Seminarska naloga

DREVESA NAPADOV IN RAČUNALNIŠKA VARNOST

PREDMET: Kriptografija in računalnška varnost
Podiplomski študij v letu 2001/02
Fakulteta za računalništvo in informatiko

MENTOR: **Aleksandar Jurišić**
AVTOR: Mitja Golouh
mitja.golouh@genis.si
Tel: (01) 423-78-80

Ljubljana, junij 2003

Kazalo

Kazalo.....
Kazalo primerov.....
1 Uvod.....
2 Drevo napadov.....
  2.1 Zgodovina.....
  2.2 Struktura in pomen.....
  2.3 Od drevesa do grafa.....
3 Razširjeno drevo napadov.....
4 Ponovna uporaba znanja o napadih.....
  4.1 Predloga napada.....
  4.2 Profil napada in knjižnica napadov.....
6 Izdelava drevesa napadov.....
  6.1 Ročno dodajanje.....
  6.2 Uporaba knjižnice napadov.....
  6.3 Računalniška podpora.....
8 Zaključek.....
9 Literatura.....

Kazalo primerov

Primer 1: Drevo napadov za vlom v sef.....
Primer 2: Dostop do varovanih dokumentov na spletnem strežniku.....
Primer 3: Drevo napadov deljene skrivnosti.....
Primer 4: Predloga napada s prekoračitvijo sklada.....
Primer 5: Profil napada.....
Primer 6: Uporaba predloge napada pri dopolnjevanju drevesa napadov.....

1 Uvod

Razvoj tehnologije v preteklih letih nas je počasi privedel do vprašanja, kdo sploh še obvladuje sodobni svet. V računalniškem svetu vse bolj pogosto izhajajo nove različice programske opreme, nastajajo novi standardi, omrežja in čedalje bolj zmogljiva računalniška oprema. Pri vsem tem drvenju je računalniška varnost ena izmed obrobnih in manj omenjanih tem v računalniškem vsakdanjiku. Jemljemo jo kot nekaj samoumevnega: »*nekaj, kar kupiš in imaš*«. Temu ni tako. Varnost ni nikoli popolna in je vedno odvisna od konteksta v katerem jo uporabljamo. Na primer investicija v drago alarmno napravo za avto je močno vprašljiva, če avto parkiramo v dvomljivem delu mesta. Ljudje so že davno postali imuni na zvoke policijskih siren in alarmnih naprav ter ignorirajo vsakršno, še tako glasno alarmno napravo. Edini učinek, ki ga dosežemo je, da se kdo zadere, naj jo že končno ugasnemo in morda v avto vrže najbližjo opeko. Vlomilcu je to okolje domače in se zato ne pusti motiti, tudi če v avtu tuli alarmna naprava.

Realne primere iz vsakdanjika kot je zgornji, si lahko predstavljamo, težje pa je s primeri iz računalniškega sveta, ki ga poznajo le posvečeni posamezniki. Kako investirati ogromno denarja v računalniško varnost zgolj na podlagi nekega priporočila, morda celo neke reklame v računalniški reviji?

Računalniška varnost ni novo področje, saj že nekaj časa obstajajo številne metodologije [3] in tudi ISO standard [7], ki definirajo smernice in postopke za zagotavljanje računalniške varnosti. Osrednji del metodologij računalniške varnosti je obvladovanje tveganja, ki se v praksi izvaja na podlagi kontrolnega seznama (*checklist*). Ta metoda ima številne pomanjkljivosti, predvsem je počasna in dolgotrajna, pogosto povečuje kompleksnost, seznami so statični, njihovo spreminjanje je težko nadzorovati in prav tako ne omogoča učinkovite izmenjave znanja. Kljub naštetim pomanjkljivostim se metoda kontrolnih seznamov s svojo preprostostjo najbolj približa mišljenju končnih uporabnikov.

V seminarju je predstavljen drevesni model, ki poskuša odpraviti pomanjkljivosti metode kontrolnih seznamov ter hkrati obdržati transparentnost in preprostost le teh. Model omogoča zapis in analizo tveganj in je tako lahko odličen pripomoček pri določanju kritičnih točk v varnosti in investicijah za njihovo odpravo. Formalno zapisan model predstavlja skupni jezik vseh sodelujočih pri oblikovanju modela varnosti računalniškega sistema. Lahko ga tudi posplošimo za uporabo izven okolja v katerem je nastal in s tem omogočimo izmenjavo znanja.

2 Drevo napadov

Računalniška varnost je zapleteno opravilo. Skrbeti moramo za osebje, izobraževanje, identifikacijo in avtentikacijo, pravice (access control list), fizično varnost, beleženje, varnostno politiko, vdore z Interneta in tako naprej. Pozornost je potrebno usmeriti na različne konce in različne probleme. Naša naloga je, da probleme predčasno zaznamo (risk assessment) in jih spravimo na sprejemljiv nivo tveganja (risk mitigation). Kateremu problemu najprej posvetimo pozornost? Vložimo ves denar v zaščito in zanemarimo odkrivanje in reakcijo? Kateri del zaščite je najbolj potreben nadgradnje?

Na podobna vprašanje lahko odgovorimo le s poznavanjem sistema. Tu je v veliko pomoč tako imenovano "Drevo napadov" (attack tree), ki se v različnih oblikah in z različnimi imeni v praksi pojavlja že precej časa. Drevo napadov je v literaturi šele pred kratkim predstavljeno kot sistematična metoda za opis varnostnih lastnosti sistema [1]. Informacija o napadu se začne v izhodišču drevesa, kjer je predstavljena grožnja varnosti ali grožnja obstoju podjetja. Načini kako napadalec uresniči grožnjo so postavljeni v nižje ležeča vozlišča.
Podjetje ima tipično skupino ali gozd dreves, ki opisujejo grožnje povezane z vsebino dela, ki ga opravlja. Korensko vozlišče vsakega drevesa predstavlja dogodek, ki lahko resno ogrozi poslanstvo podjetja. Z drevesom naštejemo in opredelimo dejanja, s katerimi lahko napadalec doseže, da se ta dogodek pripeti. Vsaka različna pot skozi drevo napadov nam predstavlja možen napad na podjetje.

2.1 Zgodovina

Drevo napadov kot nov koncept na področju računalniške varnosti temelji na idejah, ki jih je v svoji knjigi Safeware [10] avtorica opisala kot drevo napak (*fault tree*) in izvirajo iz področja zanesljivosti programske opreme (*software saftey*).
Podobne ideje uporabljajo tudi drevesa ciljev in uspehov (*goal tree*, *success tree*), ki so poznana iz računalniškega področja umetne inteligence.

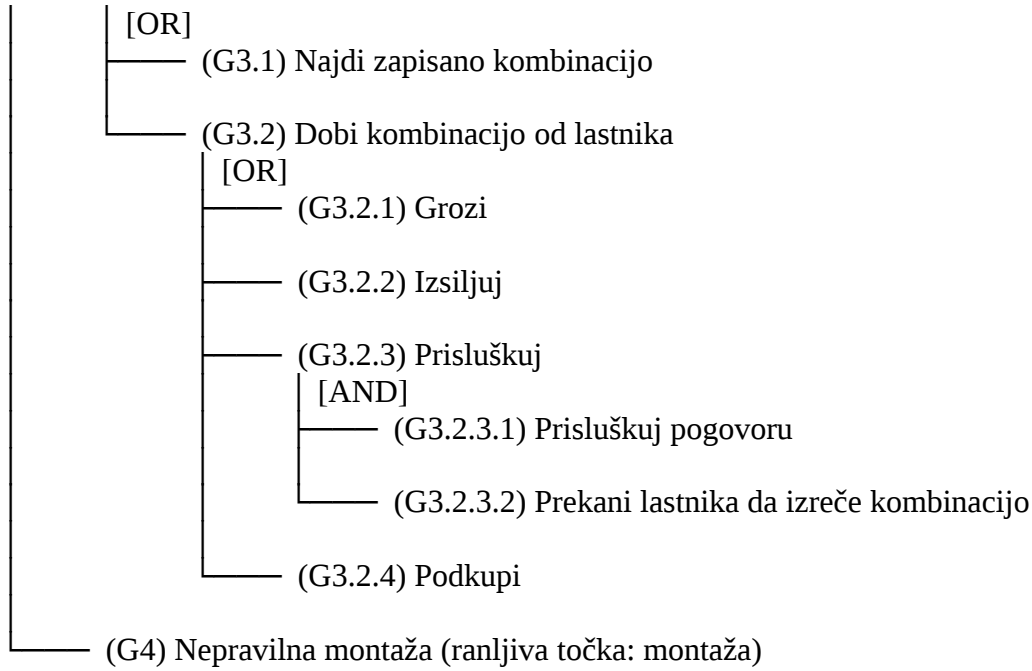
2.2 Struktura in pomen

Drevo napadov formalno zapišemo kot AND/OR drevo. Vsako vozlišče lahko razgradimo na:

- Skupino napadov ali podciljev, kjer velja, da je napad uspešen, če so doseženi vsi podcilji. Razgradnjo predstavimo kot AND-vozlišče.
- Skupino napadov ali podciljev, kjer velja, da je napad uspešen, če je dosežen vsaj en podcilj. Razgradnjo predstavimo kot OR-vozliščem.

Primer 6: Drevo napadov za vlom v sef

(G0) Odpri sef (Cilj)
[OR]
- (G1) Vlomi ključavnico (ranljiva točka: ključavnica)
- (G2) Fizično odpri sef (ranljiva točka: fizična zaščita)
- (G3) Dobi kombinacijo (ranljiva točka: kombinacija sefa)



Primer 1 kaže, da napadalec lahko odpre sef na več načinov: vlomi ključavnico (G1), sef fizično odpre (G2), dobi kombinacijo (G3) ali pa sabotira pravilno montažo in na ta način kasneje zlahka odpre sef (G4). Razvidno je, da je potrebno sef (zaščito) spremljati v celotnem življenjskem ciklu.
Vozlišča AND predstavljajo, da mora napadalec izvesti vsa dejanja, ki so povezana. V primeru, da hoče sef odpreti s kombinacijo, ki jo izve od lastnika s prisluškovanjem (G3.2.3), mora izvesti sledeče:
1. prekaniti lastnika, da izreče kombinacijo (G3.2.3.1)
2. in hkrati prisluškovati v trenutku, ko jo lastnik izreče (G3.2.3.2).
Ranljiva točka je v tem primeru pogojena z več vezanimi dejanji.

Možni scenariji, ki so generirani s tem drevesom so:

1. {G4},
2. {G3.2.4},
3. {G3.2.3.2,G3.2.3.1},
4. {G3.2.2},
5. {G3.2.1},
6. {G3.1},
7. {G2},
8. {G1}.

Razen tretjega napada, ki je del AND vozlišča, so vsi napadi dolžine 1.

Drevo lahko zapišemo v grafični ali tekstualni obliki. V knjigi Secrets and Lies [1] je prikazan nekoliko bolj kompleksen primer drevesa napadov na šifrirano sporočilo PGP s 63 vozlišči. Pri tem je porabljen tekstualni zapis, ker se izkaže, da je grafični zapis netrivialnih primerov nepregleden.
V tekstualni obliki je prikazan tudi primer 2, ki prikazuje možen napad, s katerim si napadalec pridobi dostop do varovanih dokumentov na spletnem strežniku.

Primer 6: Dostop do varovanih dokumentov na spletnem strežniku

1. **(AND)** Pridobi privilegiran dostop do podatkov na spletnem strežniku
    1.1 Identificiraj osnovne podatke domene v kateri se nahaja spletni strežnika (WHOIS)
    1.2 **(OR)** Poišči IP naslov požarnega zidu
        1.2.1 Preišči DNS strežnik, ki je odgovoren za domeno (DNS Interrogation)
        1.2.2 Poišči IP s *scan* programom za identifikacijo požarnega zidu
        1.2.3 Preveri pot do strežnika s pomočjo *traceroute*
    1.3 **(OR)** Preveri odprte poti na požarnem zidu
        1.3.1 Preveri delovanje privzetih poti (listening ports)
        1.3.2 Izvedi širše iskanje vseh odprtih poti na požarnem zidu
    1.4 **(OR)** Identificiraj operacijski sistem in verzijo spletnega strežnika
        1.4.1 Preglej spletne strani, ki jih vrača spletni strežnik
        1.4.2 TCP/IP stack fingerprinting
    1.5 **(OR)** Izkoristi ranljivost spletnega strežnika
        1.5.1 Dostop do varovanih dokumentov preko spletnega strežnika
        1.5.2 Dostop do varovanih dokumentov iz privilegiranega uporabnika v operacijskem sistemu v katerem deluje spletni strežnik.

2.3 Od drevesa do grafa

Drevo napadov ni vedno pravo drevo, ker se lahko ista vozlišča pojavljajo večkrat na različnih delih drevesa. Namesto o drevesih imamo opravka s strukturo acikličnega grafa. Primer 3 nazorno prikazuje zakaj pride do acikličnega grafa na primeru deljene skrivnosti, kjer je za doseganje cilja potrebno pridobiti dva od treh ključev.

Primer 6: Drevo napadov deljene skrivnosti

1 **(OR)** Pridobi skrivnost
    1.1 **(AND)** Pridobi ključ 1 in 2
        1.1.1 Pridobi ključ 1
        1.1.2 Pridobi ključ 2
    1.2 **(AND)** Pridobi ključ 2 in 3
        1.2.1 Pridobi ključ 2
        1.2.2 Pridobi ključ 3
    1.3 **(AND)** Pridobi ključ 1in 3
        1.3.1 Pridobi ključ 1
        1.3.2 Pridobi ključ 3

V splošnem so napadi na računalniške sisteme lahko zelo kompleksni, prihaja na primer tudi do ciklov, zaradi česar moramo drevesno strukturo oziroma strukturo acikličnega grafa nadomestiti z bolj splošnim modelom grafa [6]. Vzrok v kompleksnosti lahko iščemo v povezanosti sistemov in lahkotnosti pri njihovem povezovanju.

3 Razširjeno drevo napadov

Razširitev osnovnega drevesa napravimo tako, da vozliščem dodelimo merljive vrednosti. Vsak list ocenimo z binarno vrednostjo: je možno, ni možno. Vrednosti na listih nato z Boolovo algebro prenašamo navzgor do korenskega vozlišča. Tako lahko dodelimo in uporabimo, katerekoli da/ne vrednosti: lahko/težko, poceni/drago, posebna znanja da/ne, posebna oprema da/ne ali legalno/nelegalno. Za kriterije izberemo dejavnike, s katerimi izločimo skupine potencialnih napadalcev. Na primer če napad zahteva posebna znanja, smo s tem z veliko verjetnostjo izločili preproste zločince in teroriste.

Poleg binarne vrednosti da/ne lahko uporabimo tudi numerične vrednosti. Namesto poceni/drago ocenimo napad z numerično oceno cene napada. Podobno kot da/ne vrednosti, tudi numerične vrednosti prenašamo navzgor po drevesu, glede na sledeča pravila:
1. OR vozliščem priredimo minimalno ceno izmed njegovih otrok
2. AND vozliščem priredimo vsoto cen njegovih otrok
Na ta način dobimo v korenskem vozlišču ceno najcenejšega napada. Če je cena najcenejšega napada v zgornjem primeru 100.000 SIT in držimo v sefu samo 50.000 SIT potem je stopnja varnosti že previsoka (problem ostaja, če napadalec misli, da se v sefu nahaja več kot 100.000 SIT).

Pri modeliranju drevesa napadov v praksi uporabljamo različne kombinacije tako da/ne vrednosti kot zveznih vrednosti. Iz različnih kombinacij pridobimo več informacij o ranljivosti sistema. Primer je lahko iskanje najcenejšega napada, ki ne zahteva nobene posebne opreme. Prav tako je zanimivo iskanje najcenejšega napada, ki ima majhno stopnjo tveganja za napadalca ali najcenejšega napada, ki ima največjo možnost uspeha. Vsakič, ko na drevesu napadov izdelamo poizvedbo z iskanimi lastnostmi, se naučimo več o našem sistemu.

V virtualnem svetu so problematični predvsem napadi, ki izkoriščajo pomanjkljivosti v sistemih, za kar je navadno dovolj internetni priključek in preprost računalnik. Pri njih ne najdemo nekega smisla postavljati merljive vrednosti na primer cene napada, saj so tovrstni programi oziroma programska koda najpogosteje prosto dostopni na Internetu. Napade tega tipa lahko ocenimo z oceno je možno in ni možno, odvisno od tega ali smo na ciljni sistem instalirali popravek, ki pomanjkljivost odprav.
Veliko lažje ovrednotimo napade, ki temeljijo na posebni računalniški opremi. Lep primer je na primer metoda grobe sile pri razbijanju šifriranega sporočila, za katero navadno potrebujemo zelo zmogljive računalnike. Na primer, če za cilj napada določimo kako priti do vsebine šifriranega sporočila v 7 dneh, lahko za ceno napada postavimo na primer računalnik v vrednosti 50 milijonov SIT.

Vrednosti, ki jih priredimo vozliščem, je potrebno posodabljati na določeno obdobje. Računalniška tehnologija se razvija zelo bliskovito, tako da nekdaj nemogoče napade z grobo silo, z danes povprečnim osebnim računalnikom izvedemo v nekaj dneh. Primer so na primer šifrirani Adobe Acrobat 4.0 dokumenti, ki jih danes s povprečnim računalnikom odklenemo v nekaj dnevih.

4 Ponovna uporaba znanja o napadih

Praktična uporaba dreves napadov za opis realnih sistemov je odvisna od zmožnosti ponovne uporabe že odkritih vzorcev napadov. V nadaljevanju so predstavljene tri strukture, ki omogočajo ponovno uporabo:

- predloga napada za opis posameznega tipa napada,
- profil napada, ki združuje podobne predloge napadov v nekem kontekstu in
- knjižnica napadov v katero shranjujemo profile.

4.1 Predloga napada

Predloga napada je definirana kot splošna predstavitev namernega napada, ki se pojavlja v jasno definiranem kontekstu. Vsaka predloga napada vsebuje [2]:

- splošen cilj napada, ki ga opisuje predloga
- seznam predpogojev, ki omogočajo napad
- dejanja, ki izvedejo napad
- indikator – seznam pogojev, ki se izpolnijo, če napad uspe

Predpogoji so kar vrednosti, ki smo jih predstavili pri razširjenem drevesu napadov. Gre za predpostavke o napadalcu ali stanju združbe in njene okolice, ki se morajo uresničiti, da napad uspe. Primeri predpostavk so že omenjene veščine, sredstva (denar), dostop ali znanje, ki ga mora imeti napadalec, nivo tveganja, ki ga napadalec sprejema in tako dalje.
Seznam pogojev, ki se izpolnijo, vključuje vrednoto, ki jo pridobi napadalec in spremembe stanja združbe, ki nastanejo kot posledica izvedenega napada.

Primer 6: Predloga napada s prekoračitvijo sklada

V preteklem desetletju se kot najpogostejša ranljiva točka računalniških programov pojavlja prekoračitev sklada (buffer overflow) [4]. Ob vsakem klicu (pod)programa se na sistemski sklad v sledečem vrstnem redu doda: najprej povratni naslov, vsebina registrov pred klicem in nato lokalne spremenljivke. V primeru, da se vhodni podatek, ki ga poda uporabnik programa, zapiše neposredno v lokalno spremenljivko, se pojavi problem.

Lokalna spremenljivka X ima deklarirano dolžino 200 zlogov. Enako velikost je sistem zanjo rezerviral na skladu. Zlonameren uporabnik na vhodu poda niz dolžine 4000 zlogov, ki se zapiše v spremenljivko X. Prvih 200 zlogov se pravilno zapiše na prostor rezerviran na skladu. Kam se zapiše ostalih 3800 zlogov? Ker ni nobenega preverjanja maksimalne dolžine, se pisanje teh 3800 zlogov nadaljuje na skladu. Najprej prekrijejo prostor lokalnih spremenljivk (pod)programa, ki imajo prostor na skladu rezerviran pred spremenljivko X. Nato prekrijejo prostor, kjer je shranjena vsebina registrov pred klicem (pod)programa. Končno prekrijejo tudi vsebino povratnega naslova in tako dalje, dokler se ne zapiše vseh 3800 zlogov viška.
Ker vhodni niz definira napadalec, lahko z nekaj znanja v teh 4000 zlogih pošlje zlonamerno programsko kodo ter vsebino povratnega naslova prekrije z naslovom svoje programske kode. Programska koda, ki jo vrine v izvajanje, teče s pravicami prvotnega programa. Če ta program teče z administratorjevimi pravicami, kar je v večini primerov tako, ima napadalec odprto pot do celotnega sistema. Tipično si odpre vrata v sistem, ki mu omogočajo, da najprej prikrije svoje

sledi in nato nemoteno nadaljuje s svojim početjem (na primer požene \bin\sh po UNIX-om ali cmd.exe v okolju Windows).

Predlogo napada s prekoračitvijo sklada lahko zapišemo kot:

Cilj:
- Izkoristi ranljivost s prekoračitvijo sklada za izvedbo zlonamerne operacije na ciljnem sistemu

Predpogoj:
- Napadalec lahko izvede določene programe na ciljnem sistemu

Napad:
1. Najdi program na ciljnem sistemu, ki je podvržen ranljivosti s prekoračitvijo sklada
2. Izdelaj programsko kodo, ki izvede zlonamerno operacijo, če se izvaja s pravicami programa
3. Izdelaj vhodno vrednost, ki bo vsilila kodo v naslovni prostor programa
4. Izvedi klic programa na način, da skoči na naslov, kjer se nahaja zlonamerna programska koda

Indikator:
- Ciljni sistem izvede zlonamerno operacijo. Nepooblaščeni uporabnik postane privilegiran.

Napad s prekoračitvijo sklada je le eden izmed načinov, kako lahko napadalec izkoristi zaupanje programa v vhodne podatke, ki jih pošlje uporabnik. Je primer večje družine z imenom »*input validation attacks*«. Takšnih napadov se le stežka ubranimo, ker so posledica programerskih napak. Najboljša zaščita je redno prenašanje popravkov, ki odpravljajo odkrite napake. Morda bo možno tovrstne napade preprečiti z drugačno varnostno zasnovo na nivoju operacijskih sistemov [9], ki temelji na ideji zmožnosti procesa in ne na ideji seznamov pravic dostopa (*Access Control List*). Slednja je osnova danes najbolj razširjenih UNIX in Windows operacijskih sistemov.

4.2 Profil napada in knjižnica napadov

Podobne napade lahko združimo v obliki profila, ki ga sestavlja ena ali več parametriziranih predlog. Profile prirejamo tako različnim tipom napadalcev, virom in znanju, ki ga imajo, ali pa različnim značilnostim sistema, ki ga varujemo.
Ker ta postopek uporablja določeno mero abstrakcije je potrebno definirati sledeče robne pogoje:
- skupen referenčni model, ki opiše okolje za katerega lahko profil uporabimo
- seznam spremenljivk, ki jih lahko spreminjamo v predlogi
- seznam predlog napadov
- slovar izrazov

Znotraj tako definiranega okolja je v nadaljevanju predstavljena predloga napada s prekoračitvijo sklada.

Primer 6: Profil napada

Primeri spremenljivk so: združba, uporabnik, intranet, računalniški sistem, požarni zid in

napadalec.

Cilj:

- Izkoristi ranljivost s prekoračitvijo sklada za izvedbo *zlonamerne operacije* na ciljnem *sistemu*

Predpogoj:

- *Napadalec* lahko izvede določene programe na ciljnem *sistemu*

Napad:

1. Najdi program na ciljnem *sistemu*, ki je podvržen ranljivosti s prekoračitvijo sklada
2. Izdelaj programsko kodo, ki izvede *zlonamerno operacijo*, če se izvaja s pravicami programa
3. Izdelaj vhodno vrednost, ki bo vsilila kodo v naslovni prostor programa
4. Izvedi klic programa na način, da skoči na naslov, kjer se nahaja zlonamerna programska koda

Indikator:

- Ciljni *sistem* izvede *zlonamerno operacijo*. Nepooblaščeni uporabnik postane privilegiran.

Medtem ko je tekst predloge napada nespremenjen, so z ležečimi črkami označene spremenljivke, ki so definirane v profilu. Vseh spremenljivk ni potrebno definirati v profilu napada, kot je to prikazano s spremenljivko »zlonamerna operacija«. Podčrtan tekst je definiran v slovarčku izrazov:

> *Ranljivost s prekoračitvijo sklada:* Napaka v programu, ki ob izvajanju predolgih vhodnih parametrov povzroči prekoračitev sklada in s tem prepiše programsko kodo ali podatke programa, ki se nahajajo poleg sklada.

Združba, ki ustreza referenčnemu model profila, lahko uporabi vsebovane predloge napadov pri izpopolnitvi drevesa napadov. Na primer podjetje Genis profil uporabi, tako da spremenljivko *združba* zamenja z Genis, *intranet* z Genisov intranet in *požarni zid* z Genisov požarni zid. Preostale spremenljivke določimo kasneje, ko drevo napadov razgradimo bolj podrobno.

Izdelan profile dodajamo v *skupno knjižnico* in s tem širimo naše znanje.

6 Izdelava drevesa napadov

Izdelava drevesa napadov je iterativen proces, ki ga ponavljamo ob vsaki spremembi sistema in njegovega okolja. Ločimo ga v dva dela:

- ročno dodajanje in
- razširjanje z uporabo knjižnice napadov.

6.1 Ročno dodajanje

Ročno dodajanje je odvisno od naše količine znanja o sistemu, za katerega snujemo drevo napadov. Za protiutež neznanja o sistemu lahko kot enega izmed možnih napadov dodamo »*neznan napad*«, ki se bo z veliko verjetnostjo pojavil kot najbolj verjeten napad na računalniški sistem. V primeru pojavitve napada, ki sodi v kategorijo neznan, lahko z analizo končanega napada to novo znanje vgradimo v drevo napadov. S časom tako spremljamo razvoj preproste v bolj dovršeno različico napada. Kronološko spremljanje razvoja napadov nam služi pri napovedovanju bodočih napadov [5].

6.2 Uporaba knjižnice napadov

Postopek uporabe knjižnice napadov je iskanje profilov z referenčnim modelom, ki ustreza modelu naše združbe in njenega okolja. Razvijalec nato znotraj teh profilov poišče primerne predloge napadov, katerih cilj omogoča doseganje enega izmed ciljev v drevesu napadov.
S pomočjo predloge napada s prekoračitvijo sklada lahko dopolnimo drevo napadov iz primera 2 za doseganje cilja dostop do varovanih dokumentov na spletnem strežniku (1.5.2).

Primer 6: Uporaba predloge napada pri dopolnjevanju drevesa napadov

S pomočjo zamenjave spremenljivk iz profila dobimo sledeči izsek iz drevesa napadov:

1.5.2 **(AND)** Dostop do varovanih dokumentov iz privilegiranega uporabnika v operacijskem sistemu v katerem deluje spletni strežnik
- 1.5.2.1 **(AND)** Pridobi dostop do privilegiranega uporabniškega računa na spletnem strežniku
  - 1.5.2.1.1 Najdi program na Genisovem spletnem strežniku, ki je podvržen ranljivosti s prekoračitvijo sklada
  - 1.5.2.1.2 Izdelaj programsko kodo, ki ob izvajanju s pravicami spletnega strežnika, omogoči dostop do privilegiranega uporabniškega računa
  - 1.5.2.1.3 Izdelaj vhodno vrednost, ki bo vsilila kodo v naslovni prostor programa
  - 1.5.2.1.4 Izvedi klic programa na način, da skoči na naslov, kjer se nahaja zlonamerna programska koda
- 1.5.2.2 Poišči varovane dokumente na datotečnem sistemu

V praksi je priporočena uporaba iterativnega ponavljanja ročnega dodajanja in uporabe knjižnice profilov. Na ta način sistematično obdelamo in vključimo obstoječe znanje iz knjižnice ter hkrati z dobro mero osebnega znanja in pogosto intuicije odkrijemo primere napadov, ki niso zajeti v knjižnici.

6.3 Računalniška podpora

Trenutno obstoja en sam program za izdelavo dreves napadov, produkt SecurITree podjetja Amenaza [8], ki omogoča izdelavo dreves napadov v grafični obliki¸ kaj-če analize na podlagi vnesenih ocen vozlišč, shranjevanje znanja v obliki knjižnic in drugo. Izdelan je v javanskem okolju.

Drevesa napadov lahko rišemo tudi v grafičnih programih kot sta Visio in Micrografix, vendar tu nimamo možnost simulacij. Za tekstualen zapis drevesa napadov pa je dovolj že preprost urejevalnik besedil.

8 Zaključek

Spoznali smo, da je upravljanje tveganja eden izmed ključnih procesov pri doseganju računalniške varnosti. S pomočjo dreves napadov lahko znana tveganja zapišemo v formalni obliki, kar nam prinese številne prednosti.

Neposredna uporabnost dreves napadov se izkaže v primeru napadov, ki jih lahko predvidimo. Z utežmi, ki jih določimo posameznemu napadu, lahko s pomočjo drevesa napadov hitro poiščemo najšibkejši člen v zaščiti (računalniškega) sistema. Nad dobljenim modelom preprosto izvajamo kaj-če analize, s pomočjo katerih varnostne ukrepe usmerimo na področja, ki jih bo z veliko verjetnostjo izkoristil napadalec.
Ukrepi, ki jih predlagamo na podlagi drevesa napadov so jasni in transparentni ter zato zlahka opravičljivi pred vodstvenim kadrom, ki sponzorira investicije v računalniško varnost. Poleg znanja o sistemu, odločitve vključujejo tudi naše znanje o napadalcu.

Posredno se uporabnost dreves napade izkaže pri naknadni analizi neznanih napadov, ki so stalni del računalniške prakse. Z njihovim sistematičnim zapisovanjem, lahko do določene mere napovemo naslednji korak napadalcev.

Pomembna novost, ki jo prinaša formalen zapis znanja v drevesu napadov je izmenjava znanja.

Odprto ostaja še vprašanje kakšna je dejansko cena varnosti in ali lahko s pomočjo drevesa napadov podamo ta odgovor? Ker je v danem trenutku lahko neprecenljiva, je odgovor enostavno, da ceno postavlja stopnja varnosti, ki jo želimo in ne varnosti nasploh. Drevesa napadov nam pri tem samo omogočajo analizo in iskanje najbolj perečih problemov.

9 Literatura

[1] Bruce Schneier: Secrets & Lies, Wiley computer publishing, 2000
[2] Moore, Ellison, Linger: Attack Modeling for Information Security and Survivability, SEI, marec 2001, CMU/SEI-2001-TN-001
[3] Sterne F. Daniel in drugi: An intruduction to Computer Security: The NIST Handbook, National Institute of Standards and Technology, U.S. Department of Commerce
[4] Jim Yuill: How a Buffer-Overflow Attack Works (unpublished resarch paper), North Carolina State University, oktober 2000
[5] Greg Rice, James Davis: A Genealogical Approach to Analyzing Post-Mortem Denial of Service Attacks, Iowa State University, 2002
[6] Sheyner, Haines in drugi: Automated Generation and Analysis of Attack Graphs, DARPA, 2003
[7] ISO 17799, International Organization for Standards, December 2000
[8] SecurITree, Amenaza Technologies Limited, www.amenaza.com
[9] EROS: The Extremly Reliable Operating System, www.eros-os.org
[10] Levesson G. Nancy: Safeware: System Saftey and Computers, Addison-Wesley Pub Co, 1st edition, april 1995