Celovita zaščita spletnih aplikacij

Gregor Polančič, Romana Vajde, Tatjana Welzer, Boštjan Brumen
Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Inštitut za informatiko
{gregor.polancic, romana.vajde, welzer, bostjan.brumen}@uni-mb.si

Povzetek

Pomemben dejavnik uspešnega e-poslovanja je kakovostna zaščita spletnih aplikacij. Prispevek obravnava zaščito spletnih aplikacij z vidika njene zasnove in okolja spletne aplikacije. Predstavljen je tehnološko neodvisen, procesno naravnan in celovit pristop, namenjen zasnovi varne spletne aplikacije. Pristop opredeli najpogostejše grožnje, proces obravnavanja groženj in podaja pregled uveljavljenih ukrepov za njihovo onemogočanje.

Abstract

Overall Web Application Security

Security presents an important factor for success of e-business. The following contribution presents web application security from the point of view of its design and web environment. The presented approach for improving web application security is process oriented, consistent and technology independent. Being based on web application design, anatomy of an attack and potential threats, it establishes proper countermeasures.

1 Uvod

Uspeh poslovne spletne aplikacije je v veliki meri odvisen od zaupanjem uporabnika v varnost poslovanja. Zaupanje v aplikacijo je povezano z zaščito, ki jo zagotavlja aplikacija. Primer, ki ilustrira pomen zaupanja v aplikacijo, je nakup izdelka prek spletne trgovine, kjer se plačilo izvede z uporabo plačilne kartice. V omenjenem primeru predstavlja številka kupčeve kartice zasebno informacijo, ki se mora ob izvedbi nakupa prenesti prek svetovnega spleta vse do prodajalca. Tveganje za uporabnika se pojavi, če je številka kartice na kateremkoli mestu prenosa ali aplikacije neprimerno zaščitena. V primeru kraje številke kartice utrpita posledice kupec, ki je denarno oškodovan, in posredno tudi prodajalec, saj je omajan njegov ugled in zaupanje v poslovanje z njegovo spletno trgovino.

Stoodstotno zaščito aplikacij je praktično nemogoče doseči, saj je zaščita odvisna od številnih dejavnikov in je vedno le tako dobra, kot je dober njen najšibkejši člen. Šibki členi v zagotavljanju varnosti spletne aplikacije so najpogosteje posledica neodkritih varnostnih področij in nesistematičnega pristopa k zagotovitvi ustrezne zaščite. Zato so v prispevku zbrane in predstavljene metode, ki na celovit in sistematičen način izboljšajo zaščito spletne aplikacije. V drugem poglavju je predstavljena varnostna problematika osnovnih sestavnih delov spletne aplikacije. Metode, ki zvišajo kakovost zaščite spletne aplikacije, so podane v tretjem poglavju. Prispevek je zaključen s četrtim poglavjem, ki podaja sklepne ugotovitve, povezane z uporabo metod zagotavljanja zaščite, predstavljenih v članku.

2 Varnostna problematika spletne aplikacije

Na varnost (spletne) aplikacije vplivajo naslednji koncepti [1] [3]:

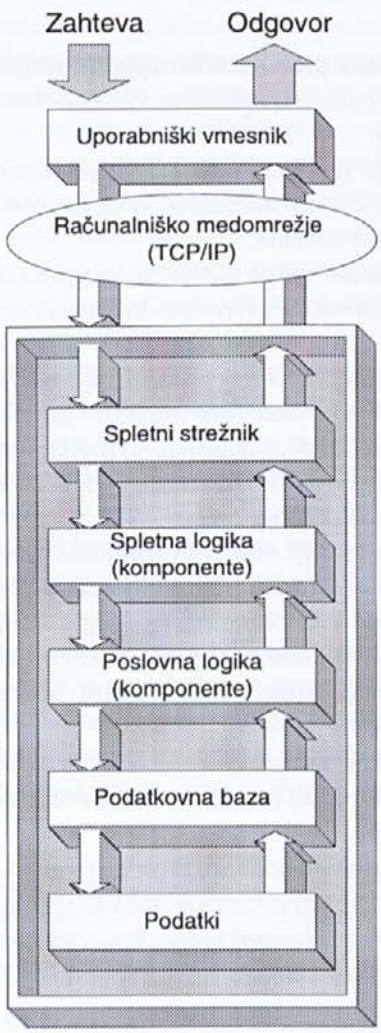
- **overjanje** (angl. *authentication*) predstavlja postopek potrjevanja identitete in upravičenosti osebe, objekta ali sistema;
- **pooblastitev** (angl. *authorization*) predstavlja proces odobritve ali zavrnitve dostopa do zaščitenega vira;
- **nadzor** (angl. *auditing*) preprečuje zmožnost zanikanja izvedbe določene akcije ali transakcije;
- **zaupnost** (angl. *confidentiality*) predstavlja lastnost, ki zagotavlja dostop do informacij le pooblaščenim osebam ali postopkom;
- **celovitost** (angl. *integrity*) sistemski pristop popolne in ne le delne obravnave nekega pojava glede na izbrani vidik obravnave, ki upošteva zapletenost in kompleksnost pojava. Zagotavlja zaščito pred naključnimi ali namernimi spremembami podatkov;
- **razpoložljivost** (angl. *availability*) predstavlja lastnost sistema, da je razpoložljiv pooblaščenim uporabnikom.

Zaščita spletnih aplikacij je problematična predvsem iz dveh razlogov. Prvi razlog je velik krog (potencialnih) uporabnikov spletne aplikacije. Posledično je velik tudi potencialni krog napadalcev na spletno aplikacijo.

Drugi razlog je (pogosto) zapletena in raznolika zasnova spletne aplikacije. Čeprav deluje navzven kot celota, je pogosto sestavljena iz programskih rešitev,

ki temeljijo na različnih tehnologijah, predpisih ali zasnovah. Raznolikost programskih rešitev, ki sestavljajo spletno aplikacijo, negativno vpliva na njeno zaščito. Posamezne programske rešitve (komponente) imajo pogosto ločen razvoj in vzdrževanje, zato predstavljajo večini razvijalcev črne škatle z nepoznanimi in specifičnimi »varnostnimi luknjami«. Za razliko od drugih vrst aplikacij ni orodij za potrditev (angl. *validation)* zaščite, ki bi lahko prepoznala spletno aplikacijo kot celoto in jo z vidika varnosti tudi ovrednotila [2].

Ker je krog uporabnikov spletne aplikacije stalnica, se prispevek osredinja na izboljšanje zaščite, ki je povezana z zasnovo spletne aplikacije. Zato so v nadaljevanju predstavljene logične komponente, ki so skupne večini spletnih aplikacij (slika 1) [2][4].



Slika 1: **Logična zasnova spletne aplikacije**

2.1 Uporabniški vmesnik

Uporabniški vmesnik predstavlja del spletne aplikacije, katerega podatki se prek omrežja prenesejo na spletnega odjemalca.

Programska zasnova uporabniškega vmesnika temelji na dveh osnovnih tehnologijah. Aktualni tehnologiji za prikaz statičnih podatkov sta označevalni jezik XHTML in stilne predloge (angl. *Cascading Style Sheets)*. Druga vrsta tehnologije omogoča izvajanje kode na strani odjemalca (npr. Java Applet, JavaScript, ActiveX, VB Script in Flash).

Uporabniški vmesnik se najpogosteje izdela z uporabo grafičnih razvojnih orodij, s samodejnim proizvajanjem programskega koda (generatorji kode) ali z uporabo ponovno uporabnih programskih izdelkov. Kljub številnim prednostim, med katere spada pospešen in poenostavljen razvoj, ima lahko takšen razvoj z vidika varnosti tudi slabe lastnosti. Programski kod, ki ga izdelajo samodejni proizvajalci koda, in polizdelki drugih proizvajalcev lahko zaradi splošnosti vsebujejo številne varnostne luknje. Tveganje lahko predstavljajo tudi razvijalci uporabniškega vmesnika, ki so pogosto grafični oblikovalci in z vidika varnosti nimajo ustreznega znanja.

2.2 Spletni strežnik

Spletni strežnik je dvosmerni posrednik podatkov med odjemalskim in strežniškim delom spletne aplikacije. Prestreza in obravnava vse uporabnikove zahteve (angl. *request),* posreduje odgovore (angl. *respond)* in upravlja z uporabnikovimi sejami. Za spletni strežnik se najpogosteje uporabi obstoječi programski proizvod, kot je na primer Apache, iPlanet ali MS IIS.

Zaradi velike konkurence med razvijalci spletnih strežnikov je njihova kakovost in varnost običajno na visoki ravni. Večje tveganje predstavlja nepravilno skrbništvo strežnikov. Privzeti uporabniški računi in privzete varnostne nastavitve, kot so vrata, storitve ali protokoli, so pogosto aktivni, kljub temu, da se ne uporabljajo.

2.3 Spletna logika

Poglavitni namen spletne logike (spletnih komponent) je, da podatke poslovne logike pretvori v obliko, ki ustreza spletnemu uporabniškemu vmesniku in da podatke, ki prispejo s strani spletnega odjemalca, posreduje ustreznim poslovnim komponentam. Spletna logika se najpogosteje realizira s strežniškimi skriptnimi jeziki, kot so ASP, JSP in PHP.

Čeprav je priporočljivo, da spletna logika ne komunicira neposredno s podatkovnimi viri (v primeru uporabe arhitekture MVC Model 2 [4]), ji tehnološka zasnova tega ne preprečuje (na primer poizvedovanje po podatkovni bazi iz JSP strani). Posledica je lahko komunikacijski kaos med posameznimi komponentami in ostalimi viri spletne aplikacije, kar negativno vpliva na varnost spletne aplikacije. Pri izdelavi spletne logike se zahteva velika mera previdnosti, saj se njeni rezultati prikazujejo na spletnem odjemalcu (brskalniku).

2.4 Poslovna logika

Poslovna logika (poslovne komponente) predstavlja »uporabno« jedro spletne aplikacije, saj zagotavlja množico »uporabnih« funkcij. Poslovna logika komunicira z operacijskim sistemom, spletno logiko, podatkovnimi viri in drugimi obstoječimi sistemi. Lahko jo zagotovijo obstoječe poslovne aplikacije, kot je na primer SAP. V primeru specifičnih funkcij sta trenutno najbolj odmevni tehnologiji ».NET« in EJB *(Enterprise Java Beans)*. Komponente na podlagi omenjenih tehnologij so primerne za spletno okolje, ker se izvajajo v okolju aplikacijskega strežnika, ki prevzame številne funkcije, kot so varnost, nadgradljivost (angl. *scalability)* in trajnost podatkov. Ker poslovna logika pogosto temelji na obstoječi kodi različnih proizvajalcev, za njeno zaščito običajno skrbijo različne skupine. Tveganje lahko predstavljajo neprepoznane vstopne točke v posamezne komponente in raznolikost varnostnih politik posameznih komponent.

2.5 Podatkovna baza

Podatkovna baza je računalniško podprta, večuporabniška, formalno definirana in centralno nadzorovana zbirka podatkov [1]. Podatkovna baza komunicira s poslovno logiko, operacijskim sistemom in z drugimi podatkovnimi viri. Trajno hranjenje podatkov, ki ga zagotovi podatkovna baza, skoraj vedno temelji na obstoječih in preizkušenih proizvodih, kot so Oracle, SQL Server ali MySQL.

Podobno kot pri spletnih strežnikih, predstavlja tudi pri podatkovni bazi poglavitno grožnjo njeno nepravilno skrbništvo. K varnosti pripomore tudi dobro zasnovan podatkovni model.

3 Metode za izboljšanje zaščite spletne aplikacije

Čeprav ne manjka metod in orodij, ki pripomorejo k zaščiti spletne aplikacije, ne more nobeden izmed njih zagotoviti celovite zaščite. Visoko stopnjo zaščite spletne aplikacije je moč zagotoviti s celovitim in sistematičnim pristopom. V nadaljevanju predstavljene metode lahko dopolnimo še z obstoječimi avtomatskimi orodji in kontrolnimi seznami (angl. *checklist)*, namenjenimi potrditvi zaščite spletnih aplikacij.

3.1 Poznavanje splošnih načel zaščite (spletnih) aplikacij

Obstaja nekaj splošnih načel zaščite, ki so neodvisna od tehnologije in zasnove spletne aplikacije [3]:

a) **zmanjšanje števila točk napada:** Množico virov, do katerih lahko dostopa potencialni napadalec, je potrebno zmanjšati. Priporočljivo je, da se vrata, protokoli in storitve, ki se ne uporabljajo, odstranijo.
b) **uporaba najmanjšega možnega privilegija:** Uporabnik naj ima dostop le do funkcij spletne aplikacije, ki so potrebne za opravljanje njegovega dela.
c) **načrtovanje obrambe v globino:** Za razliko od enega nivoja zaščite, predstavlja večnivojska zaščita učinkovitejšo obrambo, saj zadrži napadalca tudi v primeru, če mu uspe prodreti skozi prvi sloj zaščite.
č) **vhodnim podatkom se ne sme zaupati**, saj lahko predstavljajo orodje napadalca.
d) **preverjanje na vhodnih vratih:** Overjanje in pooblastitev uporabnika naj se izvede kar se da hitro, najbolje ob prvem stiku uporabnika z aplikacijo.
e) **varna odpoved:** Odpoved aplikacije ne sme nuditi napadalcu informacij o tehnološki zasnovi aplikacije. Le-te mu lahko koristijo ob snovanju napada.
f) **zaščita najšibkejšega člena:** Segment sistema, ki je najbolj ranljiv, predstavlja najšibkejši člen zaščite in mora biti dodatno zavarovan.
g) **zaščita privzetih vrednosti:** Privzeti uporabniški računi in ostale nastavitve ne smejo temeljiti na najnižji stopnji zaščite, saj s tem predstavljajo šibek člen v zaščiti.

3.2 Poznavanje metod napadalcev

»Če poznamo sovražnika, se lahko bolje pripravimo na obrambo« je splošen pristop, ki se obnese tudi na področju zaščite spletnih aplikacij. Poznavanje metod, ki jih uporabljajo napadalci, omogoča odkrivanje in analizo napadov ter izdelavo ustreznih protiukrepov. Čeprav uporablja vsak napadalec svojevrsten

način napada na spletno aplikacijo, se vsak napad izvrši po naslednjih splošnih korakih [7]:

a) **opazovanje:** Prva faza v izvedbi napada na spletno aplikacijo je opazovanje, ki se najpogosteje izvede z avtomatskim preverjanjem vrat (angl. *port scan)* spletnega strežnika. Namen preverjanja je odkrivanje odprtih vrat in pridobitev privzetih oziroma »pozdravnih strani«.
b) **zbiranje informacij:** Na podlagi odprtih vrat in pridobljenih »pozdravnih strani« lahko napadalec pridobi podatke o programski in strojni opremi strežnika. Napadalec običajno nadaljuje z identifikacijo povezav, strukture in logike spletne aplikacije. Iz spletnih strani, do katerih lahko dostopa, poskuša pridobiti čim več informacij, ki niso namenjene končnim uporabnikom in so običajno skrite v komentarjih.
c) **iskanje ranljivih mest:** Napadalec poskuša odkriti čim večje število skript (programskega koda) na strani odjemalca in dinamičnih funkcij spletne aplikacije. V primeru, da naleti na ranljivo kodo (leta je najpogosteje posledica napake razvijalca), poskuša razviti način za globlji prodor v spletno aplikacijo.
č) **načrtovanje napada:** Na podlagi informacij, ki so bile pridobljene »pasivno« (način, ki ga spletna aplikacija ne more zaznati), napadalec izbere in podrobneje načrtuje napad.
d) **izvedba napada:** Nadalec izvede napad prek najšibkejšega člena spletne aplikacije. V primeru uspešnega napada poskuša poenostaviti ponovne vdore v sistem in prikriti znake napada.

3.3 Poznavanje najpogostejših groženj

Groženj, ki pretijo spletni aplikaciji, je več vrst, mednje spadajo [3]:

- **prevare** (angl. *spoofing*): vstop v aplikacijo z uporabo napačne identitete,
- **vtikovanja** (angl. *tampering*): nepooblaščeno spreminjanje podatkov,
- **zanikanje** (angl. *repudiation*): prikritje izvedbe določene akcije ali transakcije,
- **odkrivanje informacij** (angl. *information disclosure*): neželeno odkritje zasebnih ali zaupnih podatkov,
- **odpoved storitev** (angl. *denial of service*): preobremenitev in posledično odpoved sistema (strežnika),
- **zviševanje pooblastil** (angl. *elevation of privilege*): zloraba identitete uporabnika z višjimi pooblastili.

Čeprav organizacije trošijo številne vire za zagotovitev zaščite, pogosto ne dosežejo želenih rezultatov. Posledica necelovite zaščite je lahko napad na najbolj ranljivem mestu spletne aplikacije. V nadaljevanju so predstavljena najbolj ranljiva področja spletne aplikacije in predlagani ustrezni protiukrepi [6][8].

3.3.1 Nepreverjeni vhodni podatki

V primeru, da se spletna aplikacija zanaša na neškodljivost podatkov spletne zahteve (piškotki, URL naslov, podatki elementov spletnih obrazcev in skrita polja), lahko napadalec s sestavljanjem »škodljivih« nizov doseže priklic zaupnih podatkov ali odpoved delovanja spletne aplikacije.

Preverjanje vhodnih podatkov na nivoju spletnega odjemalca z uporabo skriptnih jezikov ne zagotavlja zadostne podpore, saj se lahko onemogoči ali obide. Priporočljivo je, da se vsi vhodni podatki preverjajo z enotno vodenim mehanizmom za omejevanje, zavračanje in saniranje vhodnih podatkov, ki se nahaja na strani strežnika [3].

3.3.2 Pomanjkljiv nadzor nad dostopom

Nadzor nad dostopi oziroma pooblastitev opredeljuje, do katerih podatkov in storitev lahko dostopa pooblaščeni uporabnik, ki najpogosteje pripada določeni skupini uporabnikov. Prekinjen nadzor nad dostopom oziroma neučinkovita pooblastitev omogoči napadalcu, da lahko dostopi do podatkov in storitev, ki so namenjeni le pooblaščenim uporabnikom. Posledice so lahko spremenjeni ali izbrisani zaupni podatki, uporaba zaupnih storitev in v najslabšem primeru prevzem nadzora nad skrbništvom spletne aplikacije.

Vzpostavitev učinkovitega overjanja zahteva definiranje varnostne politike, ki opredeljuje tipe uporabnikov in njihova pooblastila v kontekstu podatkov in storitev spletne aplikacije. Sistem overjanja mora prestati obsežna testiranja, ki temeljijo na poskusih dostopa različnih vlog do zaščitenih virov.

3.3.3 Pomanjkljivi uporabniški računi in pomankljivo upravljanje s sejami

Kljub ustreznemu preverjanju identitete uporabnika in dobro izdelanemu sistemu pooblastitev so lahko zaščitena področja spletnih aplikacij še vedno ranljiva.

Ker spletni protokol HTTP ne ohranja stanja (angl. *stateless protocol)*, morajo spletne aplikacije same zago-

toviti mehanizem za ohranjanje sej (angl. *session*). Lete uporabniku, kljub ločenim spletnim zahtevam, omogočijo navidezno enovito interakcijo s spletno aplikacijo. Uporabniška seja se običajno identificira s piškotkom (angl. *cookie*), ki se hrani na odjemalcu. V primeru neustrezno zaščitenih podatkov piškotka, ga lahko napadalec ukrade in s tem prevzame identiteto overjenega uporabnika.

Druga težava je pomanjkljivo overjanje v »ozadju« spletne aplikacije (angl. *backend authentication*), ki opredeljuje, kako se spletna aplikacija overi na drugih virih, kot so podatkovne baze ali spletne storitve. Pogosta pomanjkljivost uporabniških računov (uporabniška imena in gesla) je, da se hranijo v nezaščitenih tekstovnih datotekah.

3.3.4 Škodljive skripte

Škodljive skripte (angl. *Cross-site scripting, XSS*) predstavljajo grožnjo, s pomočjo katere lahko napadalec prek spletne aplikacije (ki nezadostno preverja vhodne podatke) pošlje škodljivo kodo (na primer JavaScript) tretji osebi. V primeru, da tretja oseba, ki zaupa spletni aplikaciji, zažene škodljivo kodo, lahko izda svojo identiteto napadalcu. Navzkrižne skripte pogosto povzročijo namestitev škodljivega programa (trojanskega konja) na odjemalca. Priporočljiva zaščita pred škodljivimi skriptami je natančna specifikacija dovoljenih vrednosti na osnovi »pozitivne varnostne politike« [3] (angl. *positive security policy*) vhodnih podatkov.

3.3.5 Odprtine za ukazne injekcije

Odprtine za ukazne injekcije (angl. *command injection gaps*) so ranljiva področja spletnih aplikacij, ki omogočajo napadalcu, da škodljivo kodo s pomočjo spletne aplikacije posreduje na tretji sistem. Tipičen primer je SQL injekcija. Z uporabo SQL injekcij lahko napadalec izvaja povpraševanja in programe (angl. *stored procedures*) v podatkovni bazi. SQL injekcije je moč uporabiti, če se SQL ukazi proizvajajo dinamično z uporabo vrednosti parametrov. Preverjanje vhodnih podatkov in minimalna možna pooblastila pri dostopih do podatkovne baze se izkažeta za najboljšo zaščito pred SQL injekcijami.

3.3.6 Nepravilna obravnava izjem

Običajen postopek spletne aplikacije v primeru izjeme je, da se pošlje odjemalcu sporočilo o napaki. Privzeti prikaz izjem ni priporočljiv, ker lahko potencialnemu napadalcu poda številne informacije o spletni aplikaciji (zasnova, uporabljene aplikacije, tehnologije, SQL stavki, dostopi do drugih sistemov in podobno). Priporoča se izdelava enotnega mehanizma za obravnavo izjem, ki lovi izjeme in uporabnikom prikaže le splošne podatke o stanju spletne aplikacije, brez podrobnosti o njeni zasnovi.

3.3.7 Napačna uporaba tajnopisja (kriptografije)

Večina spletnih aplikacij hrani in obdeluje množico zaupnih podatkov, kot so uporabniški računi, številke kreditnih kartic ali osebni podatki. Kljub temu, da so dosegljive številne kakovostne kriptografske knjižnice, lahko nepravilna uporaba tajnopisa povzroči varnostne luknje. Ne glede na stopnjo zaščite niso podatki nikoli stoodstotno zaščiteni. Zato se hranjenju zaupnih podatkov izogibamo, če zadostuje, da hranimo vrednosti njihovih zgostitvenih funkcij (angl. *hash value*).

Priporoča se priklic in razkritje zaupnih podatkov v trenutku, ko jih spletna aplikacija dejansko potrebuje. Po uporabi se naj zaupen podatek prikrije ali odstrani. Zaupnih podatkov ni priporočljivo shranjevati v programskem kodu, datotekah ali v piškotkih na strani odjemalca.

3.3.8 Oddaljeno skrbništvo spletne aplikacije

Oddaljeno skrbništvo (angl. *remote administration*) omogoča, da se spletna aplikacija upravlja prek spletnega odjemalca. Mehanizem je učinkovit in udoben, vendar predstavlja šibek člen v zaščiti in zato priljubljen cilj napadalcev. Priporoča se najvišja možna zaščita skrbnikovega uporabniškega vmesnika in podatkov konfiguracije. Pooblastila skrbnika se naj omejijo. Če je mogoče, naj se dodatno razdelijo med različne uporabnike in omejijo na določene IP naslove.

3.3.9 Pomanjkljive nastavitve spletnih in aplikacijskih strežnikov

Spletni in aplikacijski strežniki igrajo pomembno vlogo v celoviti zaščiti spletne aplikacije. Njihova poglavitna naloga je obdelava vsebine in vključevanje drugih aplikacij, ki zagotavljajo spletni aplikaciji vsebino in funkcionalnosti. Težave oziroma ranljivosti na nivoju strežnikov se posredno odražajo tudi na spletni aplikaciji. Najpogostejše težave so: uporaba šibkih in privzetih gesel, neustrezna zaščita map in datotek, znane in nezaščitene varnostne luknje in najrazličnejše razvojne nastavitve, ki ne spadajo v končno (produkcijsko) izvajalno okolje. Splošni model zaščite vključuje upoštevanje

»priporočil za izboljšanje zaščite«, ki jih običajno izdela proizvajalec in vsebujejo napotke za:

- uporabo namestitvenih in varnostnih funkcij,
- ugašanje odvečnih storitev, protokolov in vrat,
- nastavitev vlog, dovoljenj in računov,
- vzpostavitve dnevnikov in obveščanj.

V času vzdrževanja opreme je poleg omenjenih opravil treba nenehno nadgrajevati strežnike z varnostnimi dodatki in spremljati varnostne informacije.

3.4 Proces prepoznavanja in ovrednotenja groženj

Namen procesa je prepoznavanje in ovrednotenje groženj, ki so odvisne od arhitekturne in tehnološke zasnove spletne aplikacije. Ovrednotenju potenciala posamezne grožnje sledi načrtovanje ustreznega protiukrepa. Ker je programska oprema podvržena nenehnim spremembam, je proces stalen in ga je priporočljivo uporabiti v vseh fazah življenjskega cikla programske opreme. Neodvisno od velikosti in vrste organizacije, se lahko definira splošen proces, sestavljen iz naslednjih aktivnosti [3]:

a) **prepoznavanje virov, ki jih je treba zaščititi:** Prepoznavanje podatkov in storitev, ki zahtevajo zaščito, sega vse od podatkov v podatkovni bazi do spletnih strani.

b) **analiza arhitekture spletne aplikacije:** Aktivnost procesa je namenjena analizi in dokumentaciji funkcij, arhitekture, tehnologij in izvajalnega okolja spletne aplikacije. Dokumentiranju funkcionalnih zahtev aplikacije ustrezajo diagrami primerov uporabe (angl. *use cases diagram*), kjer se lahko v opis samega primera uporabe vključi še vidik zaščite. Z vidika varnosti se obravnavajo še arhitekturni diagrami, ki pripomorejo k boljšemu razumevanju delovanja in tehnološke zasnove spletne aplikacije.

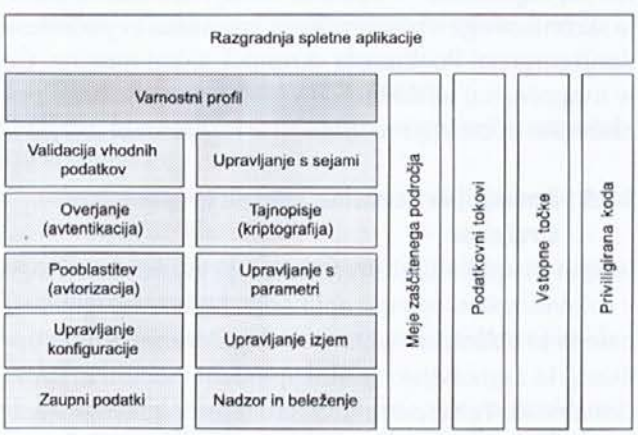
c) **razgradnja spletne aplikacije:** Razgradnja spletne aplikacije se izvede zato, da bi analizirali pogosto ranljiva področja spletne aplikacije (slika 2).

Razgradnja spletne aplikacije, na podlagi katere se izdela varnostni profil spletne aplikacije, vključuje analizo:

- vseh potencialnih »zunanjih« in »notranjih« vstopnih točk v spletno aplikacijo,[1]
- vhodnih in izhodnih podatkovnih tokov posamezne komponente,
- podatkovnih tokov med različnimi komponentami (na primer podatkovni tok med poslovno komponento in podatkovno bazo),
- delov »privilegirane kode«. Le-ta je namenjena dostopanju do zaščitenih virov in izvajanju varovanih operacij.[2]

č) **prepoznavanje groženj:** Na podlagi prejšnjih korakov procesa se prepozna čim večje število potencialnih groženj spletne aplikacije. Prepoznavanje groženj pogosto temelji na formalnih metodah, ki so lahko dodatno podprte s skupinskimi tehnikami ustvarjalnega mišljenja, kot je na primer »možganska nevihta« (angl. *brainstorming*). Metode prepoznavanja groženj so naslednje:

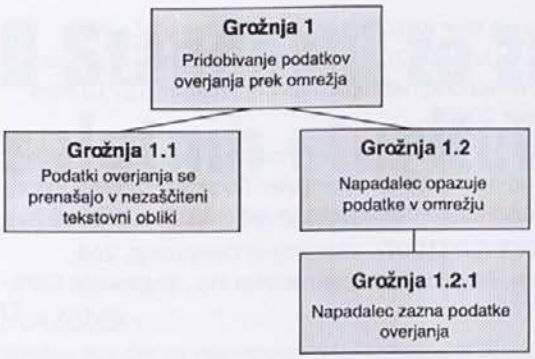
- **model STRIDE** [3] (angl. *spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege*): To je ciljno usmerjen pristop, kjer se identificirajo cilji napadalca (na primer, ali lahko napadalec prisluškuje spletnemu omrežju z namenom, da ugotovi skrbnikovo uporabniško ime in geslo spletnega strežnika).
- **seznam kategoriziranih groženj:** Na podlagi izdelanega seznama splošnih groženj se identificirajo dejanske grožnje spletne aplikacije.
- **drevesa napadov** predstavljajo hierarhičen zapis morebitnih groženj. Zapis omogoča večjo stopnjo abstraktnosti in ponovne uporabe zapisa groženj (slika 3).



Slika 2: **Varnostni profil spletne aplikacije**

[1] *Vstopne točke so lahko zunanje, kot so na primer HTTP vrata, ali notranje, kot so dostopi do podatkovne baze. Običajno so notranje točke še bolj kritične, saj jih pogosto spregledamo in jih spreten napadalec lahko uporabi za svoj napad.*

[2] *Primeri zaščitenih virov so: DNS strežnik, spremenljivke okolja, dnevniki dogodkov, datotečni sistem, sporočilne vrste, merilniki zmogljivosti, register, spletne storitve in podobno.*



Vzorec napada	SQL injekcije
Cilji napada	Izvajanje škodljive kode
Pogoji za napad	Nepreverjeni vhodni podatki
Tehnike za izvedbo napada	1. Identifikacija spletne aplikacije, ki je ranljiva na področju preverjanja vhodnih podatkov 2. Izdelava in zagon škodljive kode, ki bo sposobna delovati znotraj varnostnih omejitev spletne aplikacije 3. Konstrukcija vhodnega podatka, ki bo prenesel škodljivo kodo in preusmeril izvajanje spletne aplikacije
Posledice napada	Izvajanje škodljivih operacij v spletni aplikaciji

Slika 3: **Primer drevesa napadov (levo) in vzorca napadov (desno) [3]**

- **vzorci napadov** predstavljajo splošen in strukturiran opis groženj in so zato uporabni v različnih področjih oziroma različnih spletnih aplikacijah (slika 3).

d) **dokumentacija groženj:** Za dokumentiranje groženj se priporoča uporaba predlog s strukturiranim opisom grožnje, sestavljenim iz različnih atributov.

e) **ovrednotenje groženj:** Zadnji korak v procesu prepoznavanja in ovrednotenja groženj je ocenitev stopnje ogroženosti s strani posamezne grožnje. Na osnovi ocenitve se najprej izvedejo protiukrepi za grožnje, ki predstavljajo največje tveganje. Obstaja več metod za ocenjevanje tveganja groženj, med najpreprostejše spada formula:

tveganje = verjetnost grožnje × potencial škode grožnje

kjer se za verjetnost grožnje in potencial škode, ki lahko nastane zaradi grožnje, uporabi lestvica od 1 do 10 (10 predstavlja največjo verjetnost oziroma potencial, 1 najmanjšo). Produkt takšne lestvice sega od 1 do 100. Vrednosti produkta se pogosto razdelijo na tri področja: visoko, srednje in nizko. Slaba lastnost metode je možnost podajanja subjektivnih rezultatov.

Alternativa k zgornji metodi je metoda, ki ponuja že vnaprej definirane odgovore na ocenjevanje atributov grožnje. Primer metode je Microsoft DREAD [6]. Metoda tveganje posameznih atributov grožnje oceni že z vnaprej definiranimi odgovori. Atributi grožnje so na primer:

- potencial škode, ki lahko nastane zaradi grožnje,
- ponovljivost napada,
- stopnja težavnosti izvedbe napada,
- število prizadetih uporabnikov,
- stopnja težavnosti odkritja grožnje.

Vsak atribut ima definirane tri odgovore (ali tri skupine odgovorov), ki opredeljujejo stopnjo ogroženosti na visoko, srednjo in nizko.

4 Sklep

Kljub temu, da predstavlja zaščita spletne aplikacije enega izmed ključnih dejavnikov njenega (ne)uspeha, jih je večina pomanjkljivo zaščitenih [8]. Razlogi za pomanjkljivo zaščito so med drugimi zapletena zasnova spletnih aplikacij, pomanjkanje orodij za ovrednotenje in zagotavljanje zaščite ter napačen pristop k zagotavljanju zaščite. Namen našega prispevka je bil analizirati področja spletne aplikacije, ki vplivajo na njeno varnost in predstavitev metod, ki izboljšajo zaščito. V prispevku so bile predstavljene različne metode, namenjene ovrednotenju in vzpostavitvi zaščite spletne aplikacije. Metode obravnavajo zaščito iz različnih vidikov (vidik napadalca, najbolj pogoste grožnje, splošne in specifične metode zaščite spletne aplikacije). Razvijalcem nudijo sistematičen, celovit in dokumentiran pristop in predlagajo priporočljive protiukrepe. Metode so v večji meri neodvisne od tehnologije in uporabne za različne vrste spletnih aplikacij. Rezultat njihove uporabe pripomore k izboljšanju celovite zaščite spletne aplikacije.

Literatura

1. Terminološki slovar izrazja informatike, Slovensko društvo informatika. [Online] Available: http://www.islovar.org.
2. PETTIT S. (July 2002), »Anatomy of a Web Application, Security Considerations«, White Paper, Sanctum Inc. [Online]. Available: http://www.cgisecurity.com/lib/Web_Server.pdf [October 2003].
3. CURPHEY M., SCAMBRAY J., OLSON E. (June 2003), »Improving Web Application Security, Threats and Countermeasures, Patterns and Practices«, Microsoft. [Online[. Available: http://msdn.microsoft.com/library/en-us/ dnnetsec/html/ThreatCounter.asp [October 2003].

4. Polančič G. (2003), »Načini implementacije spletnih aplikacij«, Zbornik strokovnega srečanja OTS 2003.
5. ORRIN S. (February 2003), »Securing the Last Mile in Corporate Security: Web Applications«, Technical Support. [Online]. Available: http://www.naspa.com/PDF/2003/0203/T0302005.pdf [October 2003].
6. DYCK T. (February 2003), »Top 10 Web App Vulnerabilities«, OWASP – The Open Web Application Security Project. [Online]. Available: http://www.eweek.com/article2/0,3959,857317,00.asp [October 2003].
7. »Securing Your Web Applications: Anatomy of a Web Attack«, (July 2002), SPI Dynamics. [Online]. Available: http://www.ebizq.net/topics/security/features/1713.html [October 2003].
8. BAR-GAD I. (2001), »Proven Practices for Securing Web Enabled Applications«, Computer Society Institute.[Online] Available: http:// csiannual.com/pdf/e6.pdf.
9. PFLEGER C.P. (1997), »Security in Computing, 2nd Edition«, Prentice-Hall International Inc., Englewood Cliffs.

▪

Gregor Polančič je doktorski študent na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru, kjer je tudi zaposlen kot asistent za področje informatike. Pri raziskovalnem delu se osredinja na področja razvoja in uporabe spletnih aplikacij, razvoj na podlagi programskih ogrodij ter na odprtokodni model razvoja programske opreme.

▪

Romana Vajde Horvat je docentka na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru, kjer je nosilka predmetov in raziskuje na področjih izboljšanja procesa razvoja programske opreme, kakovosti, vodenja projektov in metod komuniciranja.

▪

Tatjana Welzer je izredna profesorica na Fakulteti za elektrotehniko, računalništvo in informatiko, kjer predava na dodiplomski stopnji in vodi laboratorij za podatkovne tehnologije. Na raziskovalnem področju se ukvarja predvsem s podatkovnimi bazami, kakovostjo podatkov in podatkovnim modeliranjem.

▪

Boštjan Brumen je doktoriral s področja napovedovalnega podatkovnega rudarjenja. Raziskovalno se ukvarja še z varovanjem podatkov in inteligentno analizo podatkov. Je strokovnjak za modeliranje podatkov in za povpraševalni jezik SQL. V okviru pedagoškega dela sodeluje pri vajah iz podatkovnih baz in pri vajah varovanja podatkov.