih opisovanja groženj in so hkrati ključne za razumevanje lastnosti sodobnih napadalcev. Knjižnica se osredotoča le na človeške grožnje in ravno te nas zanimajo pri proučevanju izkoriščanja ranljivosti v programski opremi.

Opisi atributov knjižnice TAL, njihova zaloga vrednosti in urejenost teh množic so podani v preglednici 4.2. Vrednosti atributov posameznih lastnosti so urejene od manj do bolj nevarnih. Vseh osem atributov je opisnih, od tega jih je šest opredeljenih nad urejeno množico vrednosti, kar omogoča hierarhične primerjave. Vrednostno urejeni atributi so naslednji: *Intent* (namera), *Skills* (usposobljenost), *Resources* (viri), *Access* (dostop), *Limits* (omejitve) in *Visibility* (vidnost). Preostala atributa, *Objectives* (cilji) in *Outcome* (pričakovani izid), sta večvrednostna. Na osnovi osmih atributov in njihovih vrednosti je teoretično možno opisati več tisoč različnih napadalcev.

Naša osnovna naloga je, da znamo za poljubnega napadalca v vsakem trenutku odgovoriti na vprašanje *"Ali je opazovani napadalec zmožen izkoristiti ranljivost?"*. Večina atributov, ki jih opredeljuje knjižnica TAL, omogoča hierarhične primerjave, zato so primerni za uporabo v naši raziskavi. V nadaljevanju je potrebno določiti kriterije, ki jih mora izpolniti napadalec, da ga prepoznamo kot zmožnega izkoristiti ranljivosti.

4.5 Pogoji izkoriščanja ranljivosti

Knjižnica TAL opredeljuje 21 arhetipov napadalcev (angl. Threat Agent Archetypes), ki po prepričanju avtorjev predstavljajo najtipičnejše skupine napadalcev običajnih IS. Predstavljeni so v tabeli 4.3. Poleg atributov, ki opredeljujejo njihove lastnosti, so v tabeli navedene vrednosti, ki jih v nadaljevanju uporabimo v modelu za izračun posameznih spremenljivk.

Za vsakega napadalca iz knjižnice TAL želimo ugotoviti, ali je zmožen izkoristiti ranljivost, ki je opisana z izbranim vektorjem CVSS. Zato moramo najprej opredeliti pogoje izkoriščanja. V ta namen definiramo Boolovo funkcijo 4.3.

Tabela 4.2

Atributi za opis napadalcev, ki jih opredeljuje knjižnica TAL. Poleg vsakega atributa je naveden njegov opis, zaloga vrednosti ter urejenost vrednostne množice. Vrednosti so urejene od najmanj do najbolj nevarne. [76].

Atribut	Opis in zaloga vrednosti atributa
Intent	Opredeljuje, ali je namera napadalca sovražna. `NON HOSTILE < HOSTILE`
Skills	Opredeljuje raven usposobljenosti napadalca. `NONE < MINIMAL < OPERATIONAL < ADEPT`
Resources	Opredeljuje organizacijsko raven, na kateri napadalec običajno deluje. `INDIVIDUAL < CLUB < CONTEST < TEAM < ORGANIZATION < GOVERNMENT`
Access	Opredeljuje napadalčev način dostopa do informacijskega premoženja. `EXTERNAL < INTERNAL`
Limits	Opredeljuje, do kakšne mere je napadalec pripravljen kršiti zakonodajo. `CODE OF CONDUCT < LEGAL < EXTRA LEGAL MINOR < EXTRA LEGAL MAJOR`
Visibility	Opredeljuje, v kolikšni meri je napadalec pripravljen razkriti identiteto. `CLANDESTINE < COVERT < OVERT < MULTIPLE`
Objectives	Opredeljuje aktivnosti, ki jih napadalec namerava izvesti. `COPY, DENY, DESTROY, DAMAGE, TAKE, ALL`
Outcome	Opredeljuje cilje napadalca, ki jih z napadom želi doseči. `ACQUISITION/THEFT, BUSINESS ADVANTAGE, DAMAGE, EMBARRASSEMENT, ENTERTAINMENT, TECH ADVANTAGE`

Definicija 4.3 (Funkcija izkoriščanja):

$$isExploited\ (ThreatAgent,\ CVSS\ Vector) =$$

$$capability\ \wedge\ opportunity\ \wedge\ motivation$$

Funkcija *isExploited* sledi opredelitvi človeške grožnje, ki smo jo podali z izrazom 4.2. Njena vrednost je odvisna tako od lastnosti napadalca kot tudi od lastnosti ran-

Tabela 4.3

Knjižnica arhetipov napadalcev TAL (angl. Threat Agent Library) s pripadajočimi atributi. Poleg opisnih atributov so navedene pripadajoče numerične vrednosti, ki se uporabijo pri vrednotenju funkcije izkoriščanja (povzeto po [76]).

		Employee Recless	Employee Untrained	Info Partner	Anarchist	Civil Activist	Competitor	Corrupt Government Official	Data Miner	Employee Disgruntled	Government Cyberwarrior	Government Spy	Internal Spy	Irrational Individual	Legal Adversary	Mobster	Radical Activist	Sensationalist	Terrorist	Thief	Cyber Vandal	Vendor
Intent	Non hostile (1)	*	*	*																		
	Hostile (2)				*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Access	Internal (1)	*	*	*						*		*	*							*		*
	External (2)				*	*	*	*	*		*			*	*	*	*	*	*		*	
Visibility	Clandestine (1)			*			*		*			*	*							*		*
	Covert (2)	*				*										*			*		*	
	Overt (3)		*		*			*							*		*	*				
	Multiple (4)									*	*			*								
Resources	Individual (1)	*	*	*						*				*						*		
	Club (2)				*													*				
	Contest (3)																				*	
	Team (4)								*													*
	Organization (5)					*	*						*		*	*	*		*			
	Government (6)							*			*	*										
Skills	None (1)				*									*						*		
	Minimal (2)		*															*				
	Operational (3)			*						*											*	*
	Adept (4)	*				*	*	*	*		*	*	*		*	*	*		*			
Limits	Code of Conduct (1)		*	*																		
	Legal (2)	*													*							*
	Extra-legal, minor (3)					*	*	*	*				*				*	*		*	*	
	Extra-legal, major (4)				*					*	*	*		*		*			*			
Objectives	Copy					*	*		*			*	*		*							*
	Deny							*			*											
	Destroy				*					*	*								*			
	Damage									*	*								*			
	Take														*	*			*	*		
	All of the Above	*	*	*										*			*	*			*	
Outcome	Acquisition/Theft												*			*				*		
	Business Advantage						*	*	*			*			*							*
	Damage	*	*	*	*					*	*			*			*	*	*		*	
	Embarrassment	*	*	*		*				*	*			*	*		*	*				
	Tech Advantage						*	*	*			*	*									*

ljivosti. Ideja vrednotenja funkcije *isExploited* je prikazana na sliki 4.1.

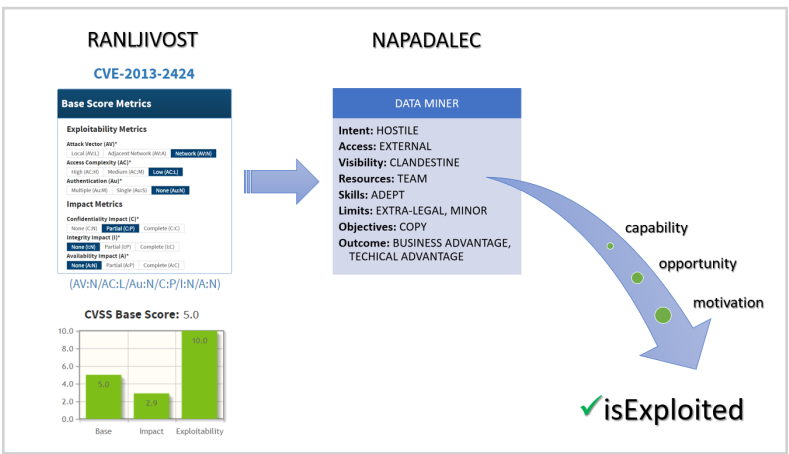
Za potrebe določanja prioritet je pomembno skupno število napadalcev iz knjižnice TAL, ki so zmožni izkoristiti dani vektor CVSS. Zato uvedemo spremenljivko TAC (angl. Threat Agent Count), ki jo uporabimo za štetje vseh zmožnih napadalcev. Opredelimo jo z definicijo 4.4, kjer vse vrednosti `TRUE` ovrednotimo z 1 in vse vrednosti `FALSE` z 0. Na sliki 4.2 je prikazan način določanja vrednosti TAC.

Definicija 4.4 (Threat Agent Count):

$$TAC = \sum_{agent \in TAL} isExploited\,(agent, vect)$$

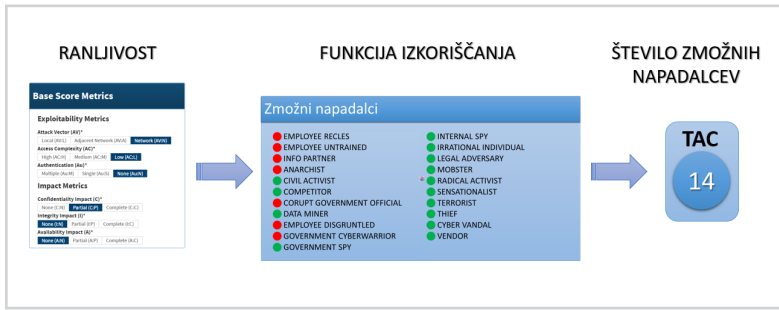
Kjer je *agent* napadalec knjižnice TAL in *vect* ranljivost, opredeljena z vektorjem CVSS.

Naš osnovni cilj je čim učinkoviteje omejevati tveganja v IS. Po naših pričakovanjih to lahko dosežemo tako, da pri odstranjevanju dajemo prednost tistim ranljivostim, za katere obstaja več zmožnih napadalcev. To pomeni, da moramo izmed vseh ranljivosti v IS kot prvo odstraniti tisto z najvišjo vrednostjo TAC. Metodo določanja prioritet, ki sledi temu opisu, bomo poimenovali HTAC (angl. Highest Threat Agent Count).



Slika 4.1

Funkcija izkoriščanja *isExploited* se ovrednoti tako, da napadalec primerja svoje lastnosti z lastnostmi ranljivosti. Če je dovolj sposoben (*capability*) in motiviran (*motivation*) in če opisana ranljivost zanj predstavlja priložnost (*opportunity*), je zmožen izkoriščanja. Ranljivost je predstavljena s CVSS vektorjem, napadalec pa z atributi knjižnice TAL.



Slika 4.2

Število TAC predstavlja število vseh napadalcev iz knjižnice TAL, ki so zmožni izkoristiti opazovano ranljivost. Za določitev števila TAC mora vsak napadalec ovrednotiti funkcijo izkoriščanja *isExploited*.

Ideja je predstavljena na sliki 4.3.

V naslednjih podpoglavjih opredeljujemo spremenljivke *capability*, *opportunity* in *motivation*, ki smo jih uporabili pri opredelitvi funkcije *isExploited* (4.3).

4.5.1 *Sposobnost izkoriščanja ranljivosti*

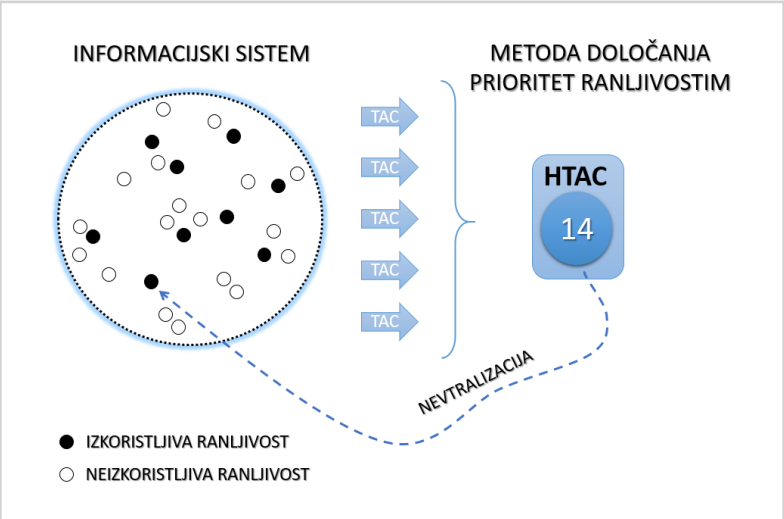
Pri vsakem napadalcu je razpolaganje s primernimi sposobnostmi osnovni pogoj za uspešno izkoriščanje izbrane ranljivosti. Sposobnost je odvisna od znanja, spretnosti in razpoložljivih virov. S spremenljivko *agentCapability* opredeljujemo sposobnost napadalca kot funkcijo atributov knjižnice TAL. Njen izračun je opredeljen v definiciji 4.5. Sposobnost napadalca je opredeljena kot produkt vrednosti urejenostnih atributov *Skills* in *Resources*. Vrednosti atributov so podane v tabeli 4.3 poleg njihovih opisov. Atribut *Skills* ima 4 in *Resources* 6 različnih vrednosti, kar skupaj predstavlja 24 različnih možnih vrednosti spremenljivke *agentCapability*. Funkcija *Ord* v definiciji 4.5 vrne pripadajočo vrednost urejenostnega atributa.

Definicija 4.5 (Sposobnost izkoriščanja):

$$agentCapability = Ord\,(Agent.Skills) * Ord\,(Agent.Resources)$$

Agent.Skills in *Agent.Resources* sta atributa napadalca po knjižnici TAL.

Ranljivosti se med seboj razlikujejo po zahtevnosti izkoriščanja. Kako zahtevna je posamezna ranljivost, razberemo iz vrednosti njenega CVSS atributa *Access Complexity* (tabela 4.1). Standard CVSS opredeljuje tri stopnje zahtevnosti: `LOW`, `MEDIUM` in `HIGH`.



Slika 4.3

Metoda HTAC deluje tako, da v prvem koraku za vsako ranljivost v IS izračuna vrednost TAC. Nato izbere ranljivost z največjo vrednostjo TAC in jo odstrani oz. nevtralizira. Če obstaja več ranljivosti z najvišjo vrednostjo, odstrani tisto, ki jo prej odkrije.

Tabela 4.4

Izračun vrednosti Boolove spremenljivke *capability* glede na zahtevnost ranljivosti, ki je podana z vrednostjo atributa *Access Complexity* in sposobnostjo napadalca. Ker ima spremenljivka *agentCapability* vrednosti na intervalu od 1 do 24, razdelimo interval na tri enake dele. Za najzahtevnejše ranljivosti mora imeti napadalec sposobnosti v zgornji tretjini intervala, za srednje zahtevne v zgornjih dveh tretjinah, za najpreprostejše ranljivosti pa zadostuje poljubna sposobnost.

CVSS Access Complexity	*capability*
`LOW`	`True`
`MEDIUM`	$agentCapability > 8$
`HIGH`	$agentCapability > 16$

Napadalci z večjo sposobnostjo so sposobni izkoriščati zahtevnejše ranljivosti. Zato zalogo vrednosti spremenljivke *agentCapability* razdelimo na tri enako velike intervale s po 8 vrednostmi v vsakem. Višja vrednost spremenljivke *agentCapability* pomeni večjo sposobnost. Skladno s tem Boolovo spremenljivko *capability* iz enačbe (4.3) izračunamo s pogoji, ki so podani v tabeli 4.4.

Za nazoren prikaz ugotavljanja sposobnosti napadalca za izkoriščanje izbrane ranljivosti si poglejmo konkreten primer. Vzemimo poljubno ranljivost, ki je opisana z

naslednjim vektorjem CVSS `[AV:N/AC:M/Au:N/C:P/I:P/A:P]`. Gre za enega tistih vektorjev CVSS, ki se najpogosteje pojavljajo v podatkovni zbirki NVD.

Iz dela vektorja, ki se nanaša na zahtevnost ranljivosti (`AC:M`), ugotovimo, da je vrednost atributa *Access Complexity* enaka `MEDIUM`. Skladno s tabelo 4.4 mora sposobnost napadalca *agentCapability* dosegati vsaj vrednost 8, da bi bila dosežena zahtevana sposobnost izkoriščanja podane ranljivosti. Preverimo zdaj, ali podano ranljivost lahko izkoristi "podatkovni rudar" (angl. Data miner) (tabela 4.3). Na osnovi vrednosti njegovih atributov *Agent.Skills* = `ADEPT` in *Agent.Resources* = `TEAM` sledi naslednji izračun:

$$agentCapability = Ord(\texttt{ADEPT}) * Ord(\texttt{TEAM}) > 8 = 4 * 4 > 8 = True \quad (4.1)$$

Ugotovili smo, da je podatkovni rudar glede na svoje lastnosti sposoben izkoristiti ranljivost z vektorjem CVSS `[AV:N/AC:M/Au:N/C:P/I:P/A:P]`.

4.5.2 *Priložnost, ki jo nudi ranljivost*

V splošnem so ranljivosti priložnosti za napadalce, da izkoristijo IS. Kljub vsemu obstajajo različne omejitve tako na strani ranljivosti kot tudi na strani napadalcev, ki odločajo o tem, ali bo do izkoriščanja v resnici prišlo. Ali določena ranljivost res predstavlja priložnost za izbranega napadalca, ugotovimo na osnovi lastnosti obeh. V ta namen opredelimo Boolovo spremenljivko *opportunity*, ki se ovrednoti skladno z definicijo 4.6. V njej nastopajo tri komponente priložnosti, in sicer priložnost dostopa do sistema (*accessOpportunity*), odsotnost preverjanja pristnosti ali dopustno preverjanje (*authOpportunity*) ter priložnost izpolnitve napadalčevega cilja (*objectiveOpportunity*). Vsi trije pogoji morajo biti izpolnjeni, da napadalec izbrano ranljivost prepozna kot dovolj veliko priložnost za izkoriščanje.

Definicija 4.6 (Priložnost za izkoriščanje):

$$opportunity = accessOpportunity \wedge authOpportunity \wedge objectiveOpportunity$$

Najočitnejša omejitev, ki napadalcem preprečuje izkoriščanje, je dostop do ranljivosti. Standard CVSS razlikuje tri stopnje zahtevnosti dostopa (tabela 4.1). Ranljivosti, ki jih lahko izkoristimo preko interneta, predstavljajo najenostavnejši dostop (*Access Vector* = `NETWORK`). Takšno ranljivost lahko samostojno doseže vsak posameznik, ki

Tabela 4.5

Vrednotenje Boolove spremenljivke *accessOpportunity* glede na vrednost *Access Vector* atributa ranljivosti ter *Agent.Access* in *Agent.Resources* lastnosti napadalca. *Agent.Access* in *Agent.Resources* sta atributa napadalca, opredeljena s knjižnico TAL.

CVSS Access Vector	*accessOpportunity*
`LOCAL`	*Agent.Access* = `INTERNAL`
`ADJACENT NETWORK`	*Agent.Resources* > `INDIVIDUAL`
`NETWORK`	`True`

ima dostop do interneta. Ker ima danes to možnost praktično že vsakdo, je vrednost spremenljivke *accessOpportunity* pri takšnih ranljivostih za vse napadalce vedno `TRUE`.

Najtežje je doseči ranljivosti, kjer je za izkoriščanje potreben lokalni dostop do sistema (*Access Vector* = `LOCAL`). Pri takšnih ranljivostih je priložnost za izkoriščanje omejena zgolj na tiste napadalce, ki imajo lokalen dostop do ranljive komponente.

Vse preostale ranljivosti so med tema ekstremoma. Njihovo izkoriščanje je možno preko bližnjih omrežij (*Access Vector* = `ADJACENT NETWORK`), česar pa ne more narediti vsakdo. Zato v tem primeru opredelimo, da takšne ranljivosti predstavljajo dovolj veliko priložnost le za tiste napadalce, ki lahko računajo na ustrezno pomoč drugih oseb. Pogoji za ovrednotenje Boolove spremenljivke *accessOpportunity* so podani v tabeli 4.5.

V primeru vektorja `[AV:N/AC:M/Au:N/C:P/I:P/A:P]`, ki smo ga uporabili že pri preverjanju sposobnosti podatkovnega rudarja, je vrednost spremenljivke *accessOpportunity* `TRUE`. Vrednost atributa *Access Vector* te ranljivosti je namreč `NETWORK` (`AV:N`), kar pomeni, da je to priložnost za katerega koli napadalca. Posledično to pomeni, da podatkovni rudar prepozna to ranljivost kot svojo priložnost.

V številnih primerih je izkoriščanje ranljivosti pogojeno s predhodno prijavo v sistem. Običajno se ob vsaki prijavi v sistem zabeležijo nekateri podatki, ki zagotavljajo sled. Slednje lahko za napadalca predstavlja nepremostljivo oviro, saj želi ostati anonimen. Ugotovimo torej lahko, da takšne ranljivosti za nekatere napadalce ne predstavljajo priložnosti. V ta namen opredelimo Boolovo spremenljivko *authOpportunity*.

Napadalci se med seboj razlikujejo v lastnosti *Visibility* (vidnost, prepoznavnost). Ta določa, do kakšne mere je napadalec pripravljen razkriti svojo identiteto. Po drugi

Tabela 4.6

Vrednotenje Boolove spremenljivke *authOpportunity* glede na vrednost *Authentication* atributa ranljivosti in *Agent.Visibility* lastnosti napadalca. *Agent.Visibility* je atribut napadalca, opredeljen s knjižnico TAL.

CVSS Authentication	*authOpportunity*
`NONE`	`True`
`SINGLE INSTANCE`	$Agent.Visibility \geq$ `OVERT`
`MULTIPLE INSTANCES`	$Agent.Visibility \geq$ `OVERT`

strani standard CVSS z atributom *Authentication* loči tri ravni zahtevnosti izkoriščanja ranljivosti z ozirom na zahteve po predhodni prijavi v sistem. V modelu, ki ga predstavljamo, ločimo le dva možna scenarija odzivanja napadalcev. Ranljivosti, ki ne zahtevajo kakršne koli predhodne prijave v sistem (*Authentication* = `NONE`), predstavljajo priložnost za vsakogar. Vse ostale ranljivosti so priložnosti samo za tiste napadalce, ki niso zaskrbljeni glede razkritja identitete. Pogoji za vrednotenje spremenljivke *authOpportunity* so podani v tabeli 4.6.

CVSS vektor ranljivosti iz prejšnjih primerov `[AV:N/AC:M/Au:N/C:P/I:P/A:P]` kaže, da je vrednost atributa *Authentication* enaka `NONE` (`Au:N`). Takšna ranljivost je priložnost za vsakogar (glej tabelo 4.6). Posledično to pomeni, da spremenljivka *authOpportunity* dobi vrednost `TRUE`. Zaključimo lahko, da podatkovni rudar odsotnost zahteve po prijavi v sistem pri izkoriščanju ranljivosti dojema kot svojo priložnost.

Tretja komponenta priložnosti, ki jo napadalcu predstavlja ranljivost, je priložnost doseči zastavljeni cilj. V ta namen opredelimo Boolovo spremenljivko *objectiveOpportunity*. Možni cilji napadalcev so v knjižnici TAL opredeljeni z atributom *Objective* (glej tabelo 4.2).

Vsako prizadevanje napadalca za dosego cilja je povezano z ogrožanjem vsaj ene komponente varnosti IS: zaupnosti (confidentiality, C), celovitosti (integrity, I) ali razpoložljivosti (availability, A). Vsak cilj, ki je opredeljen v knjižnici TAL, zato lahko povežemo z eno ali več komponentami varnosti. Povezanost ciljev s komponentami varnosti je opredeljena z množicami v definiciji 4.7.

Definicija 4.7 (Cilji napadalca, razporejeni v množice po komponentah varnosti CIA):

$$C = \{COPY, TAKE, ALL\}$$

$$I = \{DENY, DAMAGE, ALL\}$$

$$A = \{DESTROY, TAKE, ALL\}$$

Elementi množic so vse možne vrednosti atributa *Objectives* knjižnice TAL.

Po drugi strani vektorji CVSS označujejo, katere komponente varnosti so lahko ogrožene z izkoriščanjem izbrane ranljivosti in v kakšnem obsegu. V ta namen CVSS opredeljuje atribute za opis učinka ranljivosti (*Impact* atributi). Predstavljeni so v tabeli 4.1. Kadar se potencialni učinki ranljivosti in cilji napadalca nanašajo na iste komponente varnosti, ranljivost predstavlja priložnost za napadalca. Slednje opredelimo z naslednjo enačbo:

Definicija 4.8 (Priložnost za napadalca):

$$objectiveOpportunity =$$

$$(Vuln.ConfImpact \neq \texttt{NONE} \wedge Agent.Objectives \in C) \vee$$

$$(Vuln.IntegImpact \neq \texttt{NONE} \wedge Agent.Objectives \in I) \vee$$

$$(Vuln.AvailImpact \neq \texttt{NONE} \wedge Agent.Objectives \in A)$$

Agent.Objectives je lastnost napadalca, opredeljena s knjižnico TAL. $Vuln.ConfImpact$, $Vuln.IntegImpact$ in $Vuln.AvailImpact$ so lastnosti ranljivosti, opredeljene z vektorjem CVSS.

V primeru CVSS vektorja `[AV:N/AC:M/Au:N/C:P/I:P/A:P]` ugotovimo, da ima ranljivost učinek na zaupnost, celovitost in razpoložljivost. Vsi atributi za opis učinka imajo vrednost `PARTIAL` (`C:P/I:P/A:P`). Cilji podatkovnega rudarja so opredeljeni z naslednjo množico ciljev: $\{COPY\}$. Cilj je en sam in ta se nanaša na ogrožanje zaupnosti oz. na množico C, ki je podana z definicijo 4.7. Glede na enačbo (4.8) ugotovimo, da spremenljivka *objectiveOpportunity* dobi vrednost `TRUE`, saj se cilji podatkovnega

rudarja in učinki ranljivosti nanašajo na iste varnostne komponente. V prikazanem primeru je to komponenta zaupnosti.

Na našem praktičnem primeru smo do te točke preverili vse pogoje, ki se nanašajo na vrednotenje Boolove spremenljivke *opportunity*, in vsi so bili izpolnjeni. Posledično ugotovimo, da podatkovni rudar prepozna ranljivost, ki je podana s CVSS vektorjem `[AV:N/AC:M/Au:N/C:P/I:P/A:P]`, kot priložnost za izkoriščanje.

4.5.3 Motivacija za izkoriščanje ranljivosti

Motivacija napadalca za izkoriščanje ranljivosti je odvisna od več njegovih lastnosti. V ta namen opredelimo Boolovo spremenljivko *motivation* kot kombinacijo treh dejavnikov, in sicer sovražnosti, želje po doseganju rezultatov v zvezi z IS ter odnosa do spoštovanja pravnih in etičnih omejitev (4.9).

Definicija 4.9 (Motivacija napadalca):

$$motivation = motiveHostile \wedge motiveOutcome \wedge motiveMeetLimits$$

Izkoriščanje ranljivosti je povezano izključno z ljudmi, ki to delajo načrtno. Prav zato jih imenujemo napadalci. Posamezniki, ki pri svojem delu naredijo napako in s tem nenamerno ogrozijo IS, ne sodijo mednje. Namen napadalcev je povzročiti škodo ali se na nepošten način okoristiti s tujim premoženjem. Oboje povezujemo s sovražnostjo. V knjižnici TAL jih prepoznamo preko atributa *Intent*. Spremenljivko *motiveHostile* zato izračunamo po naslednji enačbi:

Definicija 4.10 (Motivacija za sovražno delovanje):

$$motiveHostile = (Agent.Intent = \texttt{HOSTILE})$$

Agent.Intent je lastnost napadalca, opredeljena s knjižnico TAL.

Motiviran napadalec ima v mislih vsaj en želeni rezultat. V knjižnici TAL prepoznamo pričakovanja napadalcev glede želenih izidov preko atributa *Agent.Outcome*. Atribut je predstavljen z večvrednostno množico. Kadar je ta neprazna, posameznika lahko prepoznamo kot motiviranega. Zato v našem modelu izračunamo motivacijo posameznega napadalca za dosego rezultata na naslednji način:

Definicija 4.11 (Motivacija za dosego želenega izida oz. rezultata aktivnosti):

$$motiveOutcome = (Agent.Outcome \neq \{\})$$

Agent.Outcome je lastnost napadalca, opredeljena s knjižnico TAL.

Nekateri napadalci imajo različne pravne in etične zadržke pri izkoriščanju ranljivosti, kar jih vsaj deloma omejuje pri kršenju zakona in ignoriranju pravil. V ta namen knjižnica TAL opredeljuje urejenostni atribut *Limits* s štirimi vrednostmi. Predstavljene so v tabeli 4.2.

Napadalci so pripravljeni uporabiti samo tiste ranljivosti, ki sovpadajo z njihovimi potrebami glede doseganja ciljev in katerih učinek hkrati ne presega njihovih omejitev. Na primer osebe, ki strogo spoštujejo zakon, niso zainteresirane za kakršno koli izkoriščanje ranljivosti (*Agent.Limits* = `CODE OF CONDUCT`). V nasprotju z njimi obstajajo napadalci brez kakršnih koli omejitev, ki ne oklevajo pri izkoriščanju katere koli ranljivosti (*Agent.Limits* = `EXTRA LEGAL MAJOR`).

Med tema ekstremoma so napadalci, ki spoštujejo zakonske omejitve (*Agent.Limits* = `LEGAL`) in temu prilagodijo svoje aktivnosti. Zainteresirani so za izkoriščanje tistih ranljivosti, s katerimi ne prekoračijo zakona. Skladno z našo interpretacijo sodijo v to skupino tiste ranljivosti, pri katerih je vrednost CVSS atributa *Confidentiality impact* enaka `PARTIAL` (`C:P`). Takšni napadalci so zainteresirani za pridobitev specifičnih informacij iz ciljnega IS, a ne tako, da bi neposredno povzročili škodo.

Napadalci z `EXTRA LEGAL MINOR` omejitvami kršijo zakon v relativno omejenem obsegu ter na nenasilen način, kot sta na primer manjši vandalizem ali zloraba. Zato so zainteresirani le za izkoriščanje takšnih ranljivosti, ki omogočajo tovrstne aktivnosti. V skladu z našo interpretacijo jih zanimajo CVSS vektorji ranljivosti z delnim učinkom na zaupnost, celovitost in razpoložljivost (`PARTIAL`, `C:P` ali `I:P` ali `A:P`) kot tudi vektorji s polnim učinkom na zaupnost (`COMPLETE`, `C:C`). Po našem mnenju namreč javno razkritje zaupnih informacij ne pomeni tako velikega prekrška kot uničenje ali povzročanje škode na informacijskem premoženju.

Odsotnost zadržkov napadalca do izkoriščanja opazovane ranljivosti opišemo z Boolovo spremenljivko *motiveMeetLimits*. Izračunamo jo po enačbi 4.12. V njej $cvssVect$ predstavlja vektor ranljivosti, ki je predmet izkoriščanja, in V množico dovoljenih vektorjev napadalca, skladno z njegovim atributom *Agent.Limits*. Množice dovoljenih

Tabela 4.7

Dovoljeni vektorji CVSS ranljivosti (V) skladno z omejitvami napadalcev, ki jih opredeljuje atribut *Agent.Limits*.

Agent.Limits	V *(dovoljeni so vektorji z naslednjimi vrednostmi CVSS Impact atributov)*
`CODE OF CONDUCT`	noben vektor
`LEGAL`	C:P
`EXTRA LEGAL MINOR`	C:C ali C:P ali I:P ali A:P
`EXTRA LEGAL MAJOR`	vsi vektorji

vektorjev za vse vrednosti atributa *Agent.Limits* so podane v tabeli 4.7.

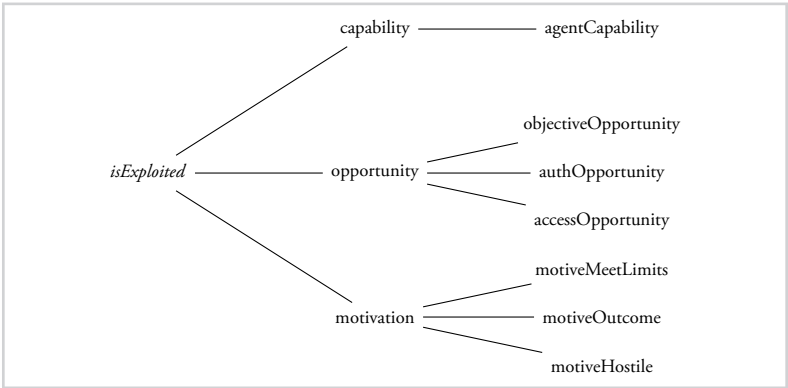
Definicija 4.12 (Motivacija):

$$motiveMeetLimits = (cvssVect \in V)$$

Pri tem je $cvssVect$ vektor CVSS, s katerim so opredeljene lastnosti ranljivosti.

Zdaj lahko ovrednotimo Boolovo spremenljivko *motivation* za podatkovnega rudarja pri ranljivosti z vektorjem CVSS `[AV:N/AC:M/Au:N/C:P/I:P/A:P]`. Spremenljivka *motiveHostile* ima vrednost `TRUE`, ker ima podatkovni rudar sovražen namen (*Agent.Intent* = `HOSTILE`). Spremenljivka *motiveOutcome* ima vrednost `TRUE`, ker ima podatkovni rudar v atributu *Outcome* opredeljena dva cilja (`BUSINESS ADVANTAGE` in `TECH ADVANTAGE`). Spremenljivka *motiveMeetLimits* ima prav tako vrednost `TRUE`, saj CVSS vektor ranljivosti sodi med dovoljene vektorje glede na omejitve podatkovnega rudarja (*Agent.Limits* = `EXTRA LEGAL MINOR`). Opazovani vektor ima namreč vse atribute učinka (angl. Impact Attributes) vrednosti `PARTIAL` (`C:P`, `I:P` in `A:P`), kar je dopustno za napadalca z omejitvami `EXTRA LEGAL MINOR` (glej tabelo 4.7). Na podlagi rezultatov posameznih komponent motivacije smo ugotovili, da je podatkovni rudar motiviran za izkoriščanje ranljivosti z vektorjem `[AV:N/AC:M/Au:N/C:P/I:P/A:P]`, saj so vsi pogoji motivacije izpolnjeni.

Opredelili smo vse spremenljivke, ki so potrebne za izračun funkcije *isExploited*



Slika 4.4

Diagram povezav vseh opredeljenih spremenljivk pri ovrednotenju funkcije izkoriščanja ranljivosti *isExploited*.

(4.3). Na sliki 4.4 je podan diagram povezav spremenljivk pri njenem ovrednotenju. Z ovrednotenjem te funkcije na našem primeru ugotovimo, da je podatkovni rudar zmožen izkoristiti ranljivost z vektorjem `[AV:N/AC:M/Au:N/C:P/I:P/A:P]`, saj je dovolj sposoben (*capability* = `TRUE`), ranljivost mu predstavlja priložnost (*opportunity* = `TRUE`) in je motiviran za njeno izkoriščanje (*motivation* = `TRUE`).

4.6 Zaključek

V tem poglavju smo predstavili idejo nove metode za določanje prioritet ranljivostim pri odstranjevanju. Inovativnost ideje se kaže v tem, da upošteva lastnosti potencialnih napadalcev. Za namen določanja prioritet smo opredelili število TAC (4.4), ki pove, koliko potencialnih napadalcev iz knjižnice TAL je zmožnih izkoristiti posamezno ranljivost. Ranljivost z najvišjo vrednostjo TAC dobi najvišjo prioriteto. Temu primerno smo predlagani metodi izbrali ime, in sicer *Highest Threat Agent Count* (HTAC). Najprej smo opredelili funkcijo izkoriščanja ranljivosti *isExploited* (4.3) in nato nad atributi knjižnice TAL še pogoje izkoriščanja. Ti se navezujejo na posamezne komponente grožnje, in sicer na sposobnost napadalca, priložnost, ki jo nudi ranljivost, ter motivacijo napadalca za izkoriščanje.

V nadaljevanju želimo preveriti učinkovitost predlagane metode in jo primerjati z drugimi obstoječimi metodami. Zato moramo najprej rešiti izziv, kako meriti učinkovitost, in nato primerjati različne metode med seboj.

5

Merjenje učinkovitosti metod za določanje prioritet ranljivostim

5.1 Uvod

V tem poglavju predstavimo model merjenja učinkovitosti metod za določanje prioritet ranljivostim pri odstranjevanju. V ta namen potrebujemo kvantitativno metriko, ki omogoča primerjave med njimi. Skupni cilj vseh metod na tem področju je v največji možni meri omejiti varnostna tveganja. Skladno z opredelitvijo varnostnih tveganj je eno izmed možnih meril izpostavljenost grožnjam. Pravo grožnjo za IS predstavljajo ranljivosti, ki jih je možno izkoristiti, zato predlagamo kazalnik, ki upošteva to dejstvo.

Sledi opis več znanih metod določanja prioritet pri odstranjevanju ranljivosti in opredelitev nekaj novih. Slednje upoštevajo lastnosti napadalcev. Osnovane so na ideji, ki smo jo predstavili v prejšnjem poglavju.

V drugem delu poglavja predstavimo eksperimentalno okolje ter uporabljene podatke. Sledi opis načrta izvedbe eksperimenta po korakih. Poglavje zaključimo s podrobno predstavitvijo rezultatov in razpravo.

5.2 Model merjenja učinkovitosti

Osnovni cilj pri odstranjevanju ranljivosti je zmanjšati izpostavljenost IS grožnjam. Zato iščemo metodo, ki bo pri tem opravilu čim učinkovitejša. Ob uvodu v raziskavo smo izpostavili domnevo, da je z upoštevanjem lastnosti napadalcev možno izboljšati obstoječe metode določanja prioritet ranljivostim. Da bi lahko preverili domnevo, potrebujemo primeren merilni instrument, ki bo omogočal kvantitativne primerjave med različnimi metodami. Po našem vedenju še ne obstaja metrika, ki bi omogočala tovrstne primerjave. Obstoječe raziskave na tem področju so namreč večinoma usmerjene v preverjanje točnosti ocen CVSS in njihove uporabnosti. V nadaljevanju se zato osredotočamo na razvoj kvantitativne metrike za merjenje izpostavljenosti IS grožnjam.

Učinkovita metoda mora upoštevati dejansko raven grožnje, ki jo posamezna ranljivost predstavlja za IS. Zavedati se moramo, da je v realnosti v vsakem trenutku le del ranljivosti takšnih, da so jih napadalci zmožni izkoristiti. Odstranjevanje ranljivosti zgolj zaradi visoke ocene CVSS zato ni preveč smiselno, ali z drugimi besedami, zagotovo ni najučinkovitejše. Pravo grožnjo za IS predstavljajo tiste ranljivosti, ki jih je že možno izkoriščati, in tiste, ki bodo predmet izkoriščanja v prihodnosti. Zaradi lažje in natančnejše razlage bomo v nadaljevanju zanje uporabljali pojem *izkoristljive* ranljivosti, saj ta opis v slovenskem jeziku najnatančneje opisuje njihovo ključno

lastnost.

Raziskave potrjujejo, da ocena CVSS ni najboljši pokazatelj realne grožnje, saj ni primerna za predvidevanje dejanskih izkoriščanj v praksi. Bistveno boljše rezultate pri predvidevanju izkoriščanja dosežemo, če se opremo na zbirko preverjenih postopkov izkoriščanja (angl. Proof-of-Concept Exploit). Najtočnejše napovedi pa so možne, ko imamo konkretne informacije o obstoju določenega postopka izkoriščanja na črnem trgu [12].

Žal informacije o obstoju postopkov izkoriščanja za posamezne ranljivosti niso vedno na voljo, poleg tega pa v tajnosti stalno nastajajo novi. Prav zato iščemo metodo, ki bo imela boljše napovedne sposobnosti glede izkoristljivih ranljivosti. Z uporabo takšne metode bo IS manj izpostavljen grožnjam, saj bo v njem manj izkoristljivih ranljivosti, kar je tudi naš osnovni cilj. Učinkovitost metod določanja prioritet bomo torej merili tako, da bomo ugotavljali izpostavljenost IS izkoristljivim ranljivostim.

Število izkoristljivih ranljivosti v IS se stalno spreminja. Na vsakem koraku uporabe izbrane metode namreč odstranimo ranljivost z najvišjo prioriteto. Denimo, da je na *i-tem* koraku uporabe metode množica preostalih izkoristljivih ranljivosti v IS predstavljena z E_i. Potem izpostavljenost IS izkoristljivim ranljivostim na intervalu $[0..k]$ opredelimo z definicijo (5.1), kjer k predstavlja število korakov uporabe metode oz. število odstranjenih ranljivosti iz IS. Skladno s to opredelitvijo nižja vrednost $ExposureEV_k$ predstavlja večjo učinkovitost metode.

Definicija 5.1 (Izpostavljenost izkoristljivih ranljivosti IS grožnjam):

$$ExposureEV_k = \sum_{i=0}^{k} |E_i|$$

$|E_i|$ je moč množice izkoristljivih ranljivosti v IS po i odstranjenih ranljivostih iz IS; k je število korakov izvajanja metode.

Predlagano metriko bomo s pomočjo eksperimenta ovrednotili na več metodah določanja prioritet. V ta namen potrebujemo verodostojne podatke. Podatke o ranljivostih lahko najdemo v različnih oblikah v več podatkovnih virih. Zanimajo nas podatki o vseh odkritih ranljivostih v določenem časovnem obdobju kot tudi podatki o njihovem izkoriščanju v praksi. Da bi dobili popolno sliko o vsaki posamezni ranljivosti,

moramo podatke obeh vrst povezati v enovito podatkovno zbirko.

Popoln seznam vseh odkritih ranljivosti skupaj z njihovimi ključnimi lastnostmi je na voljo v NVD podatkovni zbirki. Na žalost nimamo podobnega vira o vseh ranljivostih, ki so bile izkoriščane v praksi. Kljub vsemu obstajajo nekateri viri, ki omogočajo sklepanja o izkoriščanju posameznih ranljivosti. Eden takšnih, ki se pogosto uporablja v raziskovalne namene, je zbirka EDB (Exploit-db[1]). Dejansko je to arhiv vseh vrst postopkov izkoriščanja programske opreme. Poleg vsakega postopka so navedene izkoriščane ranljivosti z oznakami CVE, kar omogoča povezovanje z zbirko NVD. V osnovi se podatki iz zbirke EDB uporabljajo pri izvajanju penetracijskih testov.

Med ostalimi viri moramo izpostaviti seznam ranljivosti EKITS,[2] ki se nanaša na točno določeno skupino izkoriščanj. Gre za ranljivosti, ki se uporabljajo v celovitih programskih paketih za izkoriščanje (angl. exploit kits) in jih je možno kupiti na črnem trgu. Za potrebe našega eksperimenta bomo torej uporabili podatke iz podatkovnih zbirk EDB, EKITS in NVD.

5.3 Obstoječe metode določanja prioritet

V praksi se pogosto srečamo z določanjem prioritet opravilom v čakalni vrsti. Široko poznane so preproste metode, kot sta na primer FIFO (angl. First-In First-Out) in LIFO (angl. Last-In First-Out). Pri zagotavljanju varnosti je osnovni kriterij pri določanju prioritet raven grožnje, ki jo ranljivost predstavlja za IS.

V ta namen se najpogosteje uporablja osnovna ocena CVSS (angl. CVSS Base score), ki smo jo predstavili v poglavju 3.3.2. Žal je ne moremo enačiti z oceno tveganja, saj visoka ocena ne pomeni nujno, da je verjetnost njenega izkoriščanja visoka. Enako velja tudi za oceno VRSS, ki se podobno kot CVSS izračuna na osnovi atributov CVSS (glej 3.3.3). Glavna razlika med omenjenima metodama je v načinu izračuna ocene, ni pa povsem jasno, katera je bližje dejanskemu tveganju, ki ga ranljivost predstavlja za IS.

Metodo FIFO lahko enačimo z naključnim pristopom. Vsaka nova odkrita ranljivost v IS ima povsem poljubno vrednost atributov in ni odvisna od svojih predhodnic. Posledično to pomeni, da z odstranjevanjem ranljivosti v istem vrstnem redu, kot so bile odkrite, zmanjšujemo tveganje naključne velikosti.

[1] http://cve.mitre.org/data/refs/refmap/source-EXPLOIT-DB.html
[2] http://contagiodump.blogspot.si/2010/06/overview-of-exploit-packs-update.html

5.4 Metode na osnovi upoštevanja lastnosti napadalcev

Upoštevanje lastnosti napadalcev odpira možnosti za izgradnjo več novih metod. Po naših predvidevanjih si lahko obetamo izboljšave predvsem pri metodah, ki so osnovane na funkciji izkoriščanja (glej definicijo 4.3). V našem primeru je to metoda HTAC, ki daje prednost tisti ranljivosti, ki jo lahko izkoristi največ napadalcev (angl. Highest Threat Agent Count). Kadar ima več ranljivosti isto TAC vrednost, odstranimo najprej tisto, ki je bila prej odkrita.

Poleg metode HTAC je smiselno preveriti še druge metode, ki upoštevajo TAC število. Več različnih vektorjev CVSS ima lahko povsem enako oceno CVSS. V takem primeru je zato smiselno dati prednost tisti ranljivosti, ki ima višjo vrednost TAC. Metodo, ki sledi temu pravilu, bomo poimenovali FSTA (angl. First-Score Then-Agents). Podobno ima lahko več različnih CVSS vektorjev enako vrednost TAC. V takem primeru je smiselno dati prednost tisti ranljivosti, ki ima višjo oceno CVSS. Metodo, ki temelji na tem pravilu, bomo poimenovali FATS (angl. First-Agents Then-Score).

V raziskavi smo v prvi vrsti osredotočeni na merjenje učinkovitosti metod, ki upoštevajo lastnosti napadalcev (metode HTAC, FATS, FTSA), ter njihovo primerjavo z obstoječimi metodami. V tabeli 5.1 so zbrane vse metode, ki jih bomo primerjali med seboj.

Drugo zelo zanimivo vprašanje je, koliko so vse omenjene metode boljše od naključnega vrstnega reda odstranjevanja. Nekateri raziskovalci namreč navajajo, da je uporaba ocene CVSS enako učinkovita kot naključni pristop. Poleg omenjenih vprašanj pa nas zanimajo tudi napovedne sposobnosti posameznih metod glede izkoristljivih ranljivosti. Zato bomo z eksperimentom statistično analizirali, kako hitro lahko posamezna metoda odstrani izkoristljive ranljivosti.

5.5 Eksperimentalno okolje

Primerjavo učinkovitosti metod moramo izvesti v okolju, ki vsem metodam zagotavlja povsem enakovredne pogoje. V resničnem okolju to ni izvedljivo, zato si lahko pomagamo s simulacijami.

Eksperimentalno okolje, v katerem bomo izvajali primerjave, mora odražati lastnosti sodobnih informacijskih sistemov. Vemo, da so ti danes sestavljeni iz več različnih programskih modulov različnih proizvajalcev. Programski moduli se med seboj razlikujejo

Tabela 5.1

Opredelitev vseh metod določanja prioritet ranljivostim pri odstranjevanju, ki jih bomo primerjali v eksperimentu.

Metoda	*Opis*
FIFO	Prva odkrita ranljivost ima najvišjo prioriteto.
CVSS	Ranljivost z najvišjo CVSS vrednostjo ima najvišjo prioriteto.
VRSS	Ranljivost z najvišjo VRSS vrednostjo ima najvišjo prioriteto.
HTAC	Ranljivost z najvišjo TAC vrednostjo ima najvišjo prioriteto.
FSTA	Najprej najvišja CVSS vrednost, potem najvišja TAC vrednost.
FATS	Najprej najvišja TAC vrednost, potem najvišja CVSS vrednost.

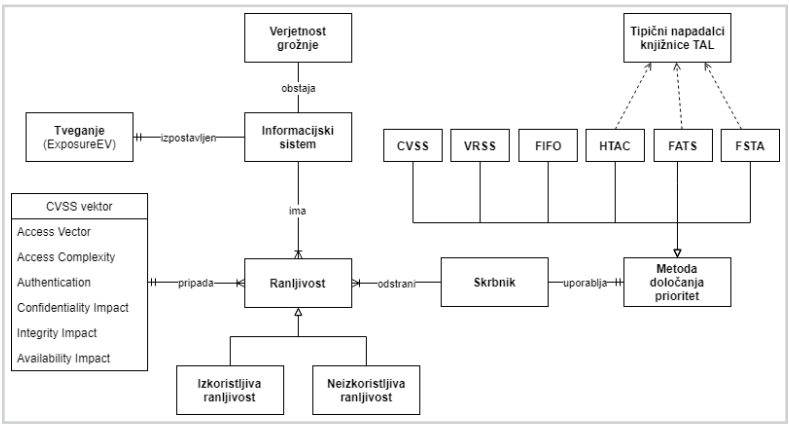
tako po številu ranljivosti kot tudi po tipu odkritih ranljivosti. Zaradi široke ponudbe programske opreme, ki je razvita na osnovi številnih informacijskih tehnologij ter različnih programskih praks, je nabor ranljivosti nepredvidljiv. Zato je za izvedbo eksperimentalnega IS smiselno uporabiti naključne ranljivosti iz zbirke odkritih ranljivosti, ki so zbrane v NVD.

5.6 Izvedba simulacijskega modela

Simulacijski model smo razvili na programski platformi Repast Symphony (Recursive Porous Agent Simulation Toolkit) z uporabo programskega jezika Java. Repast je prosto uporabno odprtokodno orodje za modeliranje in izvajanje simulacij na osnovi agentnih metod [81]. UML razredni diagram simulacijskega modela je prikazan na sliki 5.1.

Simulacijsko okolje smo zgradili po naslednjem postopku:

1. Kreiranje spremenljivke IS s praznim seznamom ranljivosti.
2. Dodajanje 1.000 naključno izbranih ranljivosti iz podatkovne zbirke NVD v spremenljivko IS.
3. Označevanje izkoristljivih ranljivosti v spremenljivki IS. Izkoristljive so vse ranljivosti, za katere v podatkovni zbirki EDB obstajajo postopki izkoriščanja. V nadaljevanju jih imenujemo PoC ranljivosti (angl. PoC, Proof Of Concept).
4. Dodajanje agentov varnostnikov (angl. security guard) v simulacijski model. Za vsako metodo določanja prioritet iz tabele 5.1 dodamo po enega varnostnika.



Slika 5.1

UML razredni diagram simulacijskega modela za merjenje učinkovitosti metod določanja prioritet ranljivostim. S simulacijskim modelom preko spremenljivke $ExposureEV_k$ merimo izpostavljenost IS grožnjam, tj. izkoristljivim ranljivostim.

Naloga varnostnika je dosledno izvajanje izbrane metode odstranjevanja ranljivosti.

5. Vsakemu agentu varnostniku dodelimo svoj klon spremenljivke IS s povsem enakim naborom ranljivosti. S tem zagotovimo povsem enakovredne pogoje za vse primerjane metode.

Učinkovitost metod bomo merili s kazalnikom $ExposureEV_k$, ki je opredeljen z enačbo (5.1). Vsako izvajanje simulacije bo zajemalo 1.000 korakov. Na vsakem koraku simulacije bo vsak varnostnik v svojem IS skladno s svojo metodo odstranjevanja odstranil ranljivost, ki predstavlja največjo grožnjo. Po 1.000 izvedenih korakih bodo vse ranljivosti odstranjene, saj med simulacijo ne bomo dodajali novih ranljivosti. Glede na opredelitev kazalnika $ExposureEV_k$ bomo na vsakem koraku simulacije opazovali, kolikšno je preostalo število PoC ranljivosti v IS. Metoda, ki to število hitreje zmanjšuje, je učinkovitejša.

5.7 Uporabljeni podatki in postopek izvedbe eksperimenta

V eksperimentu smo uporabili podatke iz podatkovne zbirke NVD iz obdobja od leta 2010 do leta 2016. Podatki o ranljivostih tega obdobja so podani v obliki, ki jo določa CVSSv2 različica standarda. V omenjenem obdobju je bilo skupaj odkritih in javno objavljenih 41.823 ranljivosti. Med njimi je 2.416 izkoristljivih ranljivosti (PoC

ranljivosti), saj zanje v podatkovni zbirki EDB obstajajo dokazi o obstoju postopkov izkoriščanja.

Posebej smo ovrednotili še 95 ranljivosti iz istega časovnega obdobja, ki jih najdemo v podatkovni zbirki EKITS. Zanje torej velja, da so bile dokazano izkoriščane v realnosti, saj so del celovitih rešitev za izkoriščanje, ki jih je možno kupiti na črnem trgu.

Vrednotenje predlaganega modela je potekalo po naslednjih korakih:

- Izračun vrednosti TAC za vse možne vektorje CVSS.
- Izračun števila izkoriščanih CVSS vektorjev po posameznih napadalcih knjižnice TAL.
- Izračun števila ranljivosti v zbirkah NVD, EDB in EKITS s posameznimi vrednostmi TAC.
- Analiza vrednosti TAC najpogostejših vektorjev CVSS v zbirkah NVD, EDB in EKITS.
- Primerjava učinkovitosti posameznih metod za določanje prioritet ranljivostim na osnovi vrednotenja kazalnika $ExposureEV_k$.
- Vrednotenje napovednih sposobnosti posameznih metod z ugotavljanjem števila odstranjenih izkoristljivih ranljivosti v posameznih kvartalih simulacije.

5.8 Evalvacija

5.8.1 Analiza izkoriščanja

V analizi izkoriščanja nas zanima, koliko napadalcev knjižnice TAL je zmožnih izkoriščati posamezne vektorje CVSS in nato dejanske ranljivosti s temi vektorji. Vektor CVSS je sestavljen iz 6 atributov, ki lahko skupaj tvorijo 729 različnih kombinacij. Skrajno levi stolpec tabele 5.2 prikazuje vse možne vrednosti TAC, ki temeljijo na 21 grožnjah (napadalcih) knjižnice TAL. TAC nikoli ni presegel vrednosti 18. Stolpec *"CVSS vektor"* predstavlja število vektorjev CVSS, ki glede na svoje atribute dosežejo isto vrednost TAC. Desni trije stolpci v tabeli 5.2 se nanašajo na dejanske ranljivosti. Stolpec *NVD* prikazuje število ranljivosti iz podatkovne zbirke NVD z isto vrednostjo

Tabela 5.2

Število vektorjev CVSS in dejanskih ranljivosti v podatkovnih zbirkah NVD, EDB in EKITS, ki jih lahko izkorišča enako število napadalcev (vrednost TAC). Podatki v prvi vrstici na primer pomenijo, da obstajajo 4 vektorji CVSS, ki jih lahko izkoristi vseh 18 napadalcev knjižnice TAL. Poleg tega je v zbirki NVD 4.085, v zbirki EDB 793, v zbirki EKITS pa so 4 ranljivosti, ki jih lahko izkoristi vseh 18 napadalcev knjižnice TAL.

TAC	*CVSS vektor*	*NVD*	*EDB*	*EKITS*
18	4	4.085	793	4
17	4	401	10	
16	4	3.257	137	41
15	8	49	2	
14	5	2.361	162	1
13	16	5.266	272	
12	10	226	16	1
11	17	4.951	216	44
10	18	2.174	75	
9	18	434	22	1
8	18	1.636	96	1
7	25	702	24	
6	29	1.608	33	
5	40	6.235	314	1
4	82	3.342	131	1
3	120	2.265	54	
2	74	1.506	47	
1	86	743	9	
0	151	582	3	
Vsota	729	41.823	2.416	95

TAC, stolpec *EDB* se nanaša na ranljivosti iz EDB in stolpec *EKITS* na ranljivosti, ki se dejansko izkoriščajo v realnosti.

Analizo izkoriščanja smo naredili tudi na napadalcih. V tabeli 5.3 so predstavljeni vsi arhetipi napadalcev knjižnice TAL, ki smo jih uporabili v eksperimentu. Poleg vsakega napadalca je navedeno število vektorjev CVSS, ki jih je posamezen napadalec zmožen izkoristiti.

Tabela 5.3

Arhetipi napadalcev knjižnice TAL s številom vektorjev CVSS, ki jih je vsak posameznik zmožen izkoristiti.

TAL arhetip	Opis	*Št. CVSS vektorjev*
Employee Recless	neroden zaposlenec	0
Employee Untrained	neusposobljen zaposlenec	0
Info Partner	informacijski partner	0
Anarchist	anarhist	108
Civil Activist	civilni aktivist	108
Competitor	tekmec	108
Corupt Government Official	podkupljiv vladni uradnik	288
Data Miner	podatkovni rudar	72
Employee Disgruntled	nezadovoljen zaposlenec	144
Government Cyberwarrior	vladni kibernetski bojevnik	432
Government Spy	vladni vohun	162
Internal Spy	notranji vohun	162
Irrational Individual	neracionalen posameznik	78
Legal Adversary	pravni nasprotnik	162
Mobster	kriminalec, mafijec	144
Radical Activist	radikalni aktivist	414
Sensationalist	senzacionalist	138
Terrorist	terorist	156
Thief	tat	44
Cyber Vandal	kibernetski vandal	92
Vendor	prodajalec	54

5.8.2 Analiza vrednosti TAC na posameznih zbirkah ranljivosti

Učinkovitost metod za določanje prioritet pri odstranjevanju ranljivosti je veliko bolj odvisna od pogostih vektorjev CVSS kot od tistih, ki se redko pojavljajo. Poleg tega si vektorji CVSS, ki se pogosto pojavljajo v zbirki ranljivosti EDB, zaslužijo več pozornosti kot tisti, ki se redkeje pojavljajo v njej. Zato smo obe podatkovni zbirki ovrednotili glede na frekvence posameznih vektorjev CVSS kot tudi na njihove TAC in CVSS vrednosti. Najpogostejši vektorji CVSS iz zbirke ranljivosti NVD so navedeni v tabeli

Tabela 5.4

Vektorji CVSS z več kot 1.000 ranljivostmi v podatkovni zbirki NVD s pripadajočima vrednostma TAC in CVSS ter deležem PoC ranljivosti.

CVSS vektor	*Število*	*TAC*	*CVSS*	*% PoC*
AV:N/AC:M/Au:N/C:N/I:P/A:N	4.764	5	4,3	5,79 %
AV:N/AC:L/Au:N/C:P/I:P/A:P	4.052	18	7,5	19,57 %
AV:N/AC:M/Au:N/C:C/I:C/A:C	3.672	11	9,3	5,45 %
AV:N/AC:L/Au:N/C:C/I:C/A:C	3.229	16	10	4,24 %
AV:N/AC:M/Au:N/C:P/I:P/A:P	3.213	13	6,8	8,22 %
AV:N/AC:L/Au:N/C:P/I:N/A:N	2.353	14	5	6,88 %
AV:N/AC:L/Au:N/C:N/I:N/A:P	1.967	10	5	3,20 %
AV:A/AC:M/Au:N/C:P/I:P/A:P	1.423	13	5,4	0,00 %
AV:L/AC:L/Au:N/C:C/I:C/A:C	1.103	4	7,2	6,71 %
AV:N/AC:M/Au:N/C:N/I:N/A:P	1.056	5	4,3	2,84 %
AV:N/AC:M/Au:N/C:P/I:N/A:N	1.050	11	4,3	0,76 %
Vsota	27.882			

5.4, najpogostejši vektorji CVSS iz EDB v tabeli 5.5 in vektorji CVSS iz zbirke EKITS v tabeli 5.6. Pri vseh tabelah je v stolpcu *% PoC* predstavljen delež izkoristljivih ranljivosti (PoC ranljivosti) v zbirki NVD, ki temeljijo na podanem CVSS vektorju. Delež izkoristljivih ranljivosti je izračunan glede na podatke o obstoju postopkov izkoriščanja za posamezne ranljivosti v podatkovni zbirki EDB.

5.8.3 Analiza učinkovitosti metod

V drugem koraku vrednotenja modela smo primerjali učinkovitost metod za določanje prioritet pri odstranjevanju ranljivosti. Opisali smo jih v podpoglavjih 5.3 in 5.4. Seznam vseh primerjanih metod je v tabeli 5.1. Vrednotenje smo izvedli z uporabo simulacijskega modela, ki smo ga podrobneje predstavili v podpoglavju 5.6.

Učinkovitost politik smo merili s kazalnikom $ExposureEV_k$. Na vsakem koraku simulacije je bila zato glavna pozornost usmerjena na število preostalih izkoristljivih ranljivosti (PoC ranljivosti) v IS. Slika 5.2 prikazuje štetje preostalih PoC ranljivosti v tipičnem procesu simulacije. V predstavljenem primeru je bilo na začetku simulacije v IS 1.000 ranljivosti, od tega 56 PoC ranljivosti. Na sliki opazimo različno uspešnost

Tabela 5.5

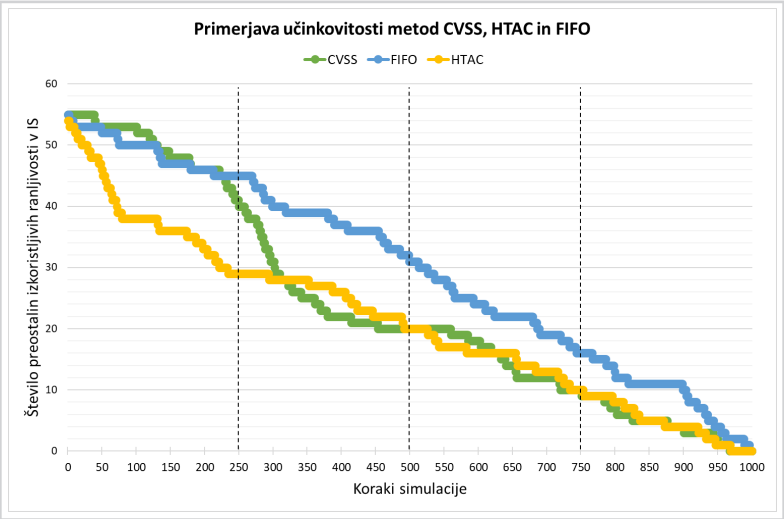
Vektorji CVSS z več kot 30 ranljivostmi v podatkovni zbirki EDB s pripadajočima vrednostma TAC in CVSS ter deležem PoC ranljivosti.

CVSS vektor	*Število*	*TAC*	*CVSS*	*% PoC*
AV:N/AC:L/Au:N/C:P/I:P/A:P	793	18	7,5	19,57 %
AV:N/AC:M/Au:N/C:N/I:P/A:N	276	5	4,3	5,79 %
AV:N/AC:M/Au:N/C:P/I:P/A:P	264	13	6,8	8,22 %
AV:N/AC:M/Au:N/C:C/I:C/A:C	200	11	9,3	5,45 %
AV:N/AC:L/Au:N/C:P/I:N/A:N	162	14	5	6,88 %
AV:N/AC:L/Au:N/C:C/I:C/A:C	137	16	10	4,24 %
AV:N/AC:L/Au:S/C:P/I:P/A:P	79	8	6,5	10,45 %
AV:L/AC:L/Au:N/C:C/I:C/A:C	74	4	7,2	6,71 %
AV:N/AC:L/Au:N/C:N/I:N/A:P	63	10	5	3,20 %
AV:L/AC:M/Au:N/C:C/I:C/A:C	34	2	6,9	5,63 %
Vsota	2.082			

Tabela 5.6

Vektorji CVSS, ki se pojavljajo v zbirki ranljivosti EKITS, s pripadajočima vrednostma TAC in CVSS ter deležem PoC ranljivosti.

CVSS vektor	*Število*	*TAC*	*CVSS*	*% PoC*
AV:N/AC:M/Au:N/C:C/I:C/A:C	43	11	9,3	5,45 %
AV:N/AC:L/Au:N/C:C/I:C/A:C	41	16	10	4,24 %
AV:N/AC:L/Au:N/C:P/I:P/A:P	4	18	7,5	19,57 %
AV:L/AC:L/Au:N/C:C/I:C/A:C	1	4	7,2	6,71 %
AV:N/AC:H/Au:N/C:C/I:C/A:C	1	9	7,6	5,21 %
AV:N/AC:L/Au:N/C:N/I:P/A:N	1	8	5	2,05 %
AV:N/AC:L/Au:N/C:P/I:N/A:N	1	14	5	6,88 %
AV:N/AC:M/Au:N/C:N/I:P/A:N	1	5	4,3	5,79 %
AV:N/AC:M/Au:N/C:P/I:N/A:N	1	11	4,3	0,76 %
AV:N/AC:M/Au:N/C:P/I:N/A:P	1	12	5,8	2,22 %
Vsota	95			



Slika 5.2

Potek tipične izvedbe simulacije. Pri vseh metodah varnostnik začne s povsem enakim IS oz. enakim naborom ranljivosti. V 1.000 korakih vsak iz svojega IS odstrani vse ranljivosti. Razlike med njimi se kažejo v hitrosti odstranjevanja izkoristljivih ranljivosti.

posameznih metod v različnih fazah simulacije. Nekaterim metodam je uspelo odpraviti vse PoC ranljivosti že pred koncem simulacije. Na sliki 5.3 je prikazan simulacijski model med izvajanjem.

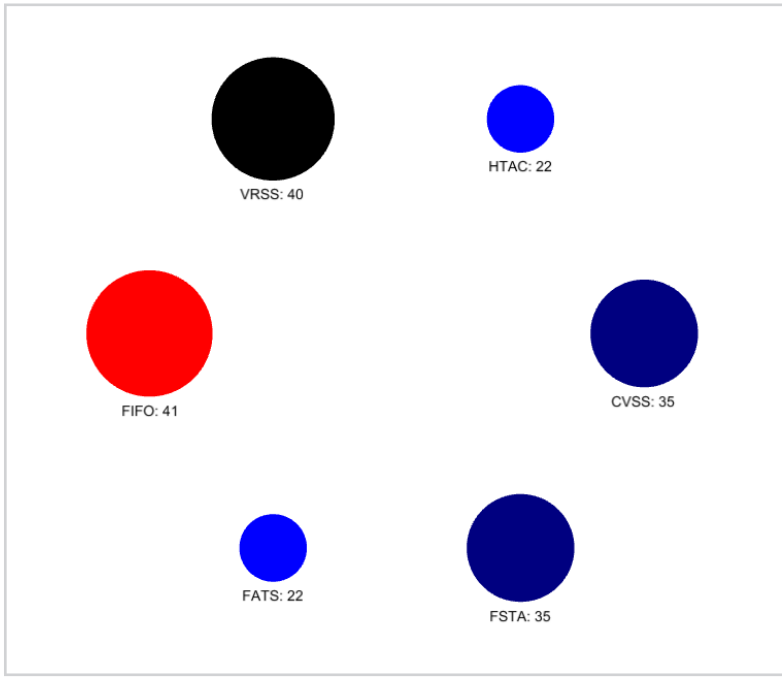
Učinkovitost metod smo z uporabo kazalnika $ExposureEV_k$ merili na vsaki četrtini simulacije: pri 25 % (korak $k = 250$), 50 % ($k = 500$), 75 % ($k = 750$) in 100 % ($k = 1000$). Za vsako metodo smo simulacijo ponovili 1.000-krat. S tem smo pridobili dovolj velik vzorec, da lahko izvedemo statistično sklepanje z vzorca na populacijo. Vsaka izvedba simulacije namreč predstavlja svoj element vzorca, je neodvisna od drugih izvedb simulacij in je enakomerno porazdeljena po vzorcu. Rezultati so predstavljeni v tabeli 5.7. Primerjave povprečnih vrednosti kazalnika $ExposureEV_k$ lahko opravimo z Z-testom.

Na zadnjem koraku vrednotenja modela smo opazovali, koliko PoC ranljivosti je bilo nevtraliziranih v vsakem kvartalu simulacije. Merjenje smo opravili za vsako izmed metod v tabeli 5.1. Metode z boljšimi sposobnostmi predvidevanja izkoristljivih ranljivosti v zgodnejših fazah nevtralizirajo več PoC ranljivosti. Rezultati so predstavljeni v tabeli 5.8.

Tabela 5.7

Izpostavljenost informacijskega sistema PoC ranljivostim, izmerjena z $ExposureEV_k$ kazalnikom na koncu vsake četrtine simulacije. Poleg podatkov o povprečni vrednosti ($\bar{x}$) je podan tudi standardni odklon (σ).

Metoda	*25 %*		*50 %*		*75 %*		*100 %*	
	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
FATS	10.093	1.468	16.054	2.541	19.503	3.265	20.281	3.471
HTAC	10.090	1.468	16.251	2.562	19.714	3.278	20.545	3.494
FSTA	12.821	1.745	19.153	2.751	22.272	3.377	23.056	3.577
CVSS	12.824	1.745	19.177	2.754	22.348	3.381	23.080	3.570
VRSS	12.890	1.754	20.739	2.939	24.115	3.570	24.899	3.761
FIFO	12.579	1.704	21.567	3.033	26.973	3.938	28.749	4.282



Slika 5.3

Primerjava učinkovitosti metod v simulacijskem okolju. Poleg imena vsake metode je navedeno število preostalih izkoristljivih ranljivosti v informacijskem sistemu. Učinkovitejša metoda hitreje odstranjuje izkoristljive ranljivosti.

Tabela 5.8

Število odstranjenih PoC ranljivosti po posameznih metodah, merjeno ob zaključku vsake četrtine izvajanja simulacije. Navedene so povprečne vrednosti. Tabela je razdeljena na štiri odseke. V zgornjem delu je podano skupno število vseh odstranjenih ranljivosti. V spodnjem delu je navedeno število odstranjenih ranljivosti glede na kvalitativno vrednost CVSS, ki ranljivosti deli v kategorije HIGH, MEDIUM in LOW (glej tabelo 3.1).

Metoda	*25 %*	*50 %*	*75 %*	*100 %*
	Vse skupaj			
CVSS	16,93	24,10	8,64	7,88
FATS	26,95	11,73	10,70	8,18
FIFO	14,56	14,20	14,45	14,35
FSTA	16,98	24,03	7,90	8,64
HTAC	26,72	11,81	10,84	8,19
VRSS	12,50	27,77	9,07	8,22
	HIGH			
CVSS	16,93	14,07	-	-
FATS	22,33	5,04	1,55	2,08
FIFO	7,86	7,58	7,77	7,79
FSTA	16,98	14,02	-	-
HTAC	22,28	4,98	1,63	2,11
VRSS	11,52	19,48	-	-
	MEDIUM			
CVSS	-	10,04	8,64	6,54
FATS	4,62	6,68	9,11	4,81
FIFO	6,37	6,29	6,33	6,24
FSTA	-	10,02	7,90	7,30
HTAC	4,43	6,83	9,17	4,79
VRSS	0,98	8,29	8,92	7,04
	LOW			
CVSS	-	-	-	1,33
FATS	-	-	0,04	1,29
FIFO	0,34	0,33	0,34	0,32
FSTA	-	-	-	1,33
HTAC	-	-	0,04	1,29
VRSS	-	-	0,15	1,19

Tabela 5.9

Povprečne, največje in najmanjše TAC vrednosti ranljivosti, ki smo jih uporabili v eksperimentu. Vrednosti so navedene ločeno po analiziranih zbirkah ranljivosti.

TAC	*NVD*	*EDB*	*EKITS*
Povprečje	5,0	10,1	10,8
Največja vrednost	18	18	18
Najmanjša vrednost	0	2	4

5.9 Razprava

5.9.1 Vrednosti TAC

Z analizo izkoriščanja smo želeli ugotoviti, koliko napadalcev knjižnice TAL je sposobnih izkoriščati posamezne vektorje CVSS. Rezultati v tabeli 5.3 kažejo, da je le 18 napadalcev izmed 21 zmožnih izkoristiti vsaj en vektor CVSS. Iz tega izhaja, da je v našem eksperimentu 18 tudi najvišja možna vrednost TAC. V tabeli 5.2 razberemo, da to vrednost dosežejo štirje vektorji CVSS. Druga pomembna ugotovitev je, da ima manj kot 10 % vektorjev CVSS vrednost TAC višjo od 10 (68 od 729 vektorjev). Poleg tega ima največ vektorjev CVSS vrednost TAC enako nič. Takšnih je več kot 20 % oz. 151 vektorjev.

Pri dejanskih ranljivostih (stolpci NVD, EDB in EKITS v tabeli 5.2) je slika precej drugačna. Ugotavljamo, da ima velika večina ranljivosti v analiziranih podatkovnih zbirkah vrednost TAC večjo od nič. Iz tega lahko sklepamo, da se v realnih okoljih redko pojavljajo ranljivosti z vektorji CVSS, katerih vrednost TAC je enaka nič. Po drugi strani ima 67 % ranljivosti v zbirki EDB in kar 96 % v zbirki EKITS vrednost TAC večjo od 10. V zbirki NVD je delež ranljivosti s tako visoko vrednostjo TAC precej nižji, in sicer 50 %. Povprečna vrednost TAC je torej višja v zbirkah EDB in EKITS, ki predstavljata izkoristljive ranljivosti (glej tabelo 5.9). Ugotavljamo tudi, da med izkoristljivimi ranljivostmi ni nobene, ki bi imela vektor CVSS z vrednostjo TAC enak 0. Na osnovi rezultatov analize lahko zaključimo, da imajo izkoristljive ranljivosti v povprečju višje vrednosti TAC kot neizkoristljive.

Posebno pozornost si zasluži CVSS vektor `[AV:N/AC:M/Au:N/C:N/I:P/A:N]`. V tabeli 5.4 se nahaja v prvi vrstici. Njegova TAC vrednost je 5, kar je dokaj nizka vrednost in pomeni, da ga lahko izkoristi dokaj malo napadalcev iz knjižnice TAL. Kljub

vsemu je to najpogostejši vektor CVSS v zbirki NVD, saj je z njim opisanih kar 4.764 ranljivosti. Med njimi je kar 5,79 % izkoristljivih, saj jih najdemo tudi v zbirki EDB (glej stolpec *% PoC*). Podrobnejši opis ranljivosti v NVD razkrije, da je kar 3.607 teh ranljivosti oz. 80 % tipa XSS (angl. Cross Site Scripting). Pri tem je zanimivo, da več avtorjev na osnovi svojih raziskav trdi, da so ranljivosti XSS opisane z napačnim vektorjem CVSS. V osnovi je ranljivostim XSS pripisano delno ogrožanje celovitosti IS. Po prepričanju avtorjev pa ranljivosti XSS delno ogrožajo tudi zaupnost in razpoložljivost [13, 68]. Primernejši bi torej bil vektor `[AV:N/AC:M/Au:N/C:P/I:P/A:P]`, ki ga v tabeli 5.4 najdemo v peti vrstici. Njegova vrednost TAC je 13, kar pomeni, da ga lahko izkoristi mnogo več napadalcev iz knjižnice TAL.

Zelo pogost vektor CVSS v zbirki NVD je tudi `[AV:A/AC:M/Au:N/C:P/I:P/A:P]`. Z njim je opisanih kar 1.423 ranljivosti (osma vrstica v tabeli 5.4). Njegova vrednost TAC je 13, kar je dokaj visoka vrednost, saj s tem vektor sodi med 10 % tistih z najvišjo vrednostjo. Kljub vsemu v zbirki izkoristljivih ranljivosti EDB ne najdemo enega samega takšnega vektorja (vrednost 0,00 % v stolpcu *% PoC*). Podrobnejši pregled opisa ranljivosti v zbirki NVD razkrije, da je bila velika večina teh ranljivosti (1.380 ranljivosti oz. 97 %) razkrita v relativno kratkem obdobju, in sicer v zadnjih treh mesecih leta 2014. Poleg tega se vse te ranljivosti nanašajo na isto težavo. NVD navaja naslednji opis: *"library does not verify X.509 certificates from SSL servers, which allows man-in-the-middle attackers to spoof servers and obtain sensitive information via a crafted certificate"*. Na osnovi opisa lahko sklepamo, da imajo vse te ranljivosti isti vzrok ter da je bil odziv ponudnikov programske opreme na to grožjo hiter in masoven. Na podlagi podatkov v zbirki NVD ugotavljamo, da je omenjeni vektor v drugih časovnih obdobjih dokaj redek.

5.9.2 Učinkovitost metod

Rezultati v tabeli 5.7 dokazujejo, da je uporaba vrednosti TAC za namene določanja prioritet pri odstranjevanju ranljivosti koristna. Metoda HTAC v povprečju zagotavlja 9 % nižjo izpostavljenost PoC ranljivosti kot metoda CVSS. V prvem kvartalu simulacije je metoda HTAC v povprečju celo za 20 % učinkovitejša od metode CVSS. Metoda FATS, ki je podobna metodi HTAC, a v primeru enakih vrednosti TAC upošteva tudi oceno CVSS, dosega celo nekoliko boljše rezultate. Kljub vsemu statistično gledano med njima ni razlik. Podobno situacijo opazimo, ko med seboj primerjamo metodi CVSS in FSTA. Zanimiva je primerjava metod CVSS in VRSS, ki po nam znanih

podatkih edini izračunavata kvantitativno oceno grožnje, ki jo predstavlja ranljivost. Izkazalo se je, da so ocene CVSS točnejše od ocen VRSS. Natančneje gledano je metoda VRSS v povprečju za 6 % slabša od metode CVSS. Najnižjo učinkovitost smo zabeležili pri metodi FIFO, saj je v povprečju za kar 22 % slabša od metode CVSS. Pri vseh primerjavah smo uporabili Z-test za povprečja nad dvema vzorcema, pri čemer smo upoštevali, da je $\alpha = 0,01$ (glej Dodatek A).

Podrobnejši pregled po posameznih časovnih odsekih simulacije razkrije, da je metoda FIFO v povprečju celo nekoliko boljša od metode CVSS v prvem kvartalu (glej tabelo 5.7). Razlog za to tiči v dejstvu, da metoda CVSS v začetku odstranjuje izključno ranljivosti z najvišjo oceno CVSS, po podatkih v tabeli 5.5 pa vemo, da PoC ranljivosti nimajo vedno najvišjih ocen CVSS. Po drugi strani metoda FIFO ves čas odstranjuje ranljivosti s poljubnimi ocenami CVSS in očitno v povprečju v prvem kvartalu odstrani več PoC ranljivosti. Iz tabele 5.8 je razvidno, da metoda CVSS v prvem kvartalu odstranjuje izključno ranljivosti kategorije HIGH, medtem ko metoda FIFO že v prvem kvartalu odstranjuje tudi ranljivosti kategorij MEDIUM in LOW.

Večjo učinkovitost metod HTAC in FATS lahko pripišemo dejstvu, da imata boljše sposobnosti predvidevanja izkoristljivih ranljivosti. Ključno je, da obe metodi že v prvem kvartalu simulacije nevtralizirata več PoC ranljivosti od ostalih metod (glej tabelo 5.8). V začetku simulacije je namreč v IS največ izkoristljivih ranljivosti, kar pomeni, da je IS izpostavljen največjim varnostnim tveganjem. Medtem ko metoda CVSS odstranjuje ranljivosti po vrsti od HIGH proti LOW, odstranjuje metoda HTAC že v prvem kvartalu tudi ranljivosti, ki sodijo v kategorijo MEDIUM. Ugotavljamo torej, da je vrednost TAC dober indikator izkoristljivih ranljivosti. Večjo kot ima ranljivost vrednost TAC, večja je verjetnost obstoja postopka njenega izkoriščanja.

5.9.3 Omejitve predstavljenega modela

Model ima nekatere omejitve. Simulacijsko okolje, ki smo ga razvili za potrebe eksperimenta, je le približek situacije v realnosti. Zavedati se moramo, da so napadalci skrita populacija in da o njihovih lastnostih (namenih, obnašanju) lahko le sklepamo. Poleg tega je uporabljena knjižnica TAL le eden izmed možnih opisov napadalcev. Obstajajo namreč še druge zanimive lastnosti, ki jih omenjena knjižnica ne obravnava. S stalnim spreminjanjem informacijsko-komunikacijskih tehnologij se spreminjajo tudi napadalci. Skupina tipičnih napadalcev se torej v prihodnje lahko spremeni tako po velikosti kot posameznih lastnostih.

Med lastnostmi, ki bi lahko koristile pri dodatnem izboljševanju učinkovitosti metode določanja prioritet, je velikost populacije posamezne skupine napadalcev. V predstavljenem modelu smo namreč upoštevali le dejstvo, da različne skupine napadalcev obstajajo, a jih v izračunih obravnavamo enakovredno. Predvidevamo sicer, da porazdelitev skupin ni enakomerna, a ne vemo natančno, kakšna je. V tej situaciji smo se zato odločili, da velikosti populacij ne upoštevamo, kar predstavlja glavno pomanjkljivost predstavljenega modela.

5.10 Zaključek

V tem poglavju smo predstavili več možnih metod določanja prioritet ter opredelili kazalnik $ExposureEV_k$, ki ga uporabimo za merjenje njihove učinkovitosti. Da bi se lahko prepričali, katera izmed metod je najučinkovitejša, smo razvili ustrezen teoretični okvir in ga implementirali z agentnim okoljem. Ob tem smo opredelili zahteve, ki jih mora izpolniti eksperimentalno okolje, da bi odražalo dovolj realistično sliko sodobnega IS in hkrati vsem metodam zagotavljajo povsem enakovredne pogoje vrednotenja.

Z izvedbo eksperimenta v simulacijskem okolju smo preverili učinkovitost predlaganih metod, ki upoštevajo lastnosti napadalcev (HTAC, FATS in FTSA), in jih primerjali z drugimi obstoječimi metodami (CVSS, VRSS in FIFO). Ugotovili smo, da so predlagane metode, ki pri določanju prioritet ranljivostim upoštevajo lastnosti napadalcev, učinkovitejše od metode CVSS, ki je danes najpogosteje uporabljena.

6

Sistem ocenjevanja ranljivosti na osnovi agentov

6.1 Uvod

V prejšnjih poglavjih smo ugotovili, da upoštevanje lastnosti napadalcev pri določanju prioritet ranljivostim prispeva k zmanjševanju tveganj. Kljub vsemu smo z uporabo knjižnice TAL v predlaganih metodah zajeli le najtipičnejše predstavnike napadalcev. Z vključevanjem dodatnih lastnosti populacije bi morda lahko še izboljšali napovedne sposobnosti naše metode.

V tem poglavju zato predstavimo izboljšavo metode z uporabo velike naključno ustvarjene populacije napadalcev. Najprej s simulacijo na agentnem modelu prepoznamo, kako široki populaciji napadalcev so izpostavljene ranljivosti z določenimi vektorji CVSS. Velikost populacije ocenimo z izračunom frekvenc izkoriščanja posameznih vektorjev CVSS s strani naključnih napadalcev. Pridobljene frekvence uporabimo pri določanju prioritet ranljivostim, pri čemer imajo vektorji z višjimi frekvencami prednost pri nevtralizaciji. Predlagano metodo v nadaljevanju primerjamo z drugimi obstoječimi metodami. V ta namen ponovimo eksperiment primerjave učinkovitosti metod, ki smo ga predstavili v poglavju 5. Poglavje zaključimo s predstavitvijo rezultatov in razpravo. V zaključku opredelimo sistem ocenjevanja ranljivosti ABVS (angl. Agent Based Vulnerability Score), ki predstavlja raven tveganja izkoriščanja posamezne ranljivosti s podanim vektorjem CVSS.

6.2 Ideja izboljšave

Knjižnica TAL vsebuje le 18 arhetipov napadalcev, ki predstavljajo dejansko grožnjo za izkoriščanje ranljivosti (glej tabelo 5.3). Razlike med njimi so zelo velike. Najnevarnejši napadalec *Government Cyberwarrior* (vrednost 432 v tabeli 5.3 v stolpcu *Št. vektorjev CVSS*) je zmožen izkoristiti kar 380 več vektorjev CVSS kot najmanj nevaren arhetip z imenom *Vendor* (52).

Pri določanju prioritet ranljivostim nas veliko bolj kot razlike med napadalci zanimajo razlike med vektorji CVSS. V tabeli 5.2 vidimo, da se ti glede na vrednosti TAC med sabo zelo malo ločijo. Zavzamejo lahko le vrednosti na intervalu od 0 do 18. Potrebujemo torej sistem ocenjevanja ranljivosti, ki bo bolj ločil ranljivosti med seboj in bo skrbniku IS nudil bolj oprijemljivo informacijo o tem, kako nevarne so posamezne ranljivosti.

Rezultati v poglavju 5 so potrdili našo domnevo, da višja vrednost TAC označuje večjo grožnjo. Pomeni namreč, da je dani vektor zmožna izkoristiti večja populacija

napadalcev. Vprašanje, na katerega še ne znamo odgovoriti, je, kako velike razlike v populacijah napadalcev za posamezni vektor CVSS obstajajo.

Napadalci so skrita populacija, katere velikost težko ocenimo. Kljub vsemu ima vsak posameznik v njej svoje lastnosti. Podobno kot so v knjižnici TAL arhetipi opisani z naborom opisnih atributov, bi lahko tudi vsakega konkretnega napadalca opisali z njimi. Opis pravega napadalca z osmimi atributi sicer ni popoln, a je dovolj dober, da lahko ugotovimo napadalčevo zmožnost izkoriščanja neke konkretne ranljivosti, le ustrezno moramo ovrednotiti funkcijo izkoriščanja, ki smo jo opredelili z izrazom 4.3 v poglavju 4.

Ker konkretnih napadalcev ne poznamo, je edina možnost, da jih simuliramo. Žal ne vemo, kako so v populaciji porazdeljene njihove sposobnosti. Prav tako ne poznamo njihovih ciljev, motivacije, omejitev, pričakovanj ipd. V našem primeru nas zanima predvsem njihova raznolikost. Zato lahko napadalce nad izbranim naborom opisnih atributov ustvarimo naključno. Ob tem moramo paziti, da jih ustvarimo dovolj, da z njimi zajamemo njihovo čim večjo raznolikost.

Na osnovi atributov knjižnice TAL lahko ustvarimo končno število različnih napadalcev. Ob upoštevanju vseh možnih kombinacij, predvsem pri večvrednostnih atributih, ta številka naraste na nekaj milijonov. Ustvariti moramo torej simulacijsko okolje, kjer bomo preverili sposobnosti naključno ustvarjenih napadalcev nad realnimi ranljivostmi, ki so zbrane v zbirki NVD. Velikost populacije uspešnih napadalcev nad izbranim vektorjem CVSS bo merilo tveganja. Po naših pričakovanjih je namreč bolj verjetno, da že obstajajo ali bodo razviti postopki izkoriščanja za tiste vektorje CVSS, kjer je večja populacija zmožnih napadalcev.

6.3 Simulacijsko okolje za oceno izkoriščanja vektorjev CVSS

Osnovni cilj simulacije je ugotoviti, koliko naključnih napadalcev je zmožnih izkoristiti posamezen vektor CVSS. Pridobljene frekvence izkoriščanj nam bodo kasneje služile pri razvoju metode za določanje prioritet ranljivostim.

Napadalci bodo naključno izbirali ranljivosti iz podatkovne zbirke NVD in ocenjevali svoje zmožnosti izkoriščanja. V zbirki NVD je veliko ranljivosti, ki pripadajo istemu vektorju CVSS. Nekateri vektorji so pogosteje zastopani, zato zanje obstaja večja verjetnost, da bodo v simulaciji izbrani. Razlog, da napadalci izbirajo ranljivosti iz zbirke NVD, je v tem, da preverjamo zmožnost izkoriščanja in ne zmožnosti odkrivanja ranljivosti. Če bi preverjali zmožnost odkrivanja, bi jo preverjali na množici vseh

možnih vektorjev CVSS in tam bi imel vsak vektor enako verjetnost izbora. Zbirka NVD torej odraža dejanski rezultat odkrivanja. Povsod, kjer so bili napadalci uspešnejši pri odkrivanju, so številke vektorjev CVSS višje. S simulacijo bomo torej izmerili frekvenco izkoriščanja posameznih vektorjev CVSS.

Simulacijsko okolje smo zgradili na naslednji način:

- Ustvarili smo informacijski sistem, ki vsebuje vse ranljivosti iz zbirke NVD od leta 2010 do leta 2016 (skupaj 41.823 ranljivosti).
- Ustvarili smo 100 napadalcev in jih opisali z atributi knjižnice TAL. Atributi so naključno izbrani. Naključna števila pri izboru vseh atributov sledijo enakomerni porazdelitvi.
- Vsak napadalec je sposoben naključno izbrati poljubno ranljivost v informacijskem sistemu in jo ovrednotiti po funkciji izkoriščanja, ki smo jo opredelili z izrazom 4.3.

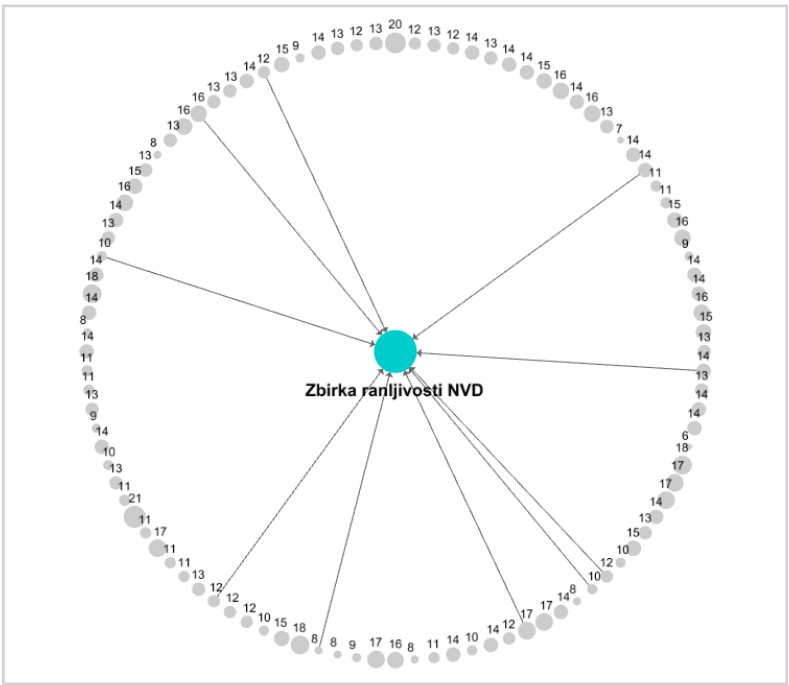
Simulacija deluje po naslednjem postopku:

1. V začetku simulacije vse frekvence vektorjev CVSS postavimo na vrednost 0.
2. V vsaki urini periodi simulacije vsak izmed napadalcev naključno izbere eno ranljivost v informacijskem sistemu in jo ovrednoti. Če je zmožen izkoristiti izbrano ranljivost, poveča števec pripadajočega vektorja CVSS za 1. Ranljivosti ves čas ostajajo v sistemu in se tudi po uspešnem ovrednotenju ne odstranijo.
3. Po vsakem ovrednotenju napadalec naključno spremeni svoje atribute, ali z drugimi besedami, v vsaki urini periodi dobimo 100 drugačnih napadalcev.

Simulacija traja 5.000 urinih period, kar pomeni, da v celotnem poteku ustvarimo 500.000 različnih napadalcev. Na sliki 6.1 je prikaz simulacijskega modela med izvajanjem.

6.4 *Metoda za določanje prioritet na osnovi agentov*

S simulacijo pridobljene frekvence izkoriščanja posameznih vektorjev CVSS želimo v nadaljevanju preveriti, če so uporabne pri določanju prioritet ranljivostim. Pred izvedbo eksperimenta ne moremo vedeti, kako visoke bodo te vrednosti. Osnovno



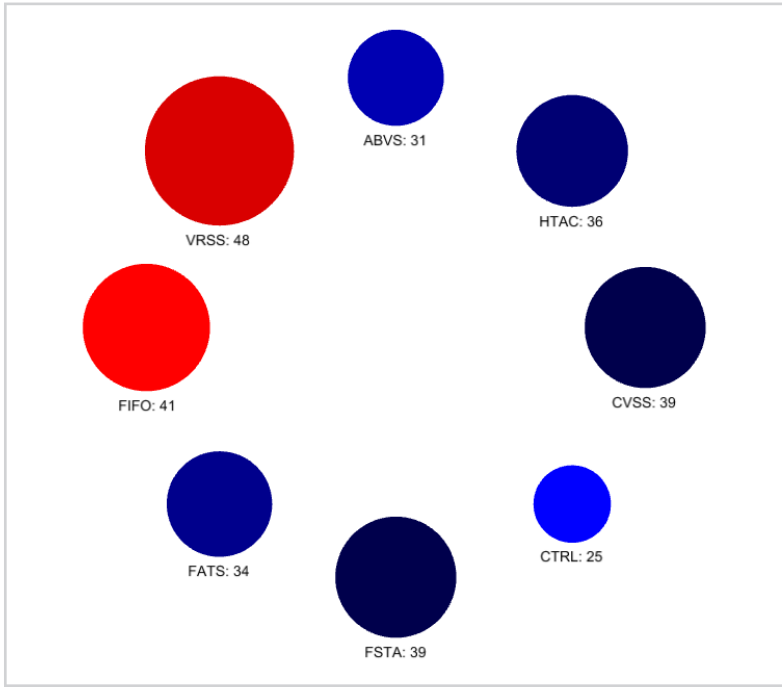
Slika 6.1

Simulacijski model za izračun frekvenc izkoriščanja vektorjev CVSS v podatkovni zbirki NVD med izvajanjem. V središču je informacijski sistem, ki vsebuje vse ranljivosti zbirke NVD od leta 2010 do leta 2016. Okrog sistema so naključno ustvarjeni napadalci, predstavljeni z različno velikimi pikami. Velikost pike je odvisna od vrednosti napadalčevih atributov (vsota vrednosti urejenostnih atributov). Večja pika predstavlja napadalca z močnejšimi atributi oz. večjo grožnjo. Puščica v smeri od napadalca k sistemu pove, da je napadalec zmožen izkoristiti naključno najdeno ranljivost v NVD.

vodilo je, da višja frekvenca pomeni večjo grožnjo, saj jo lahko izkorišča več napadalcev. Prednost bomo torej dajali ranljivostim z višjo frekvenco.

Ker pri opisani metodi frekvence izkoriščanja vektorjev CVSS izračunamo s pomočjo simulacijskega modela na osnovi agentov in ker bo to tudi osnova novega sistema ocenjevanja ranljivosti, smo metodo poimenovali ABVS (angl. *Agent Based Vulnerability Score*). Vrednotenje metode ABVS bomo izvedli po naslednjih korakih:

- Izračun frekvenc izkoriščanja posameznih vektorjev CVSS v zbirki NVD z uporabo naključnih napadalcev.
- Primerjava učinkovitosti posameznih metod določanja prioritet ranljivostim na osnovi vrednotenja kazalnika $ExposureEV_k$ (glej definicijo 5.1).
- Vrednotenje napovednih sposobnosti posameznih metod z ugotavljanjem števila



Slika 6.2

Primerjava učinkovitosti metod v simulacijskem okolju. Poleg imena vsake metode je navedeno število preostalih izkoristljivih ranljivosti v informacijskem sistemu. Učinkovitejša metoda hitreje odstranjuje izkoristljive ranljivosti.

odstranjenih izkoristljivih ranljivosti po posameznih kvartalih simulacije.

ABVS metodo bomo primerjali z metodami, ki so navedene v tabeli 5.1. Pri tem bomo uporabili simulacijski model za primerjanje učinkovitosti metod, ki smo ga podrobneje predstavili v podpoglavju 5.6. Prav tako bomo uporabili povsem iste podatke, ki smo jih predstavili že v podpoglavju 5.7.

Pri vseh predlaganih metodah do sedaj smo poizkušali predvidevati izkoristljive ranljivosti brez podatkov o dejanskih izkoriščanjih v praksi. Tokrat bomo v eksperiment vključili tudi metodo, ki se pri določanju prioritet ranljivostim opira na podatke o izkoristljivih ranljivostih v zbirki EDB. Pri tej metodi dajemo prednost tistim ranljivostim iz zbirke NVD, katerih vektorji CVSS se v večjem deležu pojavljajo v zbirki EDB. Takšna metrika z imenom *Exploitation ratio* je predstavljena v [14]. Delež določimo po naslednji formuli, kjer funkcija $count_x(vect)$ vrne število vseh vektorjev CVSS *vect* v

zbirki ranljivosti x.

$$ShareEV(vect) = \frac{count_{EDB}(vect)}{count_{NVD}(vect)} \tag{6.1}$$

Opisana metoda daje prednost ranljivostim, za katere obstaja večja verjetnost, da so izkoristljive glede na znane podatke v zbirki EDB. Pri tej metodi torej ne govorimo o predvidevanju z upoštevanjem lastnosti napadalcev. Kljub vsemu jo lahko koristno uporabimo v kontrolne namene. Nobena metoda predvidevanja izkoristljivih ranljivosti na ravni vektorjev CVSS ne more biti pri predvidevanju boljša od nje. Primerjava z njo nam lahko predvsem razkrije, kako dobro se posamezna metoda približa idealni učinkovitosti pri predvidevanju izkoristljivih ranljivosti. Označili jo bomo z oznako CTRL.

6.5 Merjenje učinkovitosti metod na osnovi zmanjševanja tveganja

Učinkovitost metod smo do sedaj merili le s funkcijo *ExposureEV* (5.1). V primeru metode CTRL, s katero lahko najhitreje odstranimo vse izkoristljive ranljivosti, se postavlja vprašanje, če res najhitreje zmanjšuje tudi tveganje. Ocene CVSS res niso najprimernejše za predvidevanje izkoristljivih ranljivosti, a so po drugi strani najbolj verodostojne ocene, če do izkoriščanja v resnici pride [82]. Zato je smiselno, da v merilni sestav za ocenjevanje učinkovitosti metod vključimo tudi oceno CVSS. Manjša kot je vsota ocen CVSS vseh preostalih izkoristljivih ranljivosti v IS, manjše je preostalo tveganje. Skladno s to opredelitvijo lahko merimo učinkovitost zmanjševanja tveganja z naslednjo funkcijo 4.1:

Definicija 6.1 (Zmanjševanje tveganja):

$$RiskMitigation_k = \sum_{i=0}^{k} R_i$$

R_i je vsota ocen CVSS preostalih izkoristljivih ranljivosti v IS po i odstranjenih ranljivostih iz IS, k je število korakov izvajanja metode.

Merjenje zmanjševanja tveganja z uporabo funkcije $RiskMitigation_k$ bo dodatni korak v našem eksperimentu pri vrednotenju metod.

Sledila bo analiza preživetja izkoristljivih ranljivosti po metodi Kaplan-Meier.[1] Slednja se uporablja pri ugotavljanju časa do nastopa določenega dogodka. Zanima nas, koliko časa je v povprečju potrebnega pri posamezni metodi, da izkoristljivo ranljivost odstranimo iz sistema.

6.6 Evalvacija

6.6.1 Analiza frekvenc izkoriščanja vektorjev CVSS

V prvem koraku vrednotenja modela ABVS smo izračunali frekvence izkoriščanja vektorjev CVSS, ki se pojavljajo v podatkovni zbirki NVD. Ta korak je ključen za določitev prioritet vektorjem. Med 500.000 poizkusi naključno ustvarjenih napadalcev je bilo uspešno izkoriščenih 81.273 naključno izbranih ranljivosti v zbirki NVD. Izkoriščene ranljivosti so opisane z 236 različnimi vektorji.

Seznam najpogosteje izkoriščanih vektorjev se nahaja v tabeli 6.1. Prikazani so vektorji (skupaj le 20 vektorjev), ki predstavljajo več kot 90 % vseh izkoriščanj (73.418). V stolpcu z naslovom *Frekvenca* je navedeno število uspešnih izkoriščanj posameznega vektorja. Stolpec je urejen padajoče, kar pomeni, da je v prvi vrstici najpogosteje izkoriščan vektor. Stolpec *NVD* predstavlja skupno število ranljivosti s tem vektorjem v zbirki NVD, stolpec *EDB* je število izkoristljivih ranljivosti s tem vektorjem v zbirki EDB in stolpec *% PoC* delež izkoristljivih ranljivosti tega vektorja, ki ga izračunamo po formuli 6.1 (*ShareEV*). Dodali smo še stolpec *EKITS*, ki predstavlja ranljivosti, ki so se že izkoriščale v praksi. V tabeli 6.2 so za iste vektorje CVSS podane kvalitativne in kvantitativne ocene CVSS. Na sliki 6.3 je prikazana porazdelitev frekvenc vseh izkoriščanih vektorjev.

6.6.2 Analiza učinkovitosti

Primerjava učinkovitosti metod, kjer smo merili vrednost kazalnika $ExposureEV_k$, je podana v tabeli 6.3. Primerjane metode so navedene po vrsticah, stolpci pa predstavljajo vrednost kazalnika $ExposureEV_k$ na 25 %, 50 %, 75 % in 100 % opravljene simulacije. Poleg izmerjenih vrednosti je naveden standardni odklon.

Primerjava učinkovitosti metod z vidika zmanjševanja tveganja, kjer smo merili vrednost kazalnika $RiskMitigation_k$, je podana v tabeli 6.4. Primerjane metode so navedene po vrsticah, stolpci pa predstavljajo vrednost kazalnika $RiskMitigation_k$ na 25

[1] http://www.real-statistics.com/survival-analysis/

Tabela 6.1

Seznam vektorjev CVSS, ki so bili v simulaciji z naključno ustvarjenimi napadalci najpogosteje izkoriščani. Število uspešnih izkoriščanj je navedeno v stolpcu *Frekvenca*. V stolpcih *NVD*, *EDB* in *EKITS* so podatki o številu ranljivosti s tem vektorjem v istoimenskih podatkovnih zbirkah. Stolpec *% PoC* predstavlja delež izkoristljivih ranljivosti danega vektorja v zbirki NVD. V prvi vrstici je vektor CVSS, ki je bil v simulaciji izkoriščen največkrat, in sicer 18.172-krat. Takšen vektor ima 4.052 ranljivosti v zbirki NVD, 793 v zbirki EDB in 4 v zbirki EKITS. Kar 20 % ranljivosti s tem vektorjem v zbirki NVD je izkoristljivih.

CVSS vektor	*Frekvenca*	*NVD*	*EDB*	*% PoC*	*EKITS*
AV:N/AC:L/Au:N/C:P/I:P/A:P	18.172	4.052	793	20 %	4
AV:N/AC:L/Au:N/C:C/I:C/A:C	9.572	3.229	137	4 %	41
AV:N/AC:L/Au:N/C:P/I:N/A:N	8.751	2.353	162	7 %	1
AV:N/AC:M/Au:N/C:P/I:P/A:P	5.976	3.213	264	8 %	0
AV:N/AC:M/Au:N/C:N/I:P/A:N	4.952	4.764	276	6 %	1
AV:N/AC:M/Au:N/C:C/I:C/A:C	4.581	3.672	200	5 %	43
AV:N/AC:L/Au:N/C:N/I:N/A:P	4.061	1.967	63	3 %	0
AV:A/AC:M/Au:N/C:P/I:P/A:P	2.782	1.423	0	0 %	0
AV:N/AC:L/Au:N/C:N/I:P/A:N	1.890	733	15	2 %	1
AV:N/AC:L/Au:S/C:P/I:P/A:P	1.730	756	79	10 %	0
AV:N/AC:M/Au:N/C:P/I:N/A:N	1.657	1.050	8	1 %	1
AV:L/AC:L/Au:N/C:C/I:C/A:C	1.580	1.103	74	7 %	1
AV:N/AC:L/Au:S/C:P/I:N/A:N	1.403	734	22	3 %	0
AV:N/AC:L/Au:N/C:P/I:P/A:N	1.251	290	7	2 %	0
AV:L/AC:L/Au:N/C:P/I:N/A:N	1.202	604	3	0 %	0
AV:N/AC:M/Au:N/C:P/I:P/A:N	948	558	5	1 %	0
AV:N/AC:M/Au:N/C:N/I:N/A:P	942	1.056	30	3 %	0
AV:N/AC:L/Au:N/C:N/I:N/A:C	828	825	19	2 %	0
AV:L/AC:L/Au:N/C:P/I:P/A:P	640	283	5	2 %	0
AV:N/AC:M/Au:S/C:N/I:P/A:N	500	935	30	3 %	0
Vsota	*73.418*	*33.600*	*2.192*		*93*

%, 50 %, 75 % in 100 % opravljene simulacije. Poleg izmerjenih vrednosti je naveden standardni odklon.

Slika 6.4 prikazuje tipičen potek simulacije odstranjevanja izkoristljivih ranljivosti iz IS ter primerjavo učinkovitosti metod ABVS, HTAC in CTRL z uporabo funkcije $ExposureEV_k$. S slike 6.5 lahko prepoznamo, za koliko je metoda ABVS boljša pri

Tabela 6.2

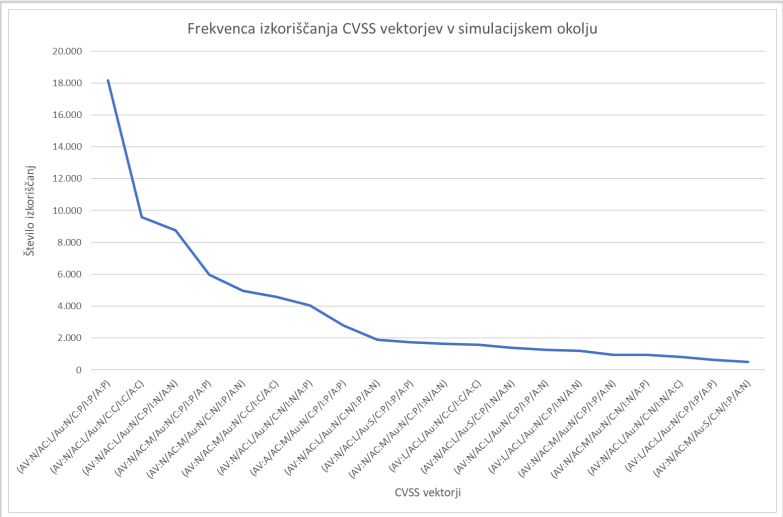
Seznam v simulaciji najpogosteje izkoriščanih vektorjev CVSS s pripadajočimi frekvencami ter kvalitativnimi in kvantitativnimi ocenami CVSS.

CVSS vektor	*Frekvenca*	*CVSS kval.*	*CVSS*
AV:N/AC:L/Au:N/C:P/I:P/A:P	18.172	HIGH	7,5
AV:N/AC:L/Au:N/C:C/I:C/A:C	9.572	HIGH	10
AV:N/AC:L/Au:N/C:P/I:N/A:N	8.751	MEDIUM	5
AV:N/AC:M/Au:N/C:P/I:P/A:P	5.976	MEDIUM	6,8
AV:N/AC:M/Au:N/C:N/I:P/A:N	4.952	MEDIUM	4,3
AV:N/AC:M/Au:N/C:C/I:C/A:C	4.581	HIGH	9,3
AV:N/AC:L/Au:N/C:N/I:N/A:P	4.061	MEDIUM	5
AV:A/AC:M/Au:N/C:P/I:P/A:P	2.782	MEDIUM	5,4
AV:N/AC:L/Au:N/C:N/I:P/A:N	1.890	MEDIUM	5
AV:N/AC:L/Au:S/C:P/I:P/A:P	1.730	MEDIUM	6,5
AV:N/AC:M/Au:N/C:P/I:N/A:N	1.657	MEDIUM	4,3
AV:L/AC:L/Au:N/C:C/I:C/A:C	1.580	HIGH	7,2
AV:N/AC:L/Au:S/C:P/I:N/A:N	1.403	MEDIUM	4
AV:N/AC:L/Au:N/C:P/I:P/A:N	1.251	MEDIUM	6,4
AV:L/AC:L/Au:N/C:P/I:N/A:N	1.202	LOW	2,1
AV:N/AC:M/Au:N/C:P/I:P/A:N	948	MEDIUM	5,8
AV:N/AC:M/Au:N/C:N/I:N/A:P	942	MEDIUM	4,3
AV:N/AC:L/Au:N/C:N/I:N/A:C	828	HIGH	7,8
AV:L/AC:L/Au:N/C:P/I:P/A:P	640	MEDIUM	4,6
AV:N/AC:M/Au:S/C:N/I:P/A:N	500	LOW	3,5
Vsota	*73.418*		

omejevanju izpostavljenosti IS grožnjam od metode CVSS (funkcija $ExposureEV_k$). Razliko učinkovitosti metod predstavlja površina med krivuljama.

6.6.3 *Analiza sposobnosti predvidevanja*

Ob zaključku vsakega kvartala simulacije smo pri vsaki metodi merili povprečno število odstranjenih izkoristljivih ranljivosti. Hkrati s tem smo po posameznih kvartalih simulacije preverjali tudi število odstranjenih ranljivosti glede na kvalitativne ocene



Slika 6.3

Frekvence izkoriščanja vektorjev CVSS v simulacijskem okolju. Na sliki je prikazanih 20 najpogosteje izkoriščanih vektorjev. Vrednosti so urejene od najvišje do najnižje. Z grafa lahko razberemo, da frekvenca pada eksponentno. Mnogi vektorji CVSS niso nikoli izkoriščeni.

Tabela 6.3

Primerjava učinkovitosti metod z uporabo funkcije $ExposureEV_k$. Podatki predstavljajo izpostavljenost informacijskega sistema PoC ranljivostim. Meritev je bila opravljena na koncu vsake četrtine simulacije. Poleg podatkov o povprečni vrednosti ($\bar{x}$) je podan tudi standardni odklon (σ).

Metoda	*25 %*		*50 %*		*75 %*		*100 %*	
	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
CTRL	9.534	1.416	14.117	2.301	15.666	2.670	15.788	2.707
ABVS	10.022	1.466	15.399	2.453	18.028	3.044	18.760	3.238
HTAC	10.091	1.473	16.235	2.564	19.664	3.297	20.473	3.513
CVSS	12.810	1.684	19.141	2.697	22.269	3.310	22.992	3.501
VRSS	12.874	1.684	20.712	2.840	24.065	3.463	24.840	3.657

CVSS. Rezultati so predstavljeni v tabeli 3.1. Zaradi preglednosti smo se omejili le na tiste metode, ki so nam v pomoč pri oceni učinkovitosti metode ABVS.

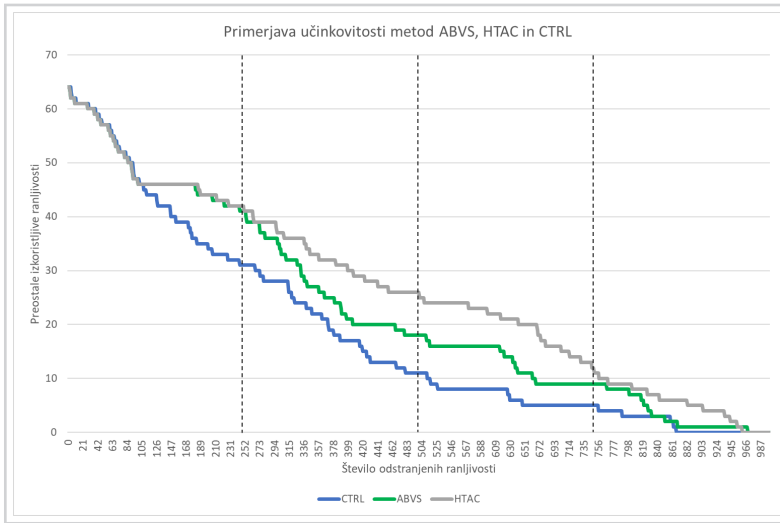
6.6.4 *Krivulja preživetja izkoristljivih ranljivosti*

Povprečni čas preživetja izkoristljivih ranljivosti v IS pri uporabi posameznih metod je predstavljen v tabeli 6.5. Čas preživetja smo izračunali po definiciji mediane preživetja

Tabela 6.4

Primerjava učinkovitosti metod z uporabo funkcije $RiskMitigation_k$. Podatki predstavljajo obseg preostalega tveganja. Meritev je bila opravljena na koncu vsake četrtine simulacije. Poleg podatkov o povprečni vrednosti ($\bar{x}$) je podan tudi standardni odklon (σ).

Metoda	*25 %*		*50 %*		*75 %*		*100 %*	
	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
CTRL	62.340	9.639	93.812	16.377	102.568	18.310	103.062	18.429
ABVS	64.633	11.094	98.841	16.020	114.595	18.012	118.944	18.538
HTAC	65.129	9.678	101.948	16.250	119.657	19.710	124.013	20.895
CVSS	83.054	9.724	116.619	16.307	130.018	19.752	132.683	20.919
VRSS	83.617	11.101	128.496	17.458	143.118	19.476	146.021	20.005

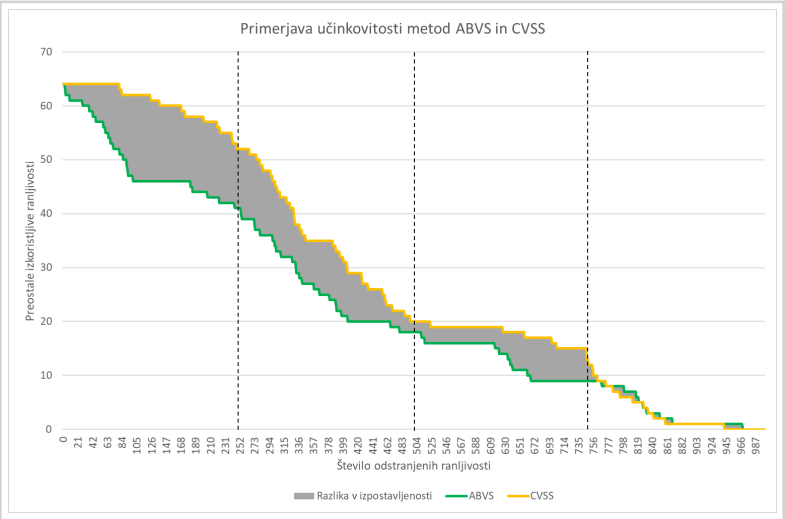


Slika 6.4

Graf prikazuje število preostalih izkoristljivih ranljivosti v IS pri posamezni metodi odstranjevanja. Na sliki je primerjava metod ABVS, HTAC in CTRL. Učinkovitejša metoda hitreje odstranjuje izkoristljive ranljivosti, saj je s tem zagotovljena nižja izpostavljenost IS grožnjam (nižja vrednost kazalnika $ExposureEV_k$).

(angl. Median Survival Time).[2] Časovno enoto pri vrednotenju mediane predstavlja zaporedni korak odstranjevanja ranljivosti. Po definiciji je mediana preživetja čas, ko krivulja preživetja doseže vrednost $S(t) = 0,5$. Krivulji preživetja metod ABVS in CVSS sta prikazani na sliki 6.6.

[2] http://www.real-statistics.com/survival-analysis/kaplan-meier-procedure/kaplan-meier-overview/



Slika 6.5

Primerjava učinkovitosti metod ABVS in CVSS. Površina med krivuljama pokaže, za koliko je metoda ABVS uspešnejša pri odstranjevanju izkoristljivih ranljivosti oz. pri omejevanju izpostavljenosti IS grožnjam.

Tabela 6.5

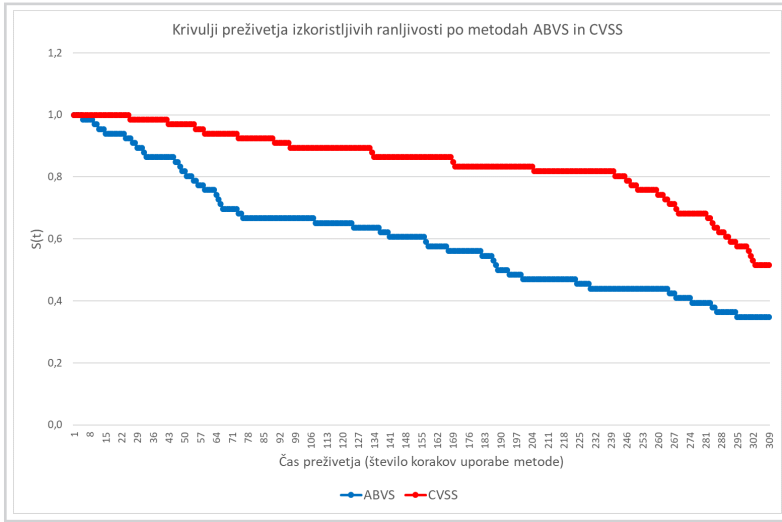
Mediana preživetja (angl. Median Survival Time) izkoristljivih ranljivosti pri uporabi posameznih metod odstranjevanja. Podatki predstavljajo povprečno mediano preživetja ($\tilde{x}$) in njen standardni odklon (σ) po 1.000 izvedenih simulacijah.

Metoda	$\tilde{x}$	σ
CTRL	211	48
ABVS	263	50
HTAC	293	68
CVSS	324	37
VRSS	383	25

6.7 Razprava

6.7.1 Frekvence izkoriščanja vektorjev CVSS

Analiza s simulacijo pridobljenih frekvenc izkoriščanja vektorjev CVSS kaže, da pogostost izkoriščanja upada eksponentno (glej tabelo 6.1). Manj kot 10 % vektorjev predstavlja 90 % vseh izkoriščanj. To se lepo vidi tudi na sliki 6.3, ki prikazuje porazdelitev frekvenc. Po naši predpostavki so to vektorji, ki predstavljajo največje tveganje



Slika 6.6

Slika prikazuje krivulji preživetja metod ABVS in CVSS na eni izmed simulacij. Čas predstavlja zaporedni korak odstranjevanja ranljivosti.

za IS, saj je zanje največ možnosti izkoriščanja. Stolpec *CVSS* v tabeli 6.2 predstavlja ocene CVSS teh vektorjev. Le te kažejo, da imajo vektorji CVSS z visokimi frekvencami sicer visoke ocene CVSS, a na samem vrhu niso najvišje. Vrednosti v stolpcu *EDB* tabele 6.1 po drugi strani dokazujejo, da so vektorji z visokimi frekvencami tudi pogosti v zbirki EDB, ali z drugimi besedami, da so pogosto izkoristljivi. Med njimi je le vektor `[AV:A/AC:M/Au:N/C:P/I:P/A:P]`, ki se v tabeli 6.1 nahaja na osmem mestu, brez ene same izkoristljive ranljivosti v zbirki EDB. Omenjeno ranljivost smo posebej obravnavali že v podpoglavju 5.9 in podali razloge za to odstopanje.

V stolpcu z oznako % *PoC* so povsem na vrhu prioritetnega seznama vektorji z visokim deležem izkoristljivih ranljivosti. Tudi v stolpcu EKITS, ki predstavlja dejansko izkoriščane ranljivosti, so te povsem na vrhu seznama in jim pripadajo visoke frekvence izkoriščanja. Na osnovi teh ugotovitev lahko zaključimo, da visoka frekvenca izkoriščanja v našem simulatorju predstavlja dober indikator izkoriščanja. Preprosto povedano, vektorji, ki jih lahko izkoristi več napadalcev in so pogosteje odkriti v programski opremi, so večkrat izkoriščani oz. bodo zanje pogosteje razviti postopki izkoriščanja.

6.7.2 Učinkovitost omejevanja izpostavljenosti IS grožnjam

Najprej podajamo oceno učinkovitosti metod na osnovi funkcije $ExposureEV_k$. Uporaba frekvenc izkoriščanja vektorjev CVSS pri dejanskem razvrščanju ranljivosti po prioriteti kaže na precejšnjo izboljšavo v primerjavi z metodo HTAC, ki smo jo predstavili v poglavju 5. Rezultati v tabeli 6.3 dokazujejo, da je metoda ABVS za 16 % boljša od metode CVSS, ki je v praksi danes najpogosteje uporabljena. Primerjava metod ABVS in HTAC kaže, da smo z novo metodo dosegli 6-% izboljšavo. Zanimiva je tudi primerjava metode ABVS in kontrolne metode CTRL, ki predstavlja optimalno metodo pri določanju prioritet na ravni vektorjev CVSS. ABVS je v celoti gledano za 14 % slabša od metode CTRL, kar pa je precej bolje od metode CVSS, ki je za kar 29 % slabša od metode CTRL. Pri vseh primerjavah smo uporabili Z-test za povprečja nad dvema vzorcema, pri čemer smo upoštevali, da je $\alpha = 0,01$ (glej Dodatek A).

6.7.3 Sposobnost predvidevanja izkoristljivih ranljivosti

Rezultati v tabeli 6.6 kažejo, da je metoda ABVS boljša pri predvidevanju izkoristljivih ranljivosti od metode HTAC, in sicer predvsem v drugem kvartalu. Metoda ABVS takrat odkrije več ranljivosti razreda MEDIUM. ABVS metoda je tudi na splošno hitrejša od metode HTAC pri odstranjevanju izkoristljivih ranljivosti, saj ji v prvih dveh kvartalih v povprečju uspe odstraniti več izkoristljivih ranljivosti. Opazimo lahko, da v zadnjih dveh kvartalih pri metodi ABVS v IS v povprečju ostaja manj ranljivosti. Zanimivo je, da kontrolna metoda CTRL, ki predstavlja optimalni pristop, občasno odstrani ranljivosti razreda LOW že v prvih dveh kvartalih. Tega nismo zaznali pri nobeni drugi metodi.

Sposobnost predvidevanja izkoristljivih ranljivosti se odraža tudi v času njihovega preživetja v IS. Z metodo Kaplan-Meier smo v eksperimentu za vsako metodo izračunali mediano preživetja izkoristljivih ranljivosti. Rezultati kažejo, da je povprečna mediana preživetja najkrajša pri metodi CTRL. Metoda ABVS je pri tej meritvi v povprečju za 17 % boljša od metode CVSS. Pri primerjavi smo uporabili Z-test za povprečja nad dvema vzorcema in upoštevali, da je $\alpha = 0,01$ (glej Dodatek A).

6.7.4 Učinkovitost zmanjševanja tveganja

Primerjava učinkovitosti metod na osnovi funkcije $RiskMitigation_k$ kaže na manjše razlike med metodami. Kljub vsemu je vrstni red metod glede na učinkovitost povsem

Tabela 6.6

Število odstranjenih PoC ranljivosti po posameznih metodah, merjeno ob zaključku vsake četrtine izvajanja simulacije. Navedene so povprečne vrednosti. Tabela je razdeljena na štiri odseke. V zgornjem delu je podano skupno število vseh odstranjenih ranljivosti. V spodnjem delu je navedeno število odstranjenih ranljivosti glede na kvalitativno oceno CVSS, ki ranljivosti deli v kategorije HIGH, MEDIUM in LOW (glej tabelo 3.1).

Metoda	*25 %*	*50 %*	*75 %*	*100 %*
	Vse skupaj			
CTRL	31,66	14,48	9,34	2,08
ABVS	27,81	15,25	7,84	6,66
HTAC	26,75	11,83	10,89	8,09
CVSS	17,16	24,06	8,64	7,69
	HIGH			
CTRL	19,64	5,96	5,11	0,35
ABVS	22,08	4,11	2,51	2,36
HTAC	22,17	4,99	1,76	2,13
CVSS	17,16	13,89	-	-
	MEDIUM			
CTRL	11,99	8,41	3,39	1,38
ABVS	5,73	11,14	5,26	3,05
HTAC	4,58	6,83	9,08	4,68
CVSS	-	10,16	8,64	6,36
	LOW			
CTRL	0,03	0,12	0,84	0,35
ABVS	-	-	0,08	1,26
HTAC	-	-	0,05	1,28
CVSS	-	-	-	1,33

enak kot pri merjenju na osnovi funkcije $ExposureEV_k$. Rezultati v tabeli 6.4 kažejo, da je metoda CTRL, ki uporablja podatke o preteklih izkoristljivih ranljivostih, za kar 20 % boljša od metode CVSS. Metoda ABVS, ki predvideva izkoristljive ranljivosti na osnovi lastnosti napadalcev, je za 8 % boljša od metode CVSS. Rezultati dokazujejo, da odstranjevanje ranljivosti zaradi visokih ocen CVSS ne vodi nujno k najhitrejšemu zmanjševanju tveganja.

Tabela 6.7

Seznam redko izkoriščenih vektorjev CVSS, ki se v visokem deležu pojavljajo med izkoristljivimi ranljivostmi. Število uspešnih izkoriščanj je navedeno v stolpcu *Frekvenca*, v stolpcu CVSS pa je podana ocena po istoimenskem standardu. V stolpcih *NVD* in *EDB* so podatki o številu ranljivosti s tem vektorjem v istoimenskih podatkovnih zbirkah. Stolpec *% PoC* predstavlja delež izkoristljivih ranljivosti danega vektorja v zbirki NVD. Stolpec *TAC* podaja vrednost TAC, ki ga uporablja metoda HTAC.

CVSS vektor	*Frekv.*	*CVSS*	*NVD*	*EDB*	*% PoC*	*TAC*
AV:N/AC:H/Au:N/C:P/I:N/A:P	3	4,0	2	1	50 %	9
AV:N/AC:L/Au:N/C:C/I:N/A:C	11	9,4	5	2	40 %	7
AV:N/AC:L/Au:S/C:N/I:C/A:C	4	8,5	3	1	33 %	4
AV:A/AC:L/Au:S/C:C/I:C/A:C	23	7,7	15	3	20 %	2
AV:L/AC:L/Au:N/C:C/I:N/A:C	10	6,6	5	1	20 %	2
AV:N/AC:L/Au:N/C:N/I:C/A:C	17	9,4	11	2	18 %	5
AV:L/AC:M/Au:N/C:P/I:N/A:C	12	5,4	6	1	17 %	3

6.7.5 Omejitve predstavljene metode

Rezultati so razkrili tudi manjšo pomanjkljivost metode ABVS. V zbirki NVD obstajajo posamezne izkoristljive ranljivosti z vektorji CVSS, ki se redko pojavijo v realnosti. Zaradi nizke frekvence izkoriščanja jim daje metoda ABVS zelo nizko prioriteto. Posledično to pomeni, da te izkoristljive ranljivosti odstrani šele proti koncu postopka odstranjevanja. Seznam vektorjev CVSS teh ranljivosti je podan v tabeli 6.7. Ranljivosti na tem seznamu imajo dokaj visoke ocene CVSS (večina jih je višja od 5), kar pomeni, da jih omenjena metoda hitreje odstrani. Metoda HTAC je pri tem precej slabša, saj razen pri prvih dveh vektorjih CVSS s tega seznama izračuna nizke vrednosti TAC.

6.8 Agent Based Vulnerability Score (ABVS)

Pri metodi HTAC, ki smo jo predstavili v poglavju 5, se nismo posebej ukvarjali z ocenami ranljivosti. Zanimalo nas je predvsem, kako koristno je lahko število TAC pri določanju prioritet ranljivostim. Takšen pristop je lahko za skrbnika IS nepregleden. Človek veliko lažje oceni resnost grožnje, ki jo predstavlja ranljivost, če ji določimo kvalitativno oceno (visoko, srednje ali nizko tveganje) ali če ji določimo kvantitativno oceno na nekem običajnem intervalu.

Ker obstoječa kvantitativna sistema ocenjevanja CVSS in VRSS ocenjujeta ranljivo-

sti z vrednostmi na intervalu od 0 do 10, je zaradi primerjav z njima smiselno, da tudi ocenjevanje po naši metodi predstavimo na tem intervalu. Frekvence vektorjev bomo preslikali na interval od 0 do 10 tako, da bo najvišji frekvenci ustrezala najvišja ocena 10. Preslikavo bomo naredili tako, da bodo vrednosti čim bolj enakomerno porazdeljene po celotnem intervalu. Na takšen način bo uporabnik lažje ocenil, kako resno grožnjo predstavlja ranljivost. Ker frekvence izkoriščanja vektorjev CVSS v tabeli 6.1 padajo eksponentno, bomo uporabili logaritemsko funkcijo. Oceno ranljivosti ABVS opredeljujemo z definicijo 6.2.

Definicija 6.2 (Agent Based Vulnerability Score (ABVS)):

$$ABVS(vect) = \frac{\ln f(vect)}{\ln f(vect_{max})} \times 10$$

Pri tem je $f(vect)$ frekvenca izkoriščanja danega vektorja CVSS in $f(vect_{max})$ najvišja frekvenca med vsemi vektorji CVSS. Če je $f(vect) = 0$, je ocena $ABVS = 0$.

Ocene ABVS za vektorje CVSS z najvišjimi frekvencami izkoriščanja podajamo v tabeli 6.8. Analiza ocen ABVS pokaže, da ima veliko vektorjev vrednost 0, saj v simulaciji ni bila nikoli izkoriščena ranljivost s takšnim vektorjem. Skupaj je kar 532 takšnih vektorjev od 729 možnih različnih kombinacij. Ker se ti vektorji tudi v zbirkah ranljivosti ne pojavljajo, jih lahko izločimo iz nadaljnje obravnave. Kljub temu imajo ostali vektorji 91 različnih možnih ocen. Pri sistemu CVSS je različnih možnih ocen le 76. Porazdelitev ocen ABVS, ki so različne od 0, je prikazana na sliki 6.7. Ugotovimo lahko, da z naraščanjem ocene ABVS pada število vektorjev CVSS, ki imajo tako visoko oceno.

Na sliki 6.8 je prikazana porazdelitev ocen ABVS vseh ranljivosti v podatkovni zbirki NVD. Tokrat lahko ugotovimo, da prevladujejo ranljivosti z visokimi ocenami ABVS. Pri nižjih ocenah ABVS pa število ranljivosti zelo upade. Pri ranljivostih zbirke EDB so razlike še očitnejše (slika 6.9). Ranljivosti z ocenami ABVS, manjšimi od 3, skorajda ni. Ugotovimo lahko, da imajo izkoristljive ranljivosti višje ocene ABVS. Na podlagi tega spoznanja lahko zaključimo, da ocene ABVS natančneje opisujejo tveganje, ki ga predstavljajo posamezne ranljivosti, kot ocene CVSS in drugi obstoječi sistemi ocenjevanja. Sistem ABVS je dinamičen in se lahko prilagaja procesu odkrivanja ranljivosti. Povsem mogoče namreč je, da se v prihodnje pojavijo ranljivosti z vektorji CVSS, ki

Tabela 6.8

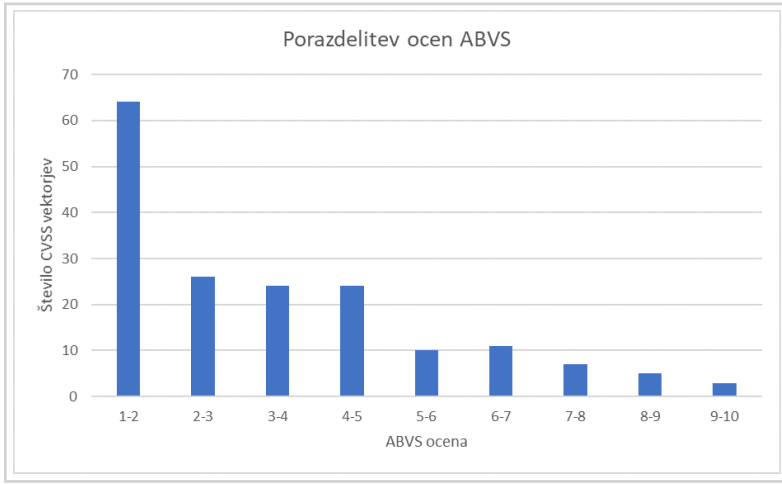
V tabeli so navedene ocene ABVS za vektorje CVSS, ki so bili v simulaciji z naključno ustvarjenimi napadalci najpogosteje izkoriščani. Frekvence so podane v stolpcu *Frekvenca*. Za primerjavo sta dodani tudi ocena CVSS in vrednost TAC, ki je osnova metode HTAC.

CVSS vektor	*Frekvenca*	*TAC*	*CVSS*	*ABVS*
AV:N/AC:L/Au:N/C:P/I:P/A:P	18.172	18	7,5	10,00
AV:N/AC:L/Au:N/C:C/I:C/A:C	9.572	7	10	9,35
AV:N/AC:L/Au:N/C:P/I:N/A:N	8.751	14	5	9,25
AV:N/AC:M/Au:N/C:P/I:P/A:P	5.976	13	6,8	8,87
AV:N/AC:M/Au:N/C:N/I:P/A:N	4.952	5	4,3	8,67
AV:N/AC:M/Au:N/C:C/I:C/A:C	4.581	4	9,3	8,60
AV:N/AC:L/Au:N/C:N/I:N/A:P	4.061	8	5	8,47
AV:A/AC:M/Au:N/C:P/I:P/A:P	2.782	13	5,4	8,09
AV:N/AC:L/Au:N/C:N/I:P/A:N	1.890	8	5	7,69
AV:N/AC:L/Au:S/C:P/I:P/A:P	1.730	8	6,5	7,60
AV:N/AC:M/Au:N/C:P/I:N/A:N	1.657	11	4,3	7,56
AV:L/AC:L/Au:N/C:C/I:C/A:C	1.580	2	7,2	7,51
AV:N/AC:L/Au:S/C:P/I:N/A:N	1.403	4	4	7,39
AV:N/AC:L/Au:N/C:P/I:P/A:N	1.251	17	6,4	7,27
AV:L/AC:L/Au:N/C:P/I:N/A:N	1.202	4	2,1	7,23
AV:N/AC:M/Au:N/C:P/I:P/A:N	948	13	5,8	6,99
AV:N/AC:M/Au:N/C:N/I:N/A:P	942	4	4,3	6,98
AV:N/AC:L/Au:N/C:N/I:N/A:C	828	5	7,8	6,85
AV:L/AC:L/Au:N/C:P/I:P/A:P	640	5	4,6	6,59
AV:N/AC:M/Au:S/C:N/I:P/A:N	500	3	3,5	6,34

danes niso tako pogosto zastopani v zbirki NVD.

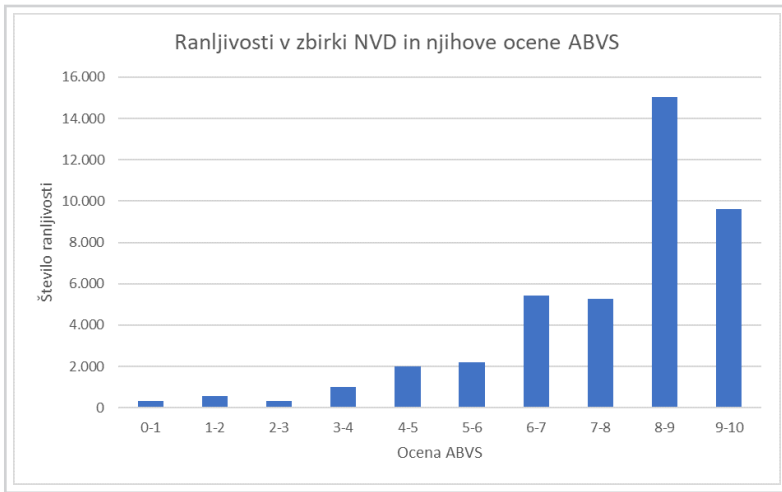
6.9 Zaključek

V tem poglavju smo predstavili izboljšano metodo določanja prioritet, ki upošteva lastnosti napadalcev. Namesto majhnega števila tipičnih napadalcev, ki so opredeljeni v knjižnici TAL, smo tokrat za določanje prioritet uporabili veliko množico naključno ustvarjenih napadalcev. Razvili smo simulacijski model za merjenje frekvenc izkoriščа-



Slika 6.7

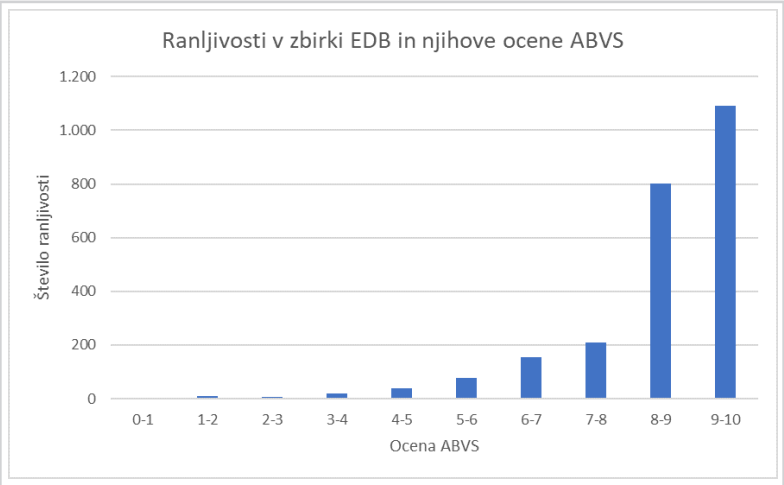
Porazdelitev ocen ABVS za vektorje CVSS, ki imajo frekvenco izkoriščanja večjo od 0.



Slika 6.8

Porazdelitev ocen ABVS ranljivosti v zbirki NVD.

nja ranljivosti zbirke NVD. V ta namen smo uporabili funkcijo izkoriščanja, ki smo jo opredelili v poglavju 4. Napadalce smo naključno ustvarili na osnovi atributov knjižnice TAL. Metoda daje prednost ranljivostim, ki imajo višjo frekvenco izkoriščanja.



Slika 6.9

Porazdelitev ocen ABVS ranljivosti v zbirki EDB. Za ranljivosti v tej zbirki obstajajo koncepti izkoriščanja (angl. Proof of Concept Exploits).

Poimenovali smo jo ABVS.

Učinkovitost metode smo izmerili s kazalnikom $ExposureEV_k$, ki smo ga opredelili v poglavju 5, ter jo primerjali z drugimi metodami določanja prioritet. V ta namen smo izvedli podoben eksperiment kot v prejšnjem poglavju pri vrednotenju metode HTAC. Ugotovili smo, da je metoda ABVS občutno učinkovitejša od prej predlagane HTAC.

Za potrebe vrednotenja metod smo dodatno opredelili funkcijo $RiskMitigation_k$, s katero izmerimo hitrost zmanjševanja tveganja v IS. Prepričali smo se, da odstranjevanje ranljivosti zaradi visokih ocen CVSS ne zagotavlja najhitrejšega zmanjševanja tveganja. Metoda CVSS je namreč slabša od metod, ki smo jih predlagali sami.

V zaključku smo na osnovi frekvenc izkoriščanja predstavili sistem ocenjevanja ranljivosti ABVS, ki ranljivostim dodeljuje ocene na intervalu od 0 do 10. Ocene ABVS so pri ocenjevanju tveganja točnejše od ocen CVSS.

Evalvacija metod v realnem okolju

7

7.1 Uvod

Predstavili smo več metod določanja prioritet odpravljanja ranljivosti in načinov ocenjevanja njihove učinkovitosti. Vrednotenje smo nato izvedli s pomočjo simulacijskega okolja. V realnem okolju bi bilo težko izvesti takšen eksperiment. Po drugi strani ima simulacijsko okolje določene omejitve in povsem možno je, da meritve učinkovitosti na realnih primerih pripeljejo do drugačnih rezultatov ali razkrijejo druge zanimive podrobnosti.

V simulacijskem okolju smo IS vsakokrat ustvarili iz naključno izbranih ranljivosti podatkovne zbirke NVD. Ob tem smo upoštevali ugotovitve in priporočila raziskovalcev, ki so to zbirko podatkov uporabljali v raziskovalne namene [12, 72]. Da bi se čim bolj približali realnim razmeram, v tem poglavju oblikujemo dva tipična primera današnjih informacijskih sistemov. V prvem primeru gre za IS, ki ga sestavlja programska oprema na strežnikih spletnega gostovanja. V drugem primeru je to nabor programskih orodij, ki se uporabljajo v tipičnem slovenskem podjetju. Predlagane metode določanja prioritet ranljivostim ovrednotimo na obeh IS in predstavimo rezultate. V zaključku poglavja rezultate primerjamo s sorodnimi raziskavami na tem področju ter ocenimo praktično uporabnost predlaganih metod v praksi.

7.2 Primera informacijskih sistemov v realnem okolju

Spletno gostovanje je storitev, ki jo danes uporablja vse več organizacij in posameznikov. Pri postavitvi svojih spletnih strani lahko izbirajo med številnimi odprtokodnimi in licenčnimi programskimi produkti. Spletna stran *Bitnami*[1] podaja podroben pregled komponent, ki jih običajno srečamo na strežnikih za gostovanje. Na osnovi teh informacij sestavljajo običajen IS strežnika za gostovanje naslednje komponente:

- operacijski sistem strežnika (Linux, Windows idr.),
- spletni strežnik (IIS, Apache, Nginx idr.),
- sistemi za upravljanje podatkov (MySQL, MSSQL, Postgres idr.),
- sistemi za obvladovanje vsebin (CMS, Blog idr.),
- programski dodatki in vtičniki za spletne strani.

[1] https://bitnami.com/

Skladno s tem seznamom smo iz zbirke ranljivosti NVD izbrali 342 produktov in njihovih dodatkov (od skupaj 16.069 evidentiranih ranljivih produktov v zbirki NVD). Za omenjene produkte je bilo v obdobju od 2010 do 2016 odkritih in javno objavljenih 2.862 ranljivosti. Opisane so s 108 različnimi vektorji CVSS. 90 % vseh ranljivosti je opisanih s 30 različnimi vektorji CVSS. Na osnovi zbirke EDB smo ugotovili, da je med njimi 253 izkoristljivih ranljivosti.

Za opis tipičnega IS v podjetju smo izbrali večje slovensko podjetje. Na osnovi intervjuja smo pridobili informacije o standardnih produktih, ki jih podjetje dovoljuje v okviru svojega IS, ter oblikovali naslednji seznam:

- operacijski sistem Windows na strežnikih in delovnih postajah,
- programska oprema Microsoft Office,
- programsko orodje Acrobat Reader,
- spletni brskalnik Microsoft IE.

V podjetju sicer uporabljajo preko 100 različnih podsistemov za podporo poslovnih in delovnih procesov, a zaradi varnostnih razlogov podrobnosti niso želeli razkriti. Na osnovi pridobljenih informacij smo tako prepoznali 11 različnih produktov. V zbirki NVD je bilo zanje v obdobju od leta 2010 do 2016 odkritih in javno objavljenih 2.909 ranljivosti. Opisane so s 67 različnimi vektorji CVSS. 90 % vseh ranljivosti je opisanih s samo 11 različnimi vektorji CVSS. Na osnovi podatkov v zbirki EDB smo ugotovili, da je med temi ranljivostmi 78 izkoristljivih. Poudariti moramo, da smo v podjetju uspeli zajeti le del uporabljene programske opreme, zato slika ni popolna.

Na obeh opisanih IS bomo preverili učinkovitost predlaganih metod ABVS, HTAC, VRSS in CTRL ter jih primerjali z metodo CVSS. Učinkovitost bomo ocenili tako z vidika predvidevanja izkoristljivih ranljivosti (spremenljivka $ExposureEV_k$) kot hitrosti zmanjševanja tveganja (spremenljivka $RiskMitigation_k$). Z eksperimentom ne moremo v popolnosti zajeti realnih razmer. Naš pristop na primer ne upošteva, da nekdo lahko uporablja varnostne mehanizme in s tem zmanjša možnost izkoriščanja.

Tudi pri drugih podobnih raziskavah so bili večinoma uporabljeni podatki zbirk NVD in EDB [12]. Dodatno možnost predstavlja seznam dejanskih izkoriščanj, ki nastaja pod okriljem podjetja Symantec[2] [14]. Podatki žal obstajajo le za komercialne produkte odjemalcev tega podjetja, kar daje le omejeno možnost vrednotenja [12].

[2]https://www.symantec.com/security_response/attacksignatures/

Tabela 7.1

Rezultati meritev učinkovitosti metod na IS za spletno gostovanje. Učinkovitost smo izmerili s kazalnikoma $ExposureEV_k$ in $RiskMitigation_k$. Poleg vsakega je navedeno razmerje učinkovitosti v primerjavi z učinkovitostjo metode CVSS.

Metoda	*ExposureEV*	*% CVSS*	*RiskMitigation*	*% CVSS*
CTRL	174.876	75 %	1.014.127	80 %
ABVS	205.168	88 %	1.188.670	94 %
HTAC	226.679	97 %	1.280.345	101 %
CVSS	232.827	100 %	1.268.065	100 %
VRSS	252.446	108 %	1.416.966	112 %

Tabela 7.2

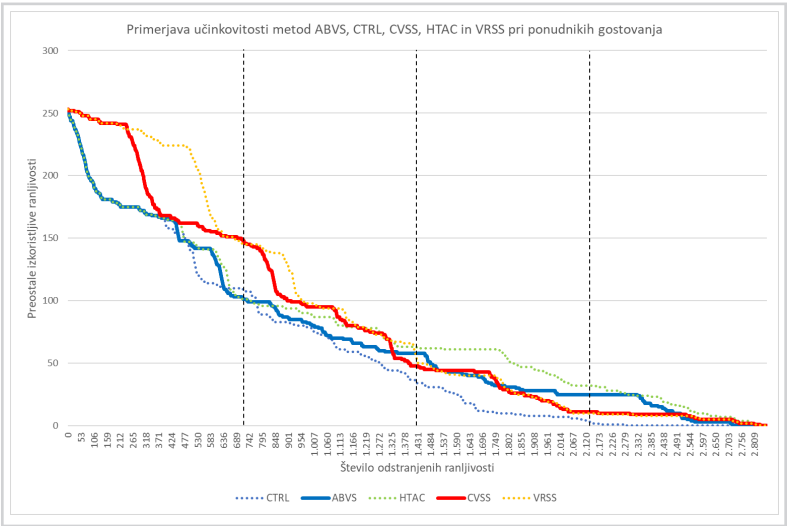
Rezultati meritev učinkovitosti metod za določanje prioritet na IS podjetja. Učinkovitost smo izmerili s kazalnikoma $ExposureEV_k$ in $RiskMitigation_k$. Poleg vsakega je navedeno razmerje učinkovitosti v primerjavi z učinkovitostjo metode CVSS.

Metoda	*ExposureEV*	*% CVSS*	*RiskMitigation*	*% CVSS*
CTRL	82.268	62 %	626.300	66 %
VRSS	130.343	98 %	932.655	98 %
CVSS	132.913	100 %	950.147	100 %
ABVS	147.740	111 %	1.075.810	113 %
HTAC	155.025	117 %	1.121.867	118 %

7.3 Evalvacija

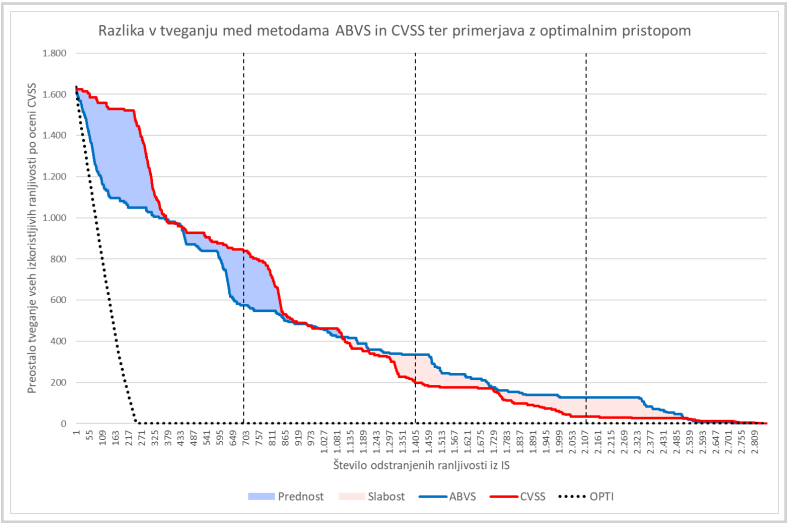
Rezultati vrednotenja metod na IS spletnega gostovanja so predstavljeni v tabeli 7.1. Slika 7.1 prikazuje gibanje števila preostalih izkoristljivih ranljivosti v IS gostitelja pri uporabi posameznih metod določanja prioritet. Na sliki 7.2 je prikazana razlika v tveganju med metodama ABVS in CVSS, izmerjena s funkcijo *RiskMitigation*. Prikazana je tudi teoretično idealna učinkovitost, ki pa je samo z uporabo atributov vektorja CVSS ne moremo doseči.

Rezultati vrednotenja metod na IS podjetja so predstavljeni v tabeli 7.2. Slika 7.3 prikazuje gibanje števila preostalih izkoristljivih ranljivosti v IS podjetja pri uporabi posameznih metod določanja prioritet. Na sliki 7.4 je prikazana razlika v tveganju med metodo CTRL, ki se je izkazala za najučinkovitejšo, in metodo CVSS. Razlika v tveganju je izmerjena s funkcijo *RiskMitigation*.



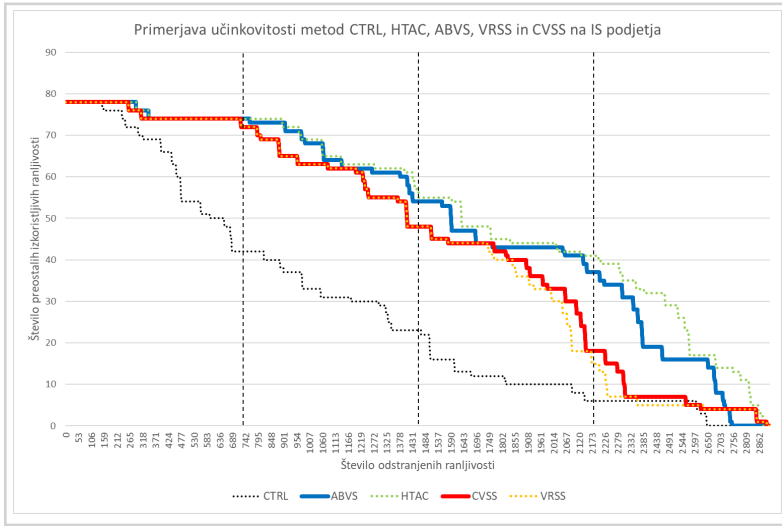
Slika 7.1

Graf prikazuje število preostalih izkoristljivih ranljivosti v IS ponudnika spletnega gostovanja. Programska oprema obsega operacijski sistem, podatkovne zbirke, spletne strežnike in spletne aplikacije različnih proizvajalcev s programskimi dodatki. Med slednjimi so najpogostejše aplikacije za upravljanje vsebin (CMS, blog). Na sliki je primerjava metod CTRL, HTAC, ABVS, VRSS in CVSS. Učinkovitejša metoda hitreje odstranjuje izkoristljive ranljivosti, saj je s tem zagotovljena nižja izpostavljenost IS grožnjam (nižja vrednost kazalnika $ExposureEV_k$).



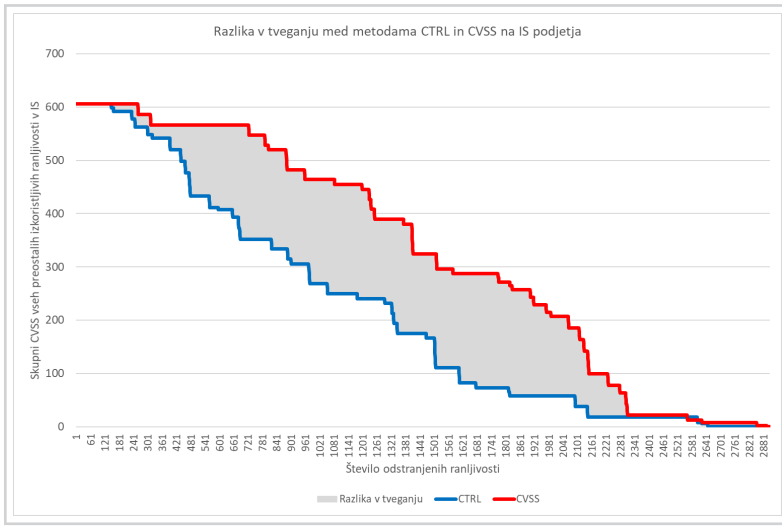
Slika 7.2

Graf prikazuje razliko v tveganju pri uporabi metod ABVS in CVSS na IS za spletno gostovanje. Tveganje je merjeno z ocenami CVSS preostalih izkoristljivih ranljivosti v IS. V prvem delu metoda ABVS hitreje zmanjšuje tveganje. V drugem delu je boljša metoda CVSS. Po kazalniku $RiskMitigation_k$ je v celoti gledano učinkovitejša tista metoda, pri kateri je preostalo tveganje manjše (razlika označenih površin na grafikonu). Krivulja OPTI predstavlja idealen pristop pri odstranjevanju ranljivosti.



Slika 7.3

Graf prikazuje število preostalih izkoristljivih ranljivosti v IS podjetja. V raziskavi smo izbrali večje slovensko podjetje. Programska oprema obsega Windows operacijski sistem, Office orodja za ustvarjanje dokumentov, IE brskalnik in Acrobat Reader orodje za pregledovanje dokumentov v PDF formatu. Na sliki je primerjava metod CTRL, HTAC, ABVS, VRSS in CVSS. Učinkovitejša metoda hitreje odstranjuje izkoristljive ranljivosti, saj je s tem zagotovljena nižja izpostavljenost IS grožnjam (nižja vrednost kazalnika $ExposureEV_k$).



Slika 7.4

Graf prikazuje razliko v tveganju pri uporabi metod CTRL in CVSS na IS podjetja. Učinkovitejša metoda hitreje zmanjšuje tveganje (nižja vrednost kazalnika $RiskMitigation_k$).

7.4 Razprava

7.4.1 Učinkovitost metod

Rezultati vrednotenja metod na IS gostitelja spletnih aplikacij so precej podobni tistim, ki smo jih dobili v simulacijskem okolju. Pri merjenju s kazalnikom $ExposureEV_k$ je metoda ABVS za 12 % učinkovitejša od metode CVSS (v simulatorju za 16 %). Podobno velja za metodo CTRL, ki je za kar 25 % boljša od metode CVSS (prej 29 %). Po drugi strani je metoda HTAC le za 3 % boljša od metode CVSS, kar je sicer precej manj kot v simulatorju (9 %), a še vedno bolje od metode CVSS. Pri merjenju s kazalnikom *RiskMitigation* sta le metodi CTRL (20 %) in ABVS (6 %) učinkovitejši od metode CVSS. Razlika med metodama ABVS in CVSS je manjša kot pri merjenju s kazalnikom *ExposureEV*. Dodatna prednost metode ABVS pred CVSS je v tem, da hitreje zmanjšuje tveganje v začetnem delu, ko je tveganje največje.

V primeru IS podjetja se rezultati precej razlikujejo od tistih v simulaciji. Samo metodi CTRL in VRSS sta učinkovitejši od metode CVSS. Pri merjenju s kazalnikom *ExposureEV* je metoda CTRL boljša za 38 %, pri kazalniku *RiskMitigation* pa za 34 %. Razlika v učinkovitosti metod VRSS in CVSS je zelo majhna (le 2 %). Ostale metode, ki smo jih predlagali (ABVS, HTAC), so dosegle slabše rezultate od metode CVSS.

7.4.2 Uporabnost metod v različnih informacijskih sistemih

Med opazovanima IS je več očitnih razlik. V primeru IS podjetja je malo produktov in večina izmed njih pripada istemu proizvajalcu (Microsoft). Njegova programska oprema je globalno zelo razširjena, kar glede na ugotovitve drugih raziskovalcev pomeni, da je zanimiva širokemu krogu oseb, ki se ukvarjajo z odkrivanjem ranljivosti [54]. To se kaže tudi v velikem številu odkritih ranljivosti. Po drugi strani je med njimi relativno malo izkoristljivih. Slednje lahko pripišemo velikemu interesu belih hekerjev za odkrivanje ranljivosti, saj jih večji proizvajalci nagrajujejo za to delo (BugBounty programi) [5, 6, 12]. Na osnovi pridobljenih informacij lahko proizvajalec hitro zagotovi zaščito ali ponudi popravek. Napadalci se zato ne trudijo z razvijanjem postopkov izkoriščanja za ranljivosti, ki jih bo težko izkoristiti. Posledično je na teh produktih manj izkoristljivih ranljivosti [14]. Funkcija izkoriščanja *isExploited*, ki predstavlja osnovo metod ABVS in HTAC, tega dejavnika ne upošteva, zato beležimo slabše rezultate. Gre namreč za dodatno komponento (de)motivacije, ki je z našim modelom ne moremo preveriti. V okviru vektorja CVSS nimamo podatkov, ki bi omogočali sklepanje

o takšnem ravnanju napadalcev.

Podroben pregled izkoristljivih ranljivosti IS podjetja pokaže tudi, da je povprečna vrednost TAC enaka 2,9 (najvišja je 7). V produktih so torej prisotne le izkoristljive ranljivosti, ki jih lahko izkoristi malo napadalcev. Pri ostalih ranljivostih je ta vrednost precej višja, in sicer 5,1 (najvišja 18). Možno je, da so bile slednje odkrite s strani belih hekerjev, zato napadalci niso razvili postopkov izkoriščanja. CTRL metoda je po drugi strani lahko zelo učinkovita, saj se osredotoči le na tiste vektorje CVSS, kjer izkoristljive ranljivosti v resnici obstajajo.

IS spletnega gostitelja je drugačen. Sestavlja ga veliko število programskih komponent številnih manjših proizvajalcev. Med njimi je tudi nekaj zelo razširjenih produktov (npr. Wordpress, Joomla, MySql, Linux ipd.). Kljub vsemu je povprečno število odkritih ranljivosti na produkt na tem IS bistveno manjše kot pri produktih IS podjetja. Poleg tega je med odkritimi ranljivostmi bistveno več izkoristljivih, in sicer kar trikrat več. To lahko pojasnimo z dejstvom, da manjši proizvajalci nimajo na voljo sredstev, s katerimi bi lahko nagrajevali bele hekerje, zato so ti manj zainteresirani za odkrivanje. Interes napadalcev, da razvijejo ustrezne postopke izkoriščanja za odkrite ranljivosti, je po drugi strani precej večji, saj je verjetnost uspeha bistveno večja. Boljše rezultate metod ABVS in HTAC lahko utemeljimo z dejstvom, da predlagana funkcija *isExploited* predvideva takšno ravnanje napadalcev. Naše izsledke potrjujejo tudi druge podobne raziskave. Tako na primer Nayak s sodelavci [14] ugotavlja, da večina izkoriščenih ranljivosti na strežnikih za gostovanje pripada spletnim aplikacijam in ne operacijskemu sistemu, kjer so nameščene. Ker je IS za spletno gostovanje že v osnovi namenjen vsem uporabnikom z dostopom do spleta, je izpostavljen zelo širokemu krogu oseb, ki lahko izkoristijo ranljivosti. V IS za spletno gostovanje je povprečna TAC vrednost izkoristljivih ranljivosti 11,7 (največja 18).

Raziskave na produktih največjih proizvajalcev programske opreme kažejo, da z vsako novo različico upade število novoodkritih izkoristljivih ranljivosti in da se hkrati zmanjšuje tudi njihov delež [14]. Še pred nekaj leti je bil pri produktih Microsofta delež izkoristljivih ranljivosti 15-%, v naši raziskavi, kjer smo zajeli le podatke iz obdobja 2010 – 2016, je ta delež le še 3-%. Pri spletnih aplikacijah, kjer se neprestano pojavljajo novi ponudniki, je ta delež znatno višji. V primeru IS za spletno gostovanje, ki smo ga ovrednotili v tem poglavju, je povprečen delež izkoristljivih ranljivosti 36-%.

Metoda CTRL se je v vseh opazovanih okoljih izkazala za najučinkovitejšo. Ker pri določanju prioritet uporablja podatke o znanih izkoristljivih ranljivostih, daje prednost

tistim vektorjem CVSS, ki so pogosteje izkoristljivi. Ranljivosti v primeru IS podjetja pripadajo manjšemu številu različnih vektorjev CVSS in pri nekaterih ni prav nobene izkoristljive. Možno je, da so te ranljivosti v večini odkrili beli hekerji. Takšne anomalije metoda CTRL dobro nevtralizira, zato je učinkovitejša od drugih. S tem smo potrdili, da je metrika *ExploitationRatio*, ki jo predlaga Nayak s sodelavci [14], precej uporabnejša pri ocenjevanju tveganj kot metoda CVSS.

Na osnovi zgornjih ugotovitev lahko zaključimo, da sta metodi ABVS in HTAC uporabni v okoljih, kjer IS sestavljajo programski produkti številnih manjših proizvajalcev in s širokim krogom uporabnikov. Ker se področje spletnih tehnologij še vedno močno razvija, lahko tudi v prihodnje pričakujemo ponudbo številnih novih programskih produktov, ki sodijo v to skupino.

Po drugi strani moramo izpostaviti tudi omejitve metod ABVS in HTAC. Uporaba teh metod ni priporočljiva v reguliranih okoljih, kjer je nabor programske opreme standardiziran in ga v večini sestavljajo zreli programski produkti vodilnih proizvajalcev programske opreme na svetu.

V celoti gledano smo z rezultati raziskave potrdili izsledke drugih raziskovalcev, ki ugotavljajo, da lahko z upoštevanjem lastnosti napadalcev bolje ocenimo tveganja [17]. Potrdili smo ugotovitve številnih raziskav, da metoda CVSS ni najboljša pri določanju prioritet ranljivostim [9, 12]. Hkrati smo s tem k obstoječim metodam za proaktivno zagotavljanje varnosti prispevali nove [1].

7.4.3 Družbenoekonomski vidiki uporabe predlaganih metod

Čeprav se je metoda CTRL v vseh primerih izkazala za najučinkovitejšo, obstaja problem njene realizacije v praksi. Za široko uporabo te metode bi potrebovali prosto dostopne podatke o izkoristljivih ranljivostih. V osnovi naj bi jih zagotavljal že sistem CVSS preko časovnih atributov, a so ti podatki na voljo le proti plačilu pri nekaterih ponudnikih tovrstnih vsebin. Nekaj časa so bili prosto dostopni v podatkovni zbirki OSVDB, a je ta pred nekaj leti prenehala delovati. Problem pomanjkanja tovrstnih podatkov izpostavljajo številni strokovnjaki s področja varnosti [13]. Ker za vzpostavitev prosto dostopne zbirke izkoristljivih ranljivosti očitno ni dovolj interesa in podpore, je lahko metoda ABVS dobra alternativa pri določenih tipih IS. Predvsem to velja za IS, ki jih sestavljajo številne spletne aplikacije manjših proizvajalcev programske opreme. Takšni sistemi so v praksi danes zelo pogosti.

Izsledki naše raziskave so koristni predvsem za proizvajalce skenerjev ranljivosti, ki

prioritete še vedno določajo na osnovi ocen CVSS. Metoda ABVS se je namreč v določenih primerih izkazala za precej učinkovitejšo od metode CVSS. Poleg tega za praktično realizacijo metode ABVS zadostujejo le prosto dostopni podatki zbirke NVD.

Zaključek

8

8.1 Povzetek vsebine

Disertacija se osredotoča na proaktivno zagotavljanje varnosti v IS, kjer je ključnega pomena, da pravočasno prepoznamo grožnje in preprečimo napade. Osnova vsakega napada so ranljivosti v IS, zato jih moramo dobro razumeti in vzpostaviti učinkovit sistem za njihovo obvladovanje. Najprej moramo odstraniti tiste ranljivosti, ki predstavljajo največje tveganje. Skrbniki informacijskih sistemov zato potrebujejo učinkovito metodo določanja prioritet, s katero kar najhitreje in v največji možni meri omejijo tveganja. V praksi se za te namene pogosto uporablja ocena CVSS, ki je opredeljena z istoimenskim standardom in predstavlja resnost grožnje ranljivosti. Žal je več študij pokazalo, da ta ocena ni najprimernejša za te namene, saj ničesar ne pove o tem, ali je ranljivost v praksi izkoristljiva. Potrebni so torej alternativni pristopi z boljšimi sposobnostmi predvidevanja izkoristljivih ranljivosti. Poleg tega je v večji meri neraziskano, kako učinkovita je v resnici metoda CVSS v primerjavi z drugimi metodami. In tu so glavni prispevki disertacije.

V 2. poglavju opredelimo ključne pojme in povezave med njimi. Ob tem opišemo glavne akterje na področju ranljivosti in spoznanja o njihovem delovanju. Predstavimo tudi pomembnejše javno dostopne zbirke ranljivosti, ki so zelo pomembne pri zagotavljanju varnosti. V 3. poglavju sledi predstavitev metod za obvladovanje tveganj na različnih razvojnih stopnjah IS.

Študije s področja zagotavljanja računalniške varnosti dokazujejo, da vključevanje značilnosti napadalcev v oceno tveganja izboljša njeno točnost. Zato smo v poglavju 4 zasnovali model za določanje prioritet ranljivostim, ki upošteva lastnosti napadalcev. V ta namen smo opredelili funkcijo izkoriščanja, ki upošteva definicijo grožnje in je odvisna tako od lastnosti ranljivosti kot tudi od napadalca.

V nadaljevanju je bilo potrebno preveriti uporabnost metode. V 5. poglavju zato razvijemo in predstavimo model za merjenje učinkovitosti metod določanja prioritet ranljivostim, saj v literaturi podobnih modelov nismo našli. Resnično tveganje za IS predstavljajo izkoristljive ranljivosti, zato smo kazalnik za merjenje učinkovitosti metod zasnovali na preverjanju izpostavljenosti izkoristljivih ranljivosti grožnjam. Rezultati vrednotenja metode so potrdili našo domnevo o smiselnosti vključevanja lastnosti napadalcev v metodo določanja prioritet. Pokazalo se je, da je podatek o številu tipičnih napadalcev, ki lahko izkoristijo ranljivost, boljši indikator izkoristljivih ranljivosti od CVSS ocene.

V 6. poglavju smo metodo nadgradili. Če v poglavju 5 usmerjamo pozornost na omejeno število tipičnih napadalcev, ki jih znamo opisati z nekaj osnovnimi atributi, se tokrat usmerjamo na skrito populacijo napadalcev. Z veliko množico naključno ustvarjenih napadalcev poizkušamo ugotoviti, kateri vektorji ranljivosti so najbolj izkoristljivi. Ob tem si pomagamo z javno dostopnimi podatki o ranljivostih in s simulacijo na agentnem modelu. Metoda, ki sloni na upoštevanju frekvenc izkoriščanih vektorjev ranljivosti, se je izkazala za še precej učinkovitejšo od tiste, ki smo jo razvili v 5. poglavju. Na podlagi spoznanj v raziskavi ter praktičnih potreb po preglednem ocenjevanju ranljivosti v zaključku predlagamo svoj sistem ocenjevanja ranljivosti z imenom ABVS.

V zadnjem, 7. poglavju, smo predlagane metode preverili na tipičnih IS, ki jih pogosto srečamo v praksi. Ugotovili smo, da je učinkovitost metod odvisna od več dejavnikov v okviru IS. Podali smo kritično oceno naših ugotovitev ter jih primerjali z dosežki sorodnih raziskav. Poglavje smo zaključili z oceno možnosti praktične uporabe predlaganih metod.

8.2 Bodoče raziskovalne priložnosti

Med raziskovanjem je bil glavni poudarek na iskanju učinkovitejše metode določanja prioritet od ocene CVSS ter iskanju načina merjenja učinkovitosti. Ob tem smo večinoma usmerjali pozornost na opisovanje ranljivosti z atributi CVSS oz. z vektorji CVSS. Poleg tega smo se z raziskavo prepričali, da na osnovi podatkov v vektorjih CVSS ni prav veliko več možnosti za izboljšave pri predvidevanju izkoristljivih ranljivosti. S predlagano metodo smo se namreč zelo približali optimalni metodi na osnovi vektorjev. To pa nikakor ne pomeni, da ne moremo izkoristljivih ranljivosti predvidevati še bolje. Z vključevanjem dodatnih podatkov bi verjetno lahko ustvarili še učinkovitejše metode določanja prioritet. Priložnost vidimo predvsem v naslednjih smereh:

Povezava metode s podatki o popularnosti programskih produktov. Metodo določanja prioritet, ki upošteva lastnosti napadalcev, bi v prihodnje lahko dopolnili s podatki o popularnosti oz. razširjenosti posameznih programskih produktov. Dodatni podatki bi prispevali k večjim razlikam med ranljivostmi z enakimi vektorji CVSS. S tem bi verjetno hitreje prepoznali izkoristljive ranljivosti.

Prilagajanje metode za posamezna okolja. Informacija o potencialnih napadalcih je

kontekstno odvisna informacija. V različnih organizacijah in uporabniških okoljih različno dojemajo grožnje za IS. Metodo določanja prioritet bi lahko nadgradili z informacijami o specifični populaciji verjetnih napadalcev. V tem primeru bi za vektorje CVSS morali izračunati prilagojene frekvence izkoriščanja oz. prilagoditi ocene ABVS, podobno kot je to predvideno pri okoljskih atributih CVSS.

Odprta so ostala tudi nekatera zanimiva vprašanja, ki jih med raziskovanjem nismo uspeli preveriti oz. na katera nismo uspeli najti ustreznih odgovorov:

Predvidevanje izkoristljivih ranljivosti za vektorje z nizko frekvenco izkoriščanja. Pri nekaterih vektorjih CVSS, ki se redko pojavijo v realnosti, smo zaznali nizko frekvenco izkoriščanja, kar pomeni, da jim naša metoda ne daje prav visoke prioritete. Hkrati se je v več primerih izkazalo, da so te ranljivosti izkoristljive. Prav verjetno je, da ranljivosti s takšnimi vektorji najdemo v primeru naprednih vztrajnih groženj. Odprto ostaja vprašanje, kako nam lahko lastnosti napadalcev pomagajo pri odkrivanju takšnih izkoristljivih ranljivosti.

Nadgradnja metode na verzijo standarda CVSSv3. Standard CVSSv3 v opis ranljivosti dodaja nekaj novih atributov in s tem poizkuša izboljšati kakovost opisa ter razlikovanje med ranljivostmi. V zbirki NVD je v tem trenutku še premalo podatkov za preizkus metode na novi verziji standarda.

Dodatek: Statistična analiza

A

A.1 Primerjava učinkovitosti metod

Z-test za primerjavo povprečij dveh velikih neodvisnih vzorcev je opredeljen s spodnjo enačbo.

$$Z = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Vrednosti $\bar{x}_1$ in $\bar{x}_2$ sta povprečji vzorcev, izraza μ_1 in μ_1 pa sta povprečji populacij. Testirali smo ničto hipotezo $H_0 : \mu_1 = \mu_2$ proti alternativni hipotezi $H_1 : \mu_1 < \mu_2$.

$$\alpha = 0.01 \rightarrow Z_\alpha = \pm 2.326348$$

Podatki o eksperimentu in skripte v programskem jeziku R za ponovitev analize so na voljo na naslednjem spletnem naslovu: https://3ad.si/doktorat/.

Primer 1: Ali je povprečna izpostavljenost izkoristljivih ranljivosti IS za vsaj 9 % nižja pri uporabi metode HTAC kot pri uporabi metode CVSS oz. ali je metoda HTAC za 9 % učinkovitejša od metode CVSS?

HTAC: $n_1 = 1000, \bar{x}_1 = 20.545, s_1 = 3.493$
CVSS: $n_2 = 1000, \bar{x}_2 = 23.080, s_2 = 3.570$
$Z = -2,898111$
$p - vrednost = 0$

Ker je $Z = -2,898111 < -2,326348$, zavrnemo hipotezo H_0. Z dovolj veliko gotovostjo (stopnja tveganja $\alpha \leq 0,01$) lahko sklepamo, da je metoda HTAC za 9 % učinkovitejša od metode CVSS. Uporabljeni so podatki iz tabele 5.7.

Primer 2: Ali je povprečna izpostavljenost izkoristljivih ranljivosti IS za vsaj 16 % nižja pri uporabi metode ABVS kot pri uporabi metode CVSS oz. ali je metoda ABVS za

16 % učinkovitejša od metode CVSS?

ABVS: $n_1 = 1000, \bar{x}_1 = 18.760, s_1 = 3.238$
CVSS: $n_2 = 1000, \bar{x}_2 = 22.992, s_2 = 3.501$
$Z = -3,668884$
$p - vrednost = 0$

Ker je $Z = -3,668884 < -2,326348$, zavrnemo hipotezo H_0. Z dovolj veliko gotovostjo (stopnja tveganja $\alpha \leq 0,01$) lahko sklepamo, da je metoda ABVS za 16 % učinkovitejša od metode CVSS. Uporabljeni so podatki iz tabele 6.3.

Primer 3: Ali je povprečna izpostavljenost izkoristljivih ranljivosti IS za vsaj 14 % nižja pri uporabi metode CTRL kot pri uporabi metode ABVS oz. ali je metoda CTRL za 16 % učinkovitejša od metode ABVS?

CTRL: $n_1 = 1000, \bar{x}_1 = 15.788, s_1 = 2.707$
ABVS: $n_2 = 1000, \bar{x}_2 = 18.760, s_2 = 3.238$
$Z = -2,589473$
$p - vrednost = 0$

Ker je $Z = -2,589473 < -2,326348$, zavrnemo hipotezo H_0. Z dovolj veliko gotovostjo (stopnja tveganja $\alpha \leq 0,01$) lahko sklepamo, da je metoda CTRL za 16 % učinkovitejša od metode ABVS. Uporabljeni so podatki iz tabele 6.3.

LITERATURA

[1] Denis Trček. Computationally Supported Quantitative Risk Management for Information Systems. In *Performance Models and Risk Management in Communications Systems*, pages 55–77. Springer, 2010. doi: 10.1007/978-1-4419-0534-5_3.

[2] David Jelenc, Eva Zupančič, Denis Trček, Franc Solina, and Jaro Berce. Obvladovanje tehničnih in gospodarsko-družbenih vidikov interneta stvari v slovenskem prostoru, 2011.

[3] Secunia Research. Vulnerability review 2017. Technical report, Flexera Software, 2017.

[4] Stefan Frei, Bernhard Tellenbach, and Bernhard Plattner. 0-day patch exposing vendors (in) security performance. In *BlackHat Europe, Amsterdam, NL*, 2008. URL http://www.techzoom.net/publications/0-day-patch/index.en.

[5] Charlie Miller. The Legitimate Vulnerability Market Inside the Secretive World of 0-day Exploit Sales. *In Sixth Workshop on the Economics of Information Security*, pages 1–10, 2007.

[6] Bruce Schneier. The Vulnerabilities Market and the Future of Security. *Forbes*, 2012. URL http://www.forbes.com/sites/bruceschneier/2012/05/30/the-vulnerabilities-market-and-the-future-of-security/.

[7] Sokratis K. Katsikas. Risk management. In *Computer and Information Security Handbook*, pages 905–927. Elsevier, Waltham, MA, USA, 2013. ISBN 9780123943972. doi: 10.1016/B978-0-12-394397-2.00053-2. URL http://www.sciencedirect.com/science/article/pii/B9780123943972000532.

[8] Christian Frühwirth and Tomi Männistö. Improving CVSS-based vulnerability prioritization and response with context information. In *3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009*, pages 535–544. IEEE, 2009. ISBN 9781424448418. doi: 10.1109/ESEM.2009.5314230.

[9] Mehran Bozorgi, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Beyond Heuristics : Learning to Classify Vulnerabilities and Predict Exploits. *KDD '10 Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 105–113, 2010. doi: 10.1145/1835804.1835821. URL http://dl.acm.org/citation.cfm?id=1835821.

[10] Peter Mell, Karen Scarfone, and Sasha Romanosky. Common Vulnerability Scoring System. *IEEE Security & Privacy Magazine*, 4:85–89, 2006. ISSN 1540-7993. doi: 10.1109/MSP.2006.145.

[11] Hannes Holm, Mathias Ekstedt, and Dennis Andersson. Empirical Analysis of System-Level Vulnerability Metrics through Actual Attacks. *IEEE Transactions on Dependable and Secure Computing*, 9(6):825–837, 2012. ISSN 1545-5971. doi: 10.1109/TDSC.2012.66. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84866600214&partnerID=tZOtx3y1.

[12] Luca Allodi and Fabio Massacci. Comparing vulnerability severity and exploits using case-control studies. *ACM Transactions on Information and System Security*, 17(1):1–20, 2014. ISSN 10949224. doi: 10.1145/2630069. URL http://dl.acm.org/citation.cfm?doid=2660572.2630069.

[13] Hannes Holm and Khalid Khan. An expert-based investigation of the Common Vulnerability Scoring System. *Computers & Security*, 53:18–30, 2015. ISSN 0167-4048. doi: 10.1016/j.cose.2015.04.012. URL http://dx.doi.org/10.1016/j.cose.2015.04.012.

[14] Kartik Nayak, Daniel Marino, Petros Efstathopoulos, and Tudor Dumitra. Some Vulnerabilities Are Different Than Others. In *Research in Attacks, Intrusions and Defenses (RAID)*, pages 426–446. Springer International Publishing, 2014. doi: 10.1007/978-3-319-11379-1_21.

[15] Hamza Ghani, Jesus Luna, and Neeraj Suri. Quantitative assessment of software vulnerabilities based on economic-driven security metrics. In *2013 International Conference on Risks and Security of Internet and Systems (CRiSIS)*, pages 1–8. IEEE, 2013. ISBN 978-1-4799-3488-1. doi: 10.1109/CRiSIS.2013.6766361. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6766361.

[16] Robin Gandhi, Anup Sharma, William Mahoney, William Sousan, Qiuming Zhu, and Phillip Laplante. Dimensions of Cyber-Attacks: Social, Political, Economic, and Cultural. *IEEE Technology and Society Magazine*, 30(1):28 – 38, 2011. doi: 10.1109/MTS.2011.940293.

[17] Lotfi Ben Othmane, Rohit Ranchal, Ruchith Fernando, Bharat Bhargava, and Eric Bodden. Incorporating attacker capabilities in risk estimation and mitigation. *Computers and Security*, 51:41–61, 2015. ISSN 01674048. doi: 10.1016/j.cose.2015.03.001. URL http://dx.doi.org/10.1016/j.cose.2015.03.001.

[18] David Evans and Sal Stolfo. The Science of Security. *CO-PUBLISHED BY THE IEEE COMPUTER AND RELIABILITY SOCIETIES*, 9(3):16–17, 2011.

[19] Christian V. Lundestad and Anique Hommels. Software vulnerability due to practical drift. *Ethics and Information Technology*, 9(2):89–100, oct 2006. ISSN 1388-1957. doi: 10.1007/s10676-006-9123-1. URL http://www.springerlink.com/index/10.1007/s10676-006-9123-1.

[20] J. A. Whittaker and R. Ford. How to think about security. *Security & Privacy, IEEE*, 4(2):68–71, 2006. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1621064.

[21] Ari Takanen, P. Vuorijärvi, Marko Laakso, and J. Röning. Agents of responsibility in software vulnerability processes. *Ethics and Information Technology*, 6(2):93–110, 2004. URL http://www.springerlink.com/index/r54850p327233608.pdf.

[22] Wiliam A. Arbaugh, Wiliam L. Fithen, and John McHugh. Windows of vulnerability: A case study analysis. *IEEE Computer Society*, 33(12): 52–59, 2000. doi: 10.1109/2.889093. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=889093.

[23] Stefan Frei, Dominik Schatzmann, Bernhard Plattner, and Brian Trammell. Modelling the Security Ecosystem - The Dynamics of (In) Security. In *Economics of Information Security and Privacy*, pages 79–106. Springer US, 2010. doi: 10.1007/978-1-4419-6967-5_6. URL http://www.techzoom.net/publications/security-ecosystem/.

[24] Leyla Bilge and Tudor Dumitras. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 833–844, 2012. ISBN 9781450316514. URL http://users.ece.cmu.edu/$\sim$tdumitra/public_documents/bilge12_zero_day.pdfhttp://doi.acm.org/10.1145/2382196.2382284.

[25] Stuart Schechter. How to Buy Better Testing Using. In *Proceedings of the International Conference on Infrastructure Security*, pages 73–87. Springer-Verlag, 2002. URL http://www.springerlink.com/index/KH7FEVX6PPB7XY0X.pdf.

[26] Andy Ozment. Bug auctions: Vulnerability markets reconsidered. *Third Workshop on the Economics of Information Security*, pages 1–23, 2004. URL http://www.andyozment.com/papers/weis04-ozment-bugauc-slides.pdf.

[27] A. Arora and R. Telang. Economics of software vulnerability disclosure. *IEEE Security and Privacy Magazine*, 3(1):20–25, 2005. ISSN 1540-7993. doi: 10.1109/MSP.2005.12. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1392695.

[28] George Aguilera, H. C. Chen, Elnara Iskandarova, and Li Zhang. *Full disclosure vs non disclosure of security vulnerabilites in the security research industry*. A capstone paper submitted as partial fulfillment of the requirements for the degree of masters in interdisciplinary telecommunications, University of Colorado, 2002. URL https://drachma.colorado.edu/dspace/handle/123456789/100.

[29] E. Levy. Approaching zero. *Security & Privacy, IEEE*, pages 0–1, 2004. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1324603.

[30] Stefan Frei, Martin May, Ulrich Fiedler, and Bernhard Plattner. Large-scale vulnerability analysis. In *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense - LSAD '06*, pages 131–138, New York, 2006. ACM Press. ISBN 1595935711. doi: 10.1145/1162666.1162671. URL http://portal.acm.org/citation.cfm?doid=1162666.1162671.

[31] Microsoft. Microsoft Security Intelligence Report. Technical report, Microsoft, 2012.

[32] D. McKinney. Vulnerability bazaar. *Security & Privacy, IEEE*, pages 69–73, 2007. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4402452.

[33] Andy Jones and Ashenden Debi. *Risk management for computer security: protecting your network and information assets*. Elsevier Butterworth Heinemann,

Newton, MA, 2005. ISBN 9780750677950. URL http://store.elsevier.com/Risk-Management-for-Computer-Security/Andy-Jones/isbn-9780750677950/.

[34] S. Evans and D. Heinbuch. Risk-based systems security engineering: Stopping attacks with intention. *Security & Privacy, IEEE*, 2(6):59–62, 2004. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1366121.

[35] V. Igure and R. Williams. Taxonomies of attacks and vulnerabilities in computer systems. *Communications Surveys & Tutorials*, 10(1):6–19, 2008. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4483667.

[36] John Steven. Threat Modeling - Perhaps It's Time. *IEEE Security & Privacy*, 8(3):83 – 86, 2010. doi: 10.1109/MSP.2010.110. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5470962.

[37] Jaziar Radianti. A Study of a Social Behavior inside the Online Black Markets. *2010 Fourth International Conference on Emerging Security Information, Systems and Technologies*, pages 189–194, jul 2010. doi: 10.1109/SECURWARE.2010.22. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5633627.

[38] Jaziar Radianti and Eliot Rich. Refinement of Supply and Demand Model for Vulnerability Black Market. In *2010 International System Dynamics Conference Seoul*, 2010.

[39] Chris Grier, Andreas Pitsillidis, Niels Provos, M. Zubair Rafique, Moheeb Abu Rajab, Christian Rossow, Kurt Thomas, Vern Paxson, Stefan Savage, Geoffrey M Voelker, Lucas Ballard, Juan Caballero, Neha Chachra, Christian J Dietrich, Kirill Levchenko, Panayiotis Mavrommatis, Damon McCoy, and Antonio Nappa. Manufacturing Compromise: The Emergence of Exploit-as-a-Service. *Proceedings of the 2012 ACM conference on Computer and communications security - CCS '12*, pages 821–832, 2012. doi: 10.1145/2382196.2382283. URL http://dl.acm.org/citation.cfm?doid=2382196.2382283.

[40] University of Texas at Dallas Alvaro A. Cárdenas, College Park Tudor Dumitras, University of Maryland, University of Luxembourg Thomas Engel, University of Luxembourg Jérôme François, AT&T Paul Giura, RSA Laboratories Ari Juels, HP Labs Pratyusa K. Manadhata, RSA Laboratories Alina Oprea, University of Texas at Dallas Cathryn Ploehn, University of Luxembourg Radu State, University of Texas at Dallas Grace St. Clair, AT&T Wei Wang, and RSA Laboratories Ting-Fang Yen. Big Data Analytics for Security Intelligence. Technical Report September, Cloud Security Alliance, 2013.

[41] Panos Kampanakis. Security Automation and Threat Information-Sharing Options. *IEEE Security &Privacy*, 12(October):42–51, 2014.

[42] Éric Dubois, Patrick Heymans, Nicolas Mayer, and Raimundas Matulevičius. *A Systematic Approach to Define the Domain of Information System Security Risk Management*, pages 289–306. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-12544-7. doi: 10.1007/978-3-642-12544-7_16. URL https://doi.org/10.1007/978-3-642-12544-7_16.

[43] Computer Security Division. Guide for Conducting Risk Assessments, 2011.

[44] Pratyusa K. Manadhata. *An Attack Surface Metric Thesis*. PhD thesis, Carnegie Mellon University, 2008.

[45] Roger G. Johnston. Being Vulnerable to the Threat of Confusing Threats with Vulnerabilities. *Journal of Physical Security*, 4(2):30–34, 2010.

[46] Chris Badger, Lee Johnson, and David Waltermire. NIST Special Publication (SP) 800-150 Guide to Cyber Threat Information Sharing, 2016.

[47] D. Trček. Security Metrics Foundations for Computer Security. *The Computer Journal*, 53(7):1106–1112, 2009. ISSN 0010-4620. doi: 10.1093/comjnl/bxp094. URL http://comjnl.oxfordjournals.org/cgi/doi/10.1093/comjnl/bxp094.

[48] Iztok Starc and Denis Trček. Towards Quantitative Risk Management for Next Generation Networks. In *Telecommunication Economics*, pages 229–239. Springer, 2012. ISBN 978-3-642-30381-4.

[49] Jeffrey R. Jones. Estimating Software Vulnerabilities. *IEEE Security & Privacy*, 5:28–32, 2007. doi: 10.1109/MSP.2007.81.

[50] Denis Trček and Borut Likar. Information Systems Security Management By Deployment of Innovations Management Techniques. In *Proceedings of the 12th International Conference on Applied Computer and Applied Computational Science (ACACOS '13)*, pages 21–24, Kuala Lumpur, Malaysia, 2013. ISBN 9781618041715. URL http://www.wseas.us/e-library/conferences/2013/Malaysia/ACACOS/ACACOS-01.pdf.

[51] Andy Ozment. *Vulnerability discovery & software security*. PhD thesis, University of Cambridge, 2007. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.167.2921&rep=rep1&type=pdf.

[52] O. H. Alhazmi and Y. K. Malaiya. Prediction capabilities of vulnerability discovery models. In *RAMS '06. Annual Reliability and Maintainability Symposium, 2006.*, pages 343–352, 2006. doi: 10.1109/RAMS.2006.1677355. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1677355.

[53] O. H. Alhazmi and Y. K. Malaiya. Modeling the Vulnerability Discovery Process. *16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)*, pages 129–138, 2005. doi: 10.1109/ISSRE.2005.30. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1544728.

[54] O. H. Alhazmi, Y. K. Malaiya, and I. Ray. Measuring, analyzing and predicting security vulnerabilities in software systems. *Computers & Security*, 26(3):219–228, 2007. ISSN 01674048. doi: 10.1016/j.cose.2006.10.002. URL http://linkinghub.elsevier.com/retrieve/pii/S0167404806001520.

[55] Hyunchul Joh, Jinyoo Kim, and Yashwant K. Malaiya. Vulnerability Discovery Modeling Using Weibull Distribution. In *2008 19th International Symposium on Software Reliability Engineering (ISSRE)*, pages 299–300. Ieee, 2008. doi: 10.1109/ISSRE.2008.32. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4700345.

[56] Awad Younis, HyunChul Joh, and Yashwant Malaiya. Modeling learningless vulnerability discovery using a folded distribution. In *Proc. of SAM*, volume 11, pages 617–623, 2011. URL http://www.lidi.info.unlp.edu.ar/WorldComp2011-Mirror/SAM8765.pdf.

[57] S. W. Woo, O. H. Alhazmi, and Y. K. Malaiya. An analysis of the vulnerability discovery process in web browsers. In *Proceedings of the 10th IASTED International Conference*, pages 172–177, 2006. URL http://www.cs.colostate.edu/$\sim$malaiya/pub/wooSEA_2006.pdf.

[58] O. H. Alhazmi and Y. K. Malaiya. Quantitative vulnerability assessment of systems software. In *Annual Reliability and Maintainability Symposium, 2005. Proceedings*, pages 615–620. Ieee, 2005. ISBN 0-7803-8824-0. doi: 10.1109/RAMS.2005.1408432. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1408432.

[59] Andy Ozment. Improving Vulnerability Discovery Models. In *Proceedings of the 2007 ACM workshop on Quality of protection*, pages 6–11, 2007. ISBN 9781595938855. doi: 10.1145/1314257.1314261.

[60] V. H. Nguyen and Fabio Massacci. An Independent Validation of Vulnerability Discovery Models. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, page 11, 2012. URL http://arxiv.org/abs/1203.5830.

[61] H. C. Joh. *Quantitative analyses of software vulnerabilities*. PhD thesis, Colorado State University, 2012. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:QUANTITATIVE+ANALYSES+OF+SOFTWARE+VULNERABILITIES#0.

[62] Stephan Neuhaus, Thomas Zimmermann, Christian Holler, and Andreas Zeller. Predicting vulnerable software components. In *Proceedings of the 14th ACM conference on Computer and communications security CCS 07*, pages 529–540. ACM, 2007. ISBN 9781595937032. doi: 10.1145/1315245.1315311. URL http://portal.acm.org/citation.cfm?doid=1315245.1315311.

[63] Riccardo Scandariato, James Walden, Aram Hovsepyan, and Wouter Joosen. Predicting Vulnerable Software Components via Text Mining. *IEEE Transactions on Software Engineering*, 40(10):993–1006, 2014. doi: 10.1109/TSE.2014.2340398.

[64] Peter Mell, Karen Scarfone, and Sasha Romanosky. The Common Vulnerability Scoring System (CVSS) and Its Applicability to Federal Agency Systems. *NIST Interagency Report 7435*, 2007. URL http://csrc.nist.gov/publications/nistir/ir7435/NISTIR-7435.pdf.

[65] FIRST. Common Vulnerability Scoring System v3.0: Specification Document, 2015. URL https://www.first.org/cvss/cvss-v30-specification-v1.7.pdf.

[66] Siv Hilde Houmb, Virginia N. L. Franqueira, and Erlend A. Engum. Quantifying security risk level from CVSS estimates of frequency and impact. *Journal of Systems and Software*, 83(9):1622–1634, 2010. ISSN 01641212. doi: 10.1016/j.jss.2009.08.023. URL http://dx.doi.org/10.1016/j.jss.2009.08.023.

[67] Aristeidis Chatzipoulidis, Dimitrios Michalopoulos, and Ioannis Mavridis. Information infrastructure risk prediction through platform vulnerability analysis. *Journal of Systems and Software*, 106:28–41, 2015. ISSN 01641212. doi: 10.1016/j.jss.2015.04.062. URL http://dx.doi.org/10.1016/j.jss.2015.04.062.

[68] Qixu Liu and Yuqing Zhang. VRSS: A new system for rating and scoring vulnerabilities. *Computer Communications*, 34(3):264–273, 2011. ISSN 01403664. doi: 10.1016/j.comcom.2010.04.006. URL http://dx.doi.org/10.1016/j.comcom.2010.04.006.

[69] Qixu Liu, Yuqing Zhang, Ying Kong, and Qianru Wu. Improving VRSS-based vulnerability prioritization using analytic hierarchy process. *Journal of Systems and Software*, 85(8):1699–1708, 2012. ISSN 01641212. doi: 10.1016/j.jss.2012.03.057. URL http://dx.doi.org/10.1016/j.jss.2012.03.057.

[70] NIST. FIPS PUB 199: Standards for Security Categorization of Federal Information and Information Systems, 2004.

[71] Su Zhang, Doina Caragea, and Xinming Ou. An empirical study on using the national vulnerability database to predict software vulnerabilities. In *Database and Expert Systems Applications*, pages 217–231. Springer Berlin Heidelberg, 2011. URL http://www.springerlink.com/index/474061656n62p508.pdf.

[72] Steve Christey and Brian Martin. Buying Into the Bias : Why Vulnerability Statistics Suck, 2013. URL https://www.blackhat.com/us-13/archives.html#Martin.

[73] Viet Hung Nguyen and Le Minh Sang Tran. Predicting vulnerable software components with dependency graphs. In *Proceedings of the 6th International Workshop on Security Measurements and Metrics - MetriSec '10*, pages 1–8. ACM, 2010. ISBN 9781450303408. doi: 10.1145/1853919.1853923. URL http://portal.acm.org/citation.cfm?doid=1853919.1853923.

[74] Sung-Whan Woo, HyunCchul Joh, Omar H. Alhazmi, and Yashwant K. Malaiya. Modeling vulnerability discovery process in Apache and IIS HTTP servers. *Computers & Security*, 30(1):50–62, 2011. ISSN 01674048. doi: 10.1016/j.cose.2010.10.007. URL http://linkinghub.elsevier.com/retrieve/pii/S0167404810000908.

[75] Alexander Kott and Curtis Arnold. The Promises and Challenges of Continuous Monitoring and Risk Scoring. *IEEE Security and Privacy*, 11(1):90–93, 2013. doi: http://doi.ieeecomputersociety.org/10.1109/MSP.2013.19.

[76] Timothy Casey. Threat Agent Library Helps Identify Information Security Risks, 2007. URL https://communities.intel.com/docs/DOC-1151.

[77] Charles P. Pfleeger and Shari Lawrence Pfleeger. *Security in Computing (4th Edition)*. Prentice Hall PTR, Upper Saddle River, NJ, 2006. ISBN 0132390779.

[78] Lukas Ruf, Anthony Thorn, Tobias Christen, Beatrice Gruber, and Roland Portmann. Threat Modeling in Security Architecture, The Nature of Threats, 2003. URL https://www.isss.ch/fileadmin/publ/agsa/ISSS-AG-Security-Architecture_Threat-Modeling_Lukas-Ruf.pdf.

[79] Christopher Alberts and Audrey Dorofee. OCTAVE Threat Profiles, 1999. URL http://www85.homepage.villanova.edu/timothy.ay/MIS2040/OCTAVEthreatProfiles[1].pdf.

[80] Donald L. Buckshaw, Gregory S. Parnell, Willard L. Unkenholz, Donald L. Parks, James M. Wallner, and Sami O. Saydjari. Mission Oriented Risk and Design Analysis of Critical Information Systems. *Military Operations Research*, 10:19–38, 2005. URL https://www.researchgate.net/publication/233644703_Mission_Oriented_Risk_and_Design_Analysis_of_Critical_Information_Systems.

[81] Michael J. North, Nicholson T. Collier, Jonathan Ozik, Eric R. Tatara, Charles M. Macal, Mark Bragen, and Pam Sydelko. Complex adaptive systems modeling with Repast Simphony. *Complex Adaptive Systems Modeling*, 1(1):1–26, 2013. doi: 10.1186/2194-3206-1-3. URL http://link.springer.com/article/10.1186/2194-3206-1-3.

[82] Pontus Johnson, Robert Lagerstrom, Mathias Ekstedt, and Ulrik Franke. Can the Common Vulnerability Scoring System be Trusted? A Bayesian Analysis. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1–1, 2016. ISSN 1545-5971. doi: 10.1109/TDSC.2016.2644614. URL http://ieeexplore.ieee.org/document/7797152/.