ZUNANJI VARNOSTNI PREGLEDI INFORMACIJSKIH SISTEMOV SO DOBRA PRAKSA

Tomaž Kosar[1], Milan Gabor[2], Polona Novak Vodopivec[2]

[1] Fakulteta za elektrotehniko, računalništvo in informatiko
Smetanova 17, 2000 Maribor
e-pošta: tomaz.kosar@uni-mb.si
[2] Viris, d.o.o.
Likozarjeva ulica 12, 1000 Ljubljana
e-pošta: info@viris.si

Povzetek

V današnjem času vse več storitev opravljamo preko Interneta. Pri tem prenašamo občutljive podatke in le redko se vprašamo ali je ta prenos varen. Glede varnosti prenosa podatkov ni odvisno samo od uporabnika in njegovega sistema, ampak v veliki meri tudi od snovalcev in razvijalcev informacijskih sistemov, ki stojijo na drugi strani omrežja. Uporabniki se lahko vprašajo ali so ti informacijski sistemi (pravilno) opremljeni z varnostnimi mehanizmi. Naročnikom informacijskih sistemov so lahko v veliko pomoč zunanji pregledi varnosti, pri katerih se svetovalec varnostnega pregleda postavi v vlogo napadalca na informacijski sistem. V članku najdemo nasvete kako se lahko varnostnega pregleda lotimo sami z uporabo specializiranih orodij ter podajamo najpogostejše programske ranljivosti informacijskih sistemov. Podane so tudi lastne izkušnje pri pregledih informacijskih sistemov.

1. UVOD

Živimo v informacijski dobi kjer je vsakodnevna uporaba računalnika in Interneta nuja. Vsak dan prenašamo osebne in poslovne informacije preko spleta in se pogosto ne vprašamo kako varen je prenos naših podatkov. Zato je osveščanje uporabnikov, da lahko v večini primerov sami poskrbijo za varnost prenosa podatkov, pomembna prioriteta informacijske družbe.

Pri varnosti prenosa podatkov, pa žal ni vse odvisno od uporabnikov, ampak v precejšnji meri tudi od razvijalcev informacijskih sistemov. Pogosto naročnik v ospredje funkcionalnih zahtev želenega informacijskega sistema postavi izključno funkcionalnost sistema. Na drugi strani se izvajalci naročila prejetih zahtev strogo držijo, saj lovijo projektne roke, ki so velikokrat zastavljeni precej na tesno in ne omogočajo veliko manevrskega prostora. Razvijalci se zato trudijo zadovoljiti zgolj projektnim zahtevam, pozabljajo pa na osnovna načela dobrega programiranja in pomembnost kvalitete zapisane kode, ki jo je v tem primeru možno ponovno uporabiti ali razširiti z novo funkcionalnostjo. Šele nekje proti koncu prioritetne liste razvoja projekta, se znajde uporaba varnostnih mehanizmov v informacijskem sistemu.

Varnosti podatkov oz. informacijskih dobrin v splošnem ne gre zanemariti. Povezanost preko spleta in storitvena usmerjenost informacijskih sistemov prinašata okoliščine in razloge, zaradi katerih v podjetjih narašča potreba po varovanju podatkov. Varnost vnosa podatkov in njena obdelava sta ključna za zagotavljanje informacijske varnosti in zagotavljanje varnega poslovanja podjetja. Podjetju lahko napadalec naredi nepopravljivo škodo, ki se meri v motenem delovanju podjetja, izgubi ugleda in s tem tudi izgubi strank. Zavedanje, da je varnost informacijskih sistemov pomembna, je ključnega pomena za podjetje.

Kot smo zapisali je pogost vzrok za zmanjšano uporabo varnostnih mehanizmov »slaba komunikacija« med naročnikom in izvajalcem razvoja informacijskega sistema. K temu lahko dodamo še neizkušenost razvijalcev, nepoznavanje tehnologij in slabo testiranje. Slednjih vzrokov ni možno odpraviti, tudi če bi zahteve naročnika zajemale potrebne varnostne vidike v želenem informacijskem sistemu. K temu lahko dodamo tudi to, da naročnik pogosto nima dovolj izkušenj, da bi lahko testiral oz. preveril ali je izdelana aplikacija tudi varna.

Zgornje ugotovitve napeljuje k temu, kadar je varnostne zahteve informacijskih sistemov potrebno preveriti, da je smiselno k sodelovanju povabiti neodvisnega presojevalca, strokovnjaka iz področja varnosti informacijskih sistemov. Svetovalec na področju varnosti lahko tako opozori naročnika na najdene pomanjkljivosti, razvijalcu pa svetuje, kako odpraviti najbolj pereče varnostne težave informacijskega sistema.

Podjetje Viris d.o.o. se z varnostnimi pregledi informacijskih sistemov ukvarja že vrsto let. V tem članku bomo najprej predstavili področje varnostnih pregledov – opisali najpogostejše varnostne pomanjkljivosti informacijskih sistemov, navedli specializirana orodja, ki nam pomagajo pri avtomatiziranem iskanju varnostnih lukenj in nakazali kako se lahko varnostnega pregleda lotimo sami. V nadaljevanju bomo predstavili izkušnje iz opravljenih pregledov in prikazali statistiko iz opravljenih pregledov najpogosteje najdenih pomanjkljivosti informacijskih sistemov.

2. VRSTE NAPADOV

Sans Institut [1] je izdal zanimivo študijo o ranljivostih uporabnikov. Na strani uporabnika največjo grožnjo varnosti in zasebnosti predstavljajo spletni brskalniki, nato pisarniška programska oprema, programi elektronske pošte in predvajalniki medijev. Na prvem mestu torej najdemo spletne brskalnike, kar ni presenetljivo. Ravno zaradi tega, je pomembno, da uporabniki nameščajo varnostne popravke, ki nam jih pošiljajo proizvajalci spletnih brskalnikov.

Med tem ko imamo vpliv na izbiro uporabljene programske opreme na lastnem sistemu, pa vpliva na strežnik, s katerim komuniciramo preko Interneta, praktično nimamo. Ista študija kaže, da na strani strežnika največjo nevarnost predstavljajo spletne aplikacije, sledijo storitve operacijskega sistema Windows, nato storitve operacijskega sistema Unix, itd. Spletni informacijski sistemi, kot so CMS (angl. Content Management Systems), portali, forumi, spletne trgovine, itd. uporabljajo zasebniki na eni, kot tudi podjetja na drugi strani. V glavnem se podjetja odločajo za nakup rešitev, ki obstajajo na tržišču. V tem primeru je tveganje za varnostne napade večja. Po podatkih javno dostopne podatkovne baze za ranljivost spletnih aplikacij @RISK [2], imajo uveljavljene aplikacije lahko dnevno tudi milijon nedovoljenih poizkusov dostopov. Vendar pa se v primeru uveljavljenih informacijskih sistemov dnevno pregledujejo napadi in izdajajo popravki. Torej, za te rešitve velja, da so veliko bolj oblegane s strani napadalcev kot tudi zaščitene v primerjavi z informacijskimi sistemi, ki jih podjetja razvijajo sama in so prikrojena njihovim potrebam. Takšne individualne rešitve so le redko ustrezno pregledane z vidika varnosti, kaj šele da bi se dnevno posvečali pregledom

morebitnih varnostnih lukenj. Prav zato je potrebno varnosti novega informacijskega sistema posvetiti posebno pozornost preden ta zaživi v produkcijskem okolju. Kasnejši pregledi varnosti so ponavadi izvedeni v primeru uspešnega napada na informacijski sistem. Takrat je škoda že povzročena, pregled varnosti pa v tem primeru služi le za gašenje požara.

3. VARNOSTNI PREGLEDI INFORMACIJSKIH SISTEMOV

Napadalci na različne načine poizkušajo z vdori v informacijske sisteme. Najpogostejši način je pritisk na najšibkejši člen verige, tj. človek. V tem primeru se napadalec osredotoči na napake in analizo uporabnika (npr. šibkost uporabniških računov). Običajno tega dela varnostnega pregleda informacijskega sistema v podjetju Viris ne opravljamo, kljub temu pa preizkusimo ali so administratorji informacijskega sistema odstranili/spremenili privzete uporabniške račune (npr. uporabnik »admin« z geslom »admin«, analogno za uporabnika »user«, ipd). Čeprav se zadnje zdi skorajda neverjetno, se po naših izkušnjah takšni uporabniški računi prepogosto puščajo v portalih in podatkovnih bazah.

Drugo področje varnostnih pregledov informacijskih sistemov predstavljajo analize uporabljene tehnologije. Lahko bi rekli, da je tehnologija najmočnejši člen verige informacijskih sistemov, vendar pa je resnica velikokrat drugačna.

Izogibanje in neprimerna raba varnostnih mehanizmov v informacijskih sistemih pušča veliko manevrskega prostora napadalcem. V varnostnih pregledih informacijskih sistemov naredimo prav to – osredotočimo se na uporabo tehnologije ter poizkušamo izvesti simulacijo napada z znanimi oblikami ranljivosti tehnologij in sistemov.

Običajna praksa v podjetju Viris je, da napade vršimo v dveh korakih:

- uporaba specializirane programske opreme in
- napad s surovo močjo.

Običajno izvedemo še tretji korak pregled programske kode in ponovni napad s surovo močjo. Ta korak izvršimo le v primeru, kadar imamo dostop do programske kode. V osnovi pa pomeni, da iščemo v kodi pomanjkljivosti (npr. slabo obdelane izjeme, prekoračitve vmesnika) in nato ponovno na informacijski sistem izvršimo napad.

V nadaljevanju podajamo orodja, ki smo jih pri svojem delu že uporabili in smo bili z rezultati uporabe zadovoljni.

3.1 ORODJA ZA VARNOSTNE ANALIZE

Pri varnostnih pregledih informacijskih sistemov si lahko pomagamo z orodji, ki avtomatizirajo pregled varnosti informacijskega sistema (angl. Security Analyzers). Glede na način opravljene analize delimo varnostna orodja v dve skupini [3]:

- statična analiza kode (angl. Static Code Analysis) in
- dinamična analiza kode (angl. Dynamic Code Analysis).

Seveda, statični analizatorji kode omogočajo precej več kot samo iskanje varnostnih lukenj. Za naše potrebe pa uporabljamo samo slednje [4]. V primeru statične analize gre za preverjanje programske kode brez zagona, v primeru dinamične analize se aplikacija najprej izvede šele nato pričnemo z iskanjem varnostnih lukenj.

Na spletu lahko najdemo tudi drugačno delitev orodij – glede na vrsto kode, ki jo orodja pregledajo: pregledovalnik izvorne kode (angl. Source Code Scanner), pregledovalnik vmesne

kode (angl. Byte Code Scanner) in pregledovalnik binarne kode (angl. Binary Code Scanner). Prvi dve vrsti (pregledovalnik izvorne in vmesne kode) v osnovi opravljata statično analizo, zato bodo orodja iz teh dveh skupin v nadaljevanju predstavljena skupaj.

3.1.1 STATIČNA ANALIZA

Kot smo zapisali, je statična analiza informacijskega sistema opravljena brez zagona. Opravi jo program, ki temelji na gramatikah (angl. Grammar Based System). Izpopolnjenost orodij za statično analizo kode se razlikuje glede na to, ali orodja pregledajo posamezne stavke in deklaracije ali vključujejo tudi analizo celotne izvorne kode. Komercialna uporaba statične analize kode je ravno na področju odkrivanja in lociranje varnostno kritičnih odsekov programske kode.

V tabeli 1 podajamo orodja, ki temeljijo na statični analizi.

Orodje	Jezik	Prost dostop	Namen
Pixy [5]	PHP	Da	Pixy je orodje, ki ga poženemo iz ukazne vrstice in analizira PHP program. Pixy izdela poročilo, v katerem so podane možne ranljive točke aplikacije z dodatnim opisom ranljivosti.
SWAAT [6]	Java, JSP, PHP	Da	Orodje SWAAT je odprto kodno orodje, ki ne vsebuje jezikovno kontekstnega znanja, ki je potrebno, da se zagotovo najdejo varnostne napake v določenem jeziku. Vsi potencialno nevarni deli kode se vrnejo v obliki poročila z referencami.
Ounce [7]	C, C++, Java, JSP, .NET	Ne	Orodje podjetja Ounce Labs vsebuje podporo za statično analizo širokega nabora programskih jezikov. Orodje poleg iskanja varnostnih pomanjkljivosti, nudi še iskanje pomanjkljivosti v zasnovi aplikacije in ponuja rešitve za znane težave.
MS FxCop [8]	.NET	Da	FxCop je orodje, ki analizira skladnost .NET programa z »NET Framework Design Guidelines«. Uporablja razpoznavanje kode na nivoju MSIL (pregledovalnik vmesne kode) ter jo preveri z več kot 200 različnimi pravili.
SSW Code Auditor [9]	.NET	Ne	SSW Code Auditor je močno orodje, ki temelji na statični analizi programske kode. Vgrajena pravila so namenjena za projekte v .NET jezikih (C#, VB.NET), tako za namizne (Win Forms) kot za spletne aplikacije (ASP.NET). Fleksibilnost orodja SSW Code Auditor je v tem, da omogoča zapis novih pravil ranljivosti programske kode.

Tabela 1. Statični analizatorji programske kode

Večina komercialnih produktov ponuja tudi preizkusno verzijo pregledovalnikov z omejenim naborom funkcionalnosti. Tako lahko z orodjem SSW Code Auditor izpišemo do 500 opozoril iz programske kode na omejenem naboru pravil.

3.1.2 DINAMIČNA ANALIZA

Dinamična analiza informacijskega sistema se naredi z izvajanjem programske kode na pravem ali virtualnem procesorju. Za učinkovito dinamično analizo je analizator potrebno zagnati z zadostnim številom testnih podatkov. Testni podatki se lahko generirajo naključno v samem analizatorju ali s pomočjo podanih vzorcev, ki jih v nekaterih orodjih lahko uporabnik dopolni.

Orodje	Jezik	Prost dostop	Namen
SecurityReview [10]	C, C++, C#, Java	Ne	Avtomatizirano orodje za statično in dinamično analizo. SecurityReview je storitev, ki jo ponuja podjetje Veracode.
DevInspect [11]	C#, Java, XML, SOAP, JavaScript	Ne	DevInspect je orodje podjetja HP, ki omogoča avtomatizirano varnostno preizkušanje in popravljanje programske kode med razvojem programske opreme. Omogoča integracijo v razvojne platforme.
AppScan [12]	Web services, JavaScript, Java	Ne	Rational AppScan omogoča iskanje ranljivosti spletnih aplikacij z uporabo statične in dinamične analize.
Acunetix WVS [13]	CGI, PHP, ASP.NET	Ne	Acunetix Web Vulnerability Scanner (WVS) je avtomatizirano orodje za testiranje varnosti spletnih aplikacij.

Tabela 2. Dinamični analizatorji programske kode

Vsa orodja iz tabele 2 so komercialna in omogočajo tako statično kot dinamično analizo.

V okviru varnostnih pregledov, imamo največ izkušenj z orodjem WVS, ki je enostavno orodje, kateremu je potrebno podati samo spletni naslov testirane aplikacije in orodje poskrbi za ostalo. Rezultat pregleda spletne strani se vrne v obliki strukture spletne strani, kjer je vsaka datoteka opremljena z morebitnimi pomanjkljivostmi.

3.2. NAPADI S SUROVO SILO

Uporaba orodij za statično in dinamično analizo programske kode je enostavna in jo lahko preizkusijo tudi uporabniki brez predznanja iz razvoja programske opreme. Vse kar uporabnik potrebuje za razumevanje rezultatov, je poznavanje ranljivosti spletnih aplikacij.

Bolj poglobljeno poznavanje spletnih pomanjkljivosti je potrebno pri »ročnem« napadu na spletno aplikacijo. V tem primeru gre za testiranje črne skrinjice (angl. black box testing), kjer pridemo do naročnika in za njegovim računalnikom preizkusimo informacijski sistem z znanimi ranljivostmi.

Najpogostejše ranljivosti in njihov kratek opis navajamo spodaj:

- Vrivanje SQL (angl. SQL Injection)

  Pomanjkljivost vrivanja SQL je zelo pogosta v spletnih aplikacijah. Je tehnika s katero napadalec poizkuša vriniti svojo kodo z namenom pridobitve podatkov iz SQL podatkovne baze. Vrivanje se pripeti takrat, kadar se uporabniški podatki pošljejo interpreterju skupaj z zlonamernim ukazom ali poizvedbo. Napadalčev sovražni ukaz ukani interpreter, ta izvrši nezaželene poizvedbe in tako vrne ali spremeni podatke. To ranljivost poznajo vse spletne tehnologije.

- Napad XSS (angl. Cross Site Scripting)

  Napad XSS pomeni prikaz uporabniških podatkov v spletnem brskalniku brez preverjanja vnosa. XSS omogoča napadalcem zagon skript v spletnem brskalniku uporabnika. Na ta način lahko napadalec ukrade podatke iz uporabniške seje, pokvari vsebino prikaze spletne strani, zažene škodljive programe, itd.

- Prekoračitev vmesnika (angl. Buffer Overflow)

  V večini spletnih jezikov ni možnosti, da bi do pomnilnika dostopali direktno preko kazalcev, vendar to še ne pomeni, da programska koda ni ranljiva na omenjen napad. V programskem jeziku je potrebno paziti na prekoračitve vmesnika (npr. pisanje čez velikost dinamičnega polja). Z vidika sistema pa lahko najdemo napako, ki omogoča prekoračitev vmesnika v storitvi (npr. RPC – Remote Procedure Call). V tem primeru prekoračitev pomnilnika omogoči napadalcu, da izvršiti kodo pod sistemskimi privilegiji.

- Upravljanje napak (angl. Error Handling)

  Če lahko napadalec vidi napake programske kode ali spletnega strežnika v »surovem« stanju, lahko iz vrnjenih informacij izve veliko koristnih podatkov o strukturi spletne strani, uporabljeni tehnologiji, nastavitvah strežnika, ipd. Vračanje informacij o napakah lahko izkušenim napadalcem prinesejo koristne informacije za vdor v informacijski sistem. Zato je na nivoju spletnega strežnika potrebno poskrbeti za ustrezne varnostne nastavitve.

- Vstavljanje datotek (angl. File Inclusion)

  Programska koda, ki je ranljiva na vključevanje datotek omogoča napadalcem vključitev sovražne kode in podatkov, ki lahko privedejo do uničujočega napada (na strežniški strani). Ranljivost je zelo pogosta v PHP aplikacijah.

Na spletni strani [14] lahko najdemo ostale zelo pogoste varnostne luknje med katere sodijo še: kraja seje (angl. Broken Session), nepreverjeni parametri (angl. Unvalidated Parameters), ukazno vrivanje (angl. Command Injection Flows), napačna uporaba kriptografije (angl. Insecure Use of Cryptography), nepravilne nastavitve spletnega strežnika (angl. Web Server Misconfiguration), nevarovana povezava (angl. Insecure Communication), napaka pri omejevanju dostopa do URL (angl. Failure to Restrict URL Access), itd. Obstaja več sto evidentiranih ranljivosti spletnih aplikacij. Večina jih je pogojena z uporabljeno tehnologijo.

Kako preizkusiti posamezne ranljivosti, ki so zapisane zgoraj, je odvisno od napadalčeve iznajdljivosti in izkušenosti v posamezni tehnologiji. V tem članku se v podrobnosti napadov za posamezen tip ranljivosti in tehnologije ne bomo spuščali. Zainteresirani bralec si lahko več o tehnikah vdora prebere na spletu.

4. PREGLEDI KVALITETE INFORMACIJSKIH SISTEMOV

Velika večina podjetij ugotavlja, da je razvoj lastne programske opreme drag. Pri sodelovanju z zunanjimi izvajalci razvoja programske opreme pa naročnik težko oceni ali je programska oprema kvalitetna, če tega znanja nima. Še težje je oceniti ali je izdelek tudi varen.

Iz preteklih izkušenj podjetja, lahko povemo, da naročniki varnostnih pregledov informacijskih sistemov pogosto želijo izvedeti kako je s kvaliteto programske kode informacijskega sistema.

Nekatera orodja iz tabele 1 in 2 omogočajo tudi kvalitetne preglede programske kode (SSW Code Auditor, SWAAT, itd). Nekaj specializiranih orodij za pregled kakovosti programske kode je naštetih tudi v tabeli 3. Iz nje je razvidno, da je nekatera orodja možno namestiti kot vstavke v razvojna okolja.

Orodje	Jezik	Prost dostop	Namen
Agent Smith [15]	.NET	Da	Visual Studio dodatek (plugin) za preverjanje kvalitete programske kode v jeziku C#.
Code Style Enforcer [16]	.NET	Da	Visual Studio 2005/2008 dodatek za preverjanje kvalitete programske kode z uveljavljenim standardom programiranja v jeziku C#. Slednji je v orodju zamenljiv oz. uporabnik ga lahko priredi lastnim prepričanjem.
StyleCop [17]	.NET	Da	StyleCop je statični analizator C#, ki ga prav tako lahko integriramo v Visual Studio.
Resharper [18]	.NET	Da	Resharper je eden tistih dodatkov v Visual Studio, brez katerega razvijalec ne more. Resharper poleg preverjanja kvalitete kode, nudi pomoč za znane napake, vsebuje več kot 30 metod za preoblikovanje programske kode, napredna orodja za testiranje enot, omogoča avtomatično generiranje programske kode iz šablon, itd.
Checkstyle [19]	Java	Da	Orodje za preverjanje kode s standardom programiranja v jeziku Java. Za orodje Checkstyle obstaja veliko število dodatkov za različna razvojna orodja: Eclipse, NetBeans, IntelliJ IDEA, BlueJ, itd.
PMD [20]	Java	Da	Tudi za PMD obstaja veliko število dodatkov za različna orodja: JDeveloper, JEdit, JBuilder, itd.

Tabela 3. Orodja za preverjanje kvalitete programske kode

Orodja za pregled kvalitete programske opreme izvajajo statično analizo kode (podobno kot smo to opisali v poglavju 3.1.1). Orodja vsebujejo bazo pravil, ki implementirajo standarde programiranja (angl. Coding Standards) za posamezen programski jezik in običajno preverjajo:

- poimenovanja v programski kodi (poimenovanja imenskih prostorov, razredov, instančnih in lokalnih spremenljivk),

- oblika programske kode,
- odkrivanje praznih odsekov kode,
- preverjanje pravilnosti vhodnih parametrov metod,
- vračanje metod,
- inicializacija spremenljivk pred njihovo uporabo,
- sproščanje zaseženega pomnilnika,
- sproščanje virov ob napakah,
- upoštevanje sodobnih trendov dobrega programiranja,
- preverjanje uporabniških sporočil,
- obravnava napak in izjem, itd.

Vpogled v izvorno kodo daje varnostnemu pregledu dodatno dimenzijo, saj lahko izvedemo sklepanje o varnosti informacijskega sistema iz kode. Npr. najdena neprimerna obdelava izjeme lahko povzroči padec aplikacije. V tem primeru bo spletni strežnik vrnil uporabniku povratne informacije in svetovalec lahko preveri ali so te informacije napadalcu koristne.

Pregledi kvalitete programske kode pa velikokrat dajejo pozitivne stranske učinke v obliki funkcionalnih napak, tako lahko kvalitetni pregled programske kode velikokrat odpravi pomanjkljivosti slabega ali neobstoječega prevzemnega testiranja informacijskega sistema. Na tem področju smo imeli zelo zanimive izkušnje, npr. velikokrat smo naleteli na funkcionalne sklope, ki so bili ponovno uporabljeni iz drugih projektov, pozabilo pa se je spremeniti zunanji izgled teh modulov, ipd.

Zelo pogosto se je pregled kvalitete programske kode pričel z ugotovitvijo, da za informacijski sistem ne obstaja projektna, tehniška, namestitvena dokumentacija, itd. Prav tako smo velikokrat pogrešali uporabniška navodila. V tem primeru smo naročnika in izvajalca pozvali k zapisu manjkajoče dokumentacije, ki je nujno potrebna za vzdrževanje in nadaljnji razvoj aplikacije. Velikokrat je bila dokumentacija tudi pomanjkljiva in informacije nesamozadostne, uporabniška navodila pa brez (opisa) ekranskih slik, manjkajoče razlage funkcionalnosti po vlogah uporabnikov, ipd.

5. IZKUŠNJE IZ PREGLEDOV

Naše izkušnje pregledov so zelo raznolike, saj je velikokrat naročnik tisti, ki poda svoje zahteve glede na kritičnost aplikacij ali informacijskih sistemov. Opravljeni pregledi segajo od spletnih portalov, raznovrstnih aplikacij za komunikacijo z obiskovalci, pa vse do spletnih aplikacij za trgovanje in avkcije ter spletne aplikacije elektronskega bančništva. Pogosto smo opravili tudi preglede internih aplikacij, ki služijo raznim službam znotraj podjetij in ustanov. Ker v primeru pregledov aplikacij veljajo določila o nerazkritju informacij, bomo v nadaljevanju predstavili samo nekaj najbolj splošnih primerov s katerimi smo se srečali v zadnjih dveh letih.

Celoviti pregledi aplikacij

Naš pregled obsega celoten življenjski cikel razvoja aplikacije, to pomeni od začetnih funkcionalnih specifikacij, dokumentacije, izvorne kode, testov, prevajanja same kode, pa do preverjanja delovanja same aplikacije. S tem pregledom, lahko naročnik dobi celotno sliko

kakovosti aplikacije. V večini primerov so te aplikacije razvite s pomočjo zunanjih izvajalcev, saj naročniki nimajo dovolj razvijalcev znotraj lastnega podjetja. Glede na naša priporočila potem naročnik prevzame aplikacijo ali pa izvajalcu naroči, da popravi določene funkcionalnosti, ki še manjkajo pri izdelani aplikaciji. Pri takšnih pregledih dobi naročnik jasno sliko o kakovosti kode, aplikacije in še drugo mnenje, ki je neodvisno od samega razvojnega procesa, ki se je uporabljal pri razvoju in mu lahko koristi pri končnem prevzemu aplikacije. Takšna praksa je lahko zelo pomembna v primerih, ko naročnik postane tudi lastnik kode in jo po koncu razvoja prevzame v lastno vzdrževanje. V primeru da bi bila aplikacija slabo napisana ali imela večje pomanjkljivosti si s tem olajša vzdrževanje izvorne kode v lastni hiši. Pri teh pregledih aplikacij lahko trdimo, da se aplikacije ne razvijajo v skladu s principi razvoja programske opreme, ampak na hitro in iz prototipa direktno v aplikacijo, kar velikokrat vodi v dejstvo, da je končni produkt pomanjkljiv.
V večini primerov smo opazili, da manjka pomembna dokumentacija, kar lahko naročniku oteži razumevanje in vzdrževanje programske kode. Pri samem načrtovanju pa opažamo, da se velikokrat spremenijo funkcionalne zahteve, saj naročnik in izvajalec ne definirata natančno vseh podrobnosti in se kasneje odkrijejo različna pojmovanja funkcionalnosti ali zahtev, kar lahko vodi do nesporazumov pri končnem izdelku.

Pregledi aplikacij

Pri tej vrsti pregledov se postavimo v vlogo spletnih uporabnikov in poskušamo z uporabo različnih načinov preizkušanja aplikacij preveriti varnost in tudi uporabnost aplikacij. V tem primeru ne pregledamo programske kode in njene dokumentacije kot v prejšnjem tipu pregleda, ampak se osredotočimo samo na aplikacijo v času izvajanja. V večini primerov so to spletni portali, spletne aplikacije, trgovalne aplikacije ali pa kašne druge namenske aplikacije, ki so lahko tudi samo intranetne.
Pri tovrstnih pregledih z različnimi tipi napadov preverjamo odzive aplikacije in poskušamo pripraviti aplikacijo do tega, da dobimo bodisi nadzor nad aplikacijo kot administratorji ali pa poskušamo spremeniti npr. podatke na spletnih straneh, kar bi lahko nekdo izkoristil kot potencialni napad na spletne strani. Pri teh napadih uporabljamo kombinacijo avtomatskih in ročnih napadov, veliko težo pa ima tudi kreativnost in iznajdljivost.
Pri teh pregledih običajno odkrijemo več različnih tipov ranljivosti (XSS, SQL injection, File inclusion, Input validation, itd., glej poglavje 3.2), ki so posledica tega, da se posveti premalo pozornosti varnosti v aplikacijah. Velikokrat pa naletimo tudi na datoteke, ki so na strežnikih puščene bodisi po naključju ali pa zaradi varnostnih kopij in lahko pomenijo potencialno nevarnost, saj lahko vsebujejo kritične podatke. Opažamo tudi, da se informatiki velikokrat ne zavedajo nevarnosti, ki jo lahko že majhna varnostna pomanjkljivost v spletnem portalu povzroči. Kot primer naj navedemo, da nam je uspelo v enem pregledu zaradi vrivanja SQL pridobiti celotno podatkovno bazo naročnika.
V sliki 1 smo povzeli pomanjkljivosti, ki smo jih odkrili v opravljenih zadnjih 15 pregledih.

Pregledi sistemov

Pri pregledih sistemov naročniku ponudimo dva tipa pregledov:

- Zunanji pregled: postavimo se v vlogo napadalca in se poskušamo dokopati do informacijskega sistema naročnika;
- Notranji pregled: kjer simuliramo uporabnika in vidimo do katerih notranjih virov se lahko dokopljemo.

Slika 1. Najpogosteje najdene pomanjkljivosti pri pregledih informacijskih sistemov (N= 15)

Pri zunanjih pregledih nismo orientirani samo na aplikacije, ampak na celoten sistem, vključno s socialnim inženiringom. Tako se moramo postaviti v vlogo potencialnega napadalca, zbrati čim več informacij o samem sistemu, zaposlenih, potencialnih šibkih točkah in potem doseči cilje, ki smo jih določili z naročnikom.
Pri teh napadih so naše tarče:

- Onemogočiti spletne strani;
- Onemogočiti poštni sistem ali druge komunikacijske kanale;
- Prevzeti nadzor nad omrežjem naročnika;
- Pridobiti ključne datoteke z datotečnih strežnikov, itd.

Pri notranjem tipu pregleda pa simuliramo na primer zunanjega izvajalca, ki se občasno priklaplja v omrežje naročnika in preverimo do katerih podatkov se lahko dokoplje in kakšno potencialno škodo bi lahko povzročil. Običajno odkrijemo še sisteme, ki jim manjkajo kritični popravki, kar olajša delo potencialnim napadalcem. Pri tej vrsti pregleda sodelujemo z upravo naročnika in sistemskimi administratorji, lahko pa izjemoma tudi samo z upravo in tako hkrati preverimo odzivnost sistemskih administratorjev glede na različne tipe nenavadnih dogodkov v omrežjih in aplikacijah, ki jih povzročimo s pregledom ali potencialnim napadom.
Poročilo, ki ga pripravimo naročniku, vsebuje celoten časovni potek pregleda, odkrite pomanjkljivosti in tudi priporočila kako te pomanjkljivosti odpraviti.

6. PREDLOGI ZA RAZVIJALCE IN SISTEMSKE ADMINISTRATORJE

Preverjanje vnosa

V večini pregledov je bilo ugotovljeno, da se premalo pozornosti posveča preverjanju pravilnosti vnosnih podatkov. V teh primerih ugotavljamo, da se razvijalci zanašajo na to, da uporabniki ne pomenijo potencialne nevarnosti za informacijski sistem in se smatrajo njihovi vnosi kot zanesljivi. Konkretne izkušnje pri pregledih kažejo, da z malo izvirnosti lahko pridemo z napačnimi vnosi do zanimivega obnašanja aplikacij. Zato je potrebno pri vsakem vnosu jasno definirati vhod in ga tudi preveriti za različnimi poskusi vnosa nepravilnih vrednosti. Zavedati se je potrebno, da se vnos vrši na samem brskalniku in se lahko z različnimi tehnikami spremeni.

Odpiranje aplikacij navzven

Že pri sami zasnovi interne aplikacije je potrebno imeti v mislih, da lahko ta aplikacija postane internetna ali pa se jo samo delno odpre navzven določenim uporabnikov. Če že pri sami zasnovi osnovni principi varnosti niso bili upoštevani, lahko postanejo aplikacije tarča napadalcev in velikokrat se zgodi, da takšne aplikacije pomenijo lahek vstop v informacijski sistem.

Izdaja nepotrebnih informacij

Pri nepravilni konfiguraciji izvajalnega okolja, nepravilnem lovljenju izjem ali neupoštevanju drugih dobrih praks se lahko pripeti, da potencialnemu napadalcu že sama spletna stran nudi precej informacij o okolju, aplikacijah, aplikacijskih in spletnih strežnikih, kar mu olajša delo. Pogosto je tudi opaziti, da dokumenti na spletnih straneh vsebujejo veliko nepotrebnih informacij kot na primer uporabniška imena uporabnikov v domenah, poti do tiskalnikov, verzije operacijskih sistemov in še kakšno nepotrebno informacijo.

Zato svetujemo, da ste pozorni na:

- Odstranitev vseh podatkov iz dokumentov, ki lahko napadalcu omogočijo lažji napad;
- Pravilno opravljeno konfiguracijo okolja, da ne izdaja verzij in ne javlja napak, ki vsebujejo pomembne podatke o okolju;
- Pravilno opravljeno konfiguracijo aplikacijskih strežnikov;
- Preverite podatke še preden se spletna stran odpre na Internetu.

7. ZAKLJUČEK

V okviru sodelovanja z različnimi podjetji smo ugotovili, da imajo podjetja težave pri varovanju na različnih nivojih: omrežja, strežniki, komunikacije in nenazadnje tudi informacijski sistemi. Slednje je tudi eno izmed področij za katero se je podjetje Viris d.o.o. specializiralo. Izvajamo etične napade na informacijske sisteme za znane naročnike.

V članku smo opisali postopek varnostnih pregledov informacijskih sistemov ter ranljivosti, ki jih s pregledi iščemo. Osredotočili smo se na predstavitev avtomatiziranih orodij, njihove prednosti in razlike. Podali smo tudi lastne izkušnje iz pregledov.

Želimo poudariti, da je pred razvojem informacijskih sistemov varnost velikokrat samoumevna tako naročniku kot izvajalcu razvoja. Razvoj varnosti informacijskega sistema mora obsegati vse faze življenjskega cikla – informacijski sistem je varen, če je varen celoten SDLC (angl. Software Development LifeCycle). Varnost ni enkraten dogodek. Zaradi različnih vplivov, se pogosto pozornost v kakšni fazi premalo obravnava. Zaradi tega so kasnejši zunanjimi pregledi dobra praksa s katero preverimo varnost informacijskega sistema.

LITERATURA

1. http://www.sans.org/top20/
   Sans Institut, Top 20 Security Risks.

2. http://www.sans.org/newsletters/risk/
   @Risk, podatkovna baza za evidentiranje spletnih ranljivosti.

3. LIVSHITS, Benjamin, *Improving Software Security with Precise Static and Runtime Analysis*, doktorska disertacija, Stanford University, 2005.

4. FONSECA, J., VIEIRA, M., MADEIRA, H. Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks, številka 17-19, december 2007, str.365 – 372.

5. http://pixybox.seclab.tuwien.ac.at/pixy/index.php
   Pixy, statični analizator PHP kode.

6. http://www.securitycompass.com/inner_swaat.shtml
   SWAAT, statični analizator programske kode.

7. http://www.ouncelabs.com/products/
   Ounce, statični analizator programske kode.

8. http://code.msdn.microsoft.com/codeanalysis/Release/ProjectReleases.aspx?ReleaseId=553
   Microsoft FxCop, statični analizator .NET kode.

9. http://www.ssw.com.au/SSW/CodeAuditor/
SSW Code Auditor, plačjivi statični analizator .NET kode.

10. http://www.veracode.com/solutions/sdlc-security-assessment.html
SecurityReview, dinamični analizator.

11. https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-201-200^9564_4000_100__
DevInspect, dinamični analizator.

12. http://www-01.ibm.com/software/rational/offerings/websecurity/
AppScan, dinamični analizator.

13. http://www.acunetix.com/
Acunetix WVS, dianmični analizator programske kode.

14. http://www.owasp.org/index.php/Top_10_2007
OWASP Top 10 2007, 10 najpogostejših ranljivosti spletnih aplikacij po podatkih združenja OWASP (Open Web Application Security Project).

15. http://www.agentsmithplugin.com/
Agent Smith, pregledovalnik kvalitete programske kode.

16. *http://joel.fjorden.se/static.php?page=CodeStyleEnforcer*
Code Style Enforcer, pregledovalnik kvalitete C# kode.

17. http://code.msdn.microsoft.com/Release/ProjectReleases.aspx?ProjectName=sourceanalysis&ReleaseId=1425
StyleCop, pregledovalnik kvalitete C# kode.

18. http://www.jetbrains.com/resharper/
Resharper, pregledovalnik kvalitete C# kode.

19. http://checkstyle.sourceforge.net/
Checkstyle, orodje za statično analizo Java kode.

20. http://pmd.sourceforge.net/
PMD, orodje za statično analizo Java kode.