Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Študijný odbor: Počítačové Systémy a Siete

Analýza problému, špecifikácia požiadaviek riešenia spolu s hrubým návrhom

Penetračné testovanie

Tímový projekt

Vedúci projektu: Doc. Ing. Ladislav Hudec, CSc.; Ing. Adrian Bagala

Borlok Ján	jan.borlok@allenovery.com
Krištof Ján	deity@pobox.sk
Kubík Matej	kubik@zuikaku.org
Lenz Roman	nolimits@pobox.sk
Mateja Miroslav	hooki@r3.roburnet.sk

Obsah

1 Úvod

Bezpečnosť počítačov a počítačových sietí je dnes často skloňovaným pojmom. Je to celkom pochopiteľné, pretože neoprávnená osoba, ktorá prenikne do počítačovej siete spoločnosti (napríklad firmy), môže spôsobiť veľmi veľké škody, či už priame alebo nepriame. Jednou z dôležitých častí počítačovej bezpečnosti je takzvaná sieťová bezpečnosť, ktorá sa zaoberá ochranou pred útokmi prichádzajúcimi cez počítačovú sieť. Jeden z najčastejšie používaných prvkov sieťovej bezpečnosti je takzvaná bezpečnostná brána, ktorá chráni privátnu časť siete od pred útokmi s verejnej siete.

Na udržanie primeranej bezpečnosti brány pred vonkajšími útokmi je nutné jej opakované testovanie. Jedným z možných prístupov k tomuto testovaniu je takzvané penetračné testovanie, ktoré napodobňuje správanie sa útočníka pri útoku (pokuse o penetráciu) na systém.

2 Analýza problému

Penetračné testovanie je testovanie bez znalosti štruktúry siete, alebo testovaného zariadenia takzvanou metódou čiernej skrinky. Testujúci obvykle napodobňuje správanie sa útočníka snažiaceho sa preniknúť do siete, t.j. zameriava sa na viac i menej známe bezpečnostné diery v používaných produktoch, cez ktoré postupne skúša preniknúť. Nevýhodou tohoto systému je, že útočník môže mať znalosti o zraniteľnostiach, ktoré testujúci nemá a ktoré mu uľahčia prienik do systému. Výhodou je, že je možné kroky penetračného testovania zautomatizovať. Pri tomto testovaní sa obvykle testuje firewall z vonkajšej i vnútornej siete.

2.1 Počítačová bezpečnosť

Pokiaľ má byť bezpečnosť správne implementovaná, musí sa dynamicky prispôsobovať zmenám, ktoré vznikli po prvotnej implementácii bezpečnostného projektu. V ideálnom prípade by sa malo jednať o stály proces sledovania bezpečnosti, analýzy vzniknuvších incidentov, reakcií na ne a potrebnej zmeny bezpečnostných opatrení. Jednotlivé fázy bezpečnostného procesu sú:

– fáza prípravy, v ktorej sa rozhodne o zavedení bezpečnostného procesu do spoločnosti,

– fáza analýzy, kedy sa analyzujú potreby spoločnosti, určuje bezpečnostná politika a navrhujú ďalšie opatrenia týkajúce sa bezpečnosti,

– vo fáze implementácie sa opatrenia, navrhnuté vo fáze analýzy, zavádzajú do praxe,

– vo fáze prevádzky systému sa vykonáva stály dohľad nad systémom, ktorý umožňuje včas odhaliť potrebu zmeny bezpečnostného modelu alebo bezpečnostný incident,

– fáza reakcie, kedy sa analyzuje vzniknutá zmena alebo bezpečnostný incident; výsledky tejto analýzy sa prenesú ďalej do fázy analýzy.

Na správnu aplikáciu bezpečnosti treba vedieť, ktoré veci a pred akými typmi útokov treba chrániť. Útokmi ohrozené sú najmä:

1) Dáta. Pri ich zabezpečení musíme klásť dôraz na tri základné požiadavky:
   - diskrétnosť, dáta sa nesmú dostať do rúk nepovolaným osobám;
   - integrita, dáta nesmú byť neoprávneným spôsobom modifikované alebo vymazané;
   - a dostupnosť, dáta musia byť prístupné povereným užívateľom, ktorí by inak nemohli vykonávať svoju prácu.

2) Zdroje, to jest výkon počítačov a iných zariadení zapojených v sieti.

3) Povesť spoločnosti. Šikovný votrelec môže využiť systém, do ktorého prenikol, na útok na iné systémy, čo môže učiniť majiteľa trestne zodpovedným, pokiaľ nedokáže, že sa útok neudial jeho zavinením. Spoločnosť, ktorá spracováva chúlostivé údaje svojich obchodných partnerov alebo ponúka bezpečnostné riešenia, pravdepodobne po incidente zaznamená veľké poškodenie svojej povesti.

Útoky, ktorým môžu byť počítače a ich siete vystavené sa delia do troch základných kategórií:

1) Krádež informácií. Umožňuje útočníkovi získať bežne nedostupné dáta bez toho, aby musel použiť počítačovú techniku spoločnosti. Krádež informácie môže byť aktívna, kedy sa útočník aktívne pokúša informáciu získať napríklad predstieraním, že sa jedná o oprávneného užívateľa, alebo pasívna, keď sa útočník napríklad napichne na počítačovú sieť obdobne ako na telefónnu linku a čaká, kedy k nemu potrebná informácia dorazí. Krádež informácií vôbec nemusí využívať počítače – nedávna štúdia vo Veľkej Británia ukázala, že vyše 30% ľudí je ochotných prezradiť svoje heslo neznámemu človeku.

2) Odoprenie služby. Tieto útoky neposkytujú útočníkovi žiadne nové privilégiá ani informácie, naopak znemožňujú oprávneným používateľom používať techniku spôsobom, na aký sú autorizovaní. Tieto útoky môžu mať následky trvalé, teda zničenie výpočtovej techniky napríklad sabotážou alebo prírodnou katastrofou, alebo dočasné, keď je možné po ukončení útoku jeho následky odstrániť. Najväčšou nevýhodou týchto útokov je, že dokonalá obrana proti nim neexistuje. Ak totiž komunikujeme s vonkajším svetom, je tu vždy určitá možnosť zaplavenia nadmerným množstvom požiadaviek. Väčšina „náhodných“ útokov, ako sú prírodné katastrofy alebo omyly, sa radí do tejto kategórie.

3) Vniknutie je veľmi častým typom útoku, pri ktorom je cieľom útočníka využívať počítačové zdroje tak, ako to môžu autorizovaní používatelia.

2.2 Všeobecné princípy obrany

Každá dobrá bezpečnostná stratégia kombinuje viacero princípov obrany do jedného celku. Základnými princípmi, z ktorých sa tieto stratégie budujú, sú:

1) „Security through obscurity“, bezpečnosť pomocou záhadnosti. Tento princíp zvyšuje bezpečnosť tým, že o systéme takmer nikto nič nevie. Je nechválne známy tým, že mnoho užívateľov sa z neznalosti alebo snahy ušetriť uchyľuje k nemu ako

jedinému bezpečnostnému opatreniu. Bohužiaľ, k informáciám sa dá veľmi ľahko dostať a tak je tento princíp vhodný len ako doplnkové zabezpečenie k ostatným opatreniam a nie je vhodné sa naň spoliehať.

2) Minimálne privilégium diktuje, že každý komponent alebo užívateľ by mal mať len tie práva, ktoré potrebuje k svojej činnosti, ale nie viac. Tento princíp je jedným zo základov bezpečnosti. Príkladom na jeho využitie sú napríklad ACL v systéme Netware alebo VMS, ktoré umožňujú presne určiť, ktorý užívateľ bude mať aké práva na manipuláciu so súborom, alebo setuid a setgid programy pod operačným systémom UNIX, ktoré umožňujú používateľovi prideliť vyššie privilégiá počas behu programu, ale tieto privilégiá neprideliť v ostatných prípadoch. Niekedy sa však možno stretnúť s problémom nedostatočnej granularity privilégií, kedy sú potrebné práva viazané na iné, nepotrebné a nie je možné ich prideľovať zvlášť.

3) Žiadna obrana nie je nepreniknuteľná. Prienik však môže stáť útočníka toľko úsilia, že sa rozhodne, že pokračovať v útoku je preňho nevýhodné. Toto je základom stratégie hĺbkovej obrany: každý komponent je zdvojený iným, pričom útočník, ak chce úspešne vykonať útok, musí prekonať oba.

4) Škrtiaci bod je stratégia, ktorá minimalizuje miesta, z ktorých môže prísť útok. Ak sú napríklad všetky modemy, pripojenia do iných sietí a Internetu oddelené od siete spoločnosti, toto oddelenie musí prekonať každý útočník, preto je to škrtiaci bod. Tento prístup síce kladie riziko na zabezpečenie tohoto bodu, ale odbremeňuje od nutnosti sledovať iné cesty, ktorými sa útočník môže dostať dovnútra.

5) Jednou zo základných myšlienok bezpečnosti je, že reťaz je tak silná ako jej najslabší článok, preto je vhodné všetky zraniteľné miesta zabezpečovať rovnomerne. Pokiaľ totiž budú niektoré oblasti zanedbané, stanú sa slabý článok a útok tade pravdepodobne uspeje.

6) Odolnosť proti zlyhaniu je v bezpečnosti tiež dôležitá. Znamená návrh prostriedkov ochrany tak, aby ich zlyhanie nezvýšilo bezpečnostné riziko. Tento postoj napríklad diktuje aj preferenciu stratégie implicitného zákazu (čo nie je zvlášť povolené, je zakázané) pred implicitným povolením (čo nie je zvlášť zakázané, je povolené). Pokiaľ totiž nie je povolená nutná činnosť, príde sa na túto chybu veľmi rýchlo, ak sa niekto pokúsi túto činnosť vykonať. Na opačnú chybu, zabudnutie zákazu nebezpečnej služby sa veľmi pravdepodobne príde, až keď je neskoro.

7) Dôležitá je všeobecná účasť ľudí, ktorých sa bezpečnosť týka, na nej. Títo ľudia majú zvýšené privilégiá a pokiaľ sa rozhodnú o ne podeliť s neoprávnenými užívateľmi, nie je možné ich ustrážiť. Navyše ich implementovaná bezpečnosť obmedzuje v

ich činnostiach. Preto je vhodné, aby chápali bezpečnosť a snažili sa k nej prispieť, ako by ju mali obchádzať.

8) Rozmanitosť obrany je adaptáciou a doplnkom stratégie hĺbkovej obrany. Spočíva v tom, že jednotlivé obranné prvky hĺbkovej obrany by mali mať rozdielny charakter, aby útočník, ktorý pozná spôsob, ako obísť jeden alebo viacero z nich, bol zastavený ostatnými.

9) Jednoduchosť. Pokiaľ sa niekto v niečom nevyzná, nedokáže to urobiť bezpečné, či už sa jedná o človeka, ktorý navrhuje bezpečnostnú stratégiu, alebo autora programu...

2.3 Sieťová bezpečnosť

S dnešným rozvojom počítačových sietí, najmä Internetu, sa význam ochrany počítačov pred útokom zo siete zvyšuje, pretože tieto siete prinášajú množstvu útočníkov jednoduchý prístup k počítačom iných bez vysokých nákladov, ako tomu bolo v minulosti.

2.3.1 Riziká v počítačových sieťach

Riziká v počítačových sieťach častokrát súvisia so službami, ktoré sú poskytované a ktoré sa využívajú.

Elektronická pošta je jednou z najobľúbenejších sieťových služieb. Ľudí s nízkym bezpečnostným povedomím a neoprávnene vysokou dôverou je možné primäť k vyzradeniu citlivej informácie alebo aktivácii trójskeho koňa či sieťového červa v sieti. Veľkým množstvom prijatej pošty je možné vykonať útok odoprením služby. To sa pri nesprávne nakonfigurovaných poštových serveroch, ktorých je, žiaľ, stále príliš veľa, vykonať aj s malým úsilím útočníka. Programy spracovávajúce poštu (servery i klienti) majú nezriedka zle (alebo vôbec) implementovanú metódu najmenšieho privilégia a stávajú sa tak lákavým cieľom pre útok.

Prístup na WWW cez HTTP je z bezpečnostného hľadiska veľmi nepríjemný, pretože vlastnosti, ktoré ho činia tak lákavým pre užívateľov, zároveň znižujú jeho bezpečnosť. Je totiž tak kompexný, že na mnohé operácie klienta, ale i serveru, je nutné volať externé programy, ktoré nemusia počítať s tým, že svoj vstup dostávajú z nepreverených zdrojov a môžu pri správnom vstupe vykonať operácie, ktoré od nich útočník požaduje. Chyby častokrát obsahujú aj samotní klienti a servery.

Real-time konferenčné služby (IRC, ICQ a iné) sú tiež veľmi používanou službou. Sami o sebe prinášajú malé riziká, problémom však môže byť nadmerná dôvera užívateľov voči tomuto spôsobu komunikácie, čo sa dá využiť na útok metódou tzv. sociálneho inžinierstva.

DNS je služba, ktorá prekladá ľahko zapamätateľné mená sieťových uzlov na ich adresy, ktoré sú potrebné na komunikáciu. Opätovne je najväčším problémom príliš veľká dôvera tomuto systému, mnohí užívatelia a administrátori si neuvedomujú, že preklad mena na adresy a adresy na meno nemusí v konečnom dôsledku vykonávať ten istý server. Staršie servey služby DNS tiež nedostatočne kontrolovali prijaté odpovede a mohlo sa stať, že prijali odpoveď, na ktorú sa vôbec nepýtali a túto potom šírili ďalej.

I samotné protokoly nižšej úrovne, ako sú IP, TCP alebo UDP, sú náchylné na útoky, ktoré obvykle využívajú paket s neštandartnou kombináciou príznakov na útok zabránením služby alebo k tomu, aby tieto pakety prenikli ochranou.

Spoločnými pre viacero protokolov sú útoky cez príkazový kanál, kedy útočník pošle serveru alebo klientovi, s ktorým komunikuje, neočakávanú odpoveď alebo požiadavku, ktorou môže obísť kontroly v ňom implementované, útoky na odoprenie služby hrubou silou, napríklad otvorením veľkého množstva spojení a získavanie informácií pomocou odpočúvania, pokiaľ dáta prechádzajú miestom, kde sa nachádza.

2.3.2 Stratégie sieťovej bezpečnosti

Sieťová bezpečnosť sa rieši v zásade troma spôsobmi. Žiadna bezpečnosť je najjednoduchším riešením. K tomuto riešeniu niet čo dodať.

Ďalší spôsob je bezpečnosť na úrovni počítača. Táto zabezpečuje každý počítač zvlášť. Jej výhodou je, že chráni počítače aj pred útočníkom, ktorý sa nachádza vo vnútornej sieti. Nevýhodou je, že v sieti sa môžu nachádzať počítače s veľkým množstvom rozdielnych aplikácií a OS, ktoré majú vlastné bezpečnostné problémy, ktoré je treba zabezpečiť. Taktiež to znamená zvýšenie počtu privilegovaných užívateľov, z ktorých každý musí mať dobré úmysly a dostatočné vedomosti – inak systém nebude bezpečný. Celkovo sa táto stratégia doporučuje používať len ako doplnková.

Bezpečnosť na úrovni siete zabezpečuje naraz celú počítačovú sieť alebo jej veľkú časť. Jej nevýhodou je, že nechráni proti útočníkom, ktorí sú internými užívateľmi siete.

2.3.3 Firewally

Firewall predstavuje veľmi efektívny typ zabezpečenia na úrovni siete. Je obvykle tvorený jedným alebo viacerými zariadeniami a počítačmi a oddeľuje zabezpečovanú časť siete od zvyšku. Cieľom firewallu je zamedziť neautorizovanej premávke pochádzajúcej mimo chránenej časti siete tak, aby ostatná sieťová premávka prešla.

Základné vlastnosti firewallu sú:

1) Firewall je centrom bezpečnostných rozhodnutí.
2) Pomocou firewallu je možné vynútiť si bezpečnostnú taktiku.
3) Firewall môže účinne zaznamenávať všetku aktivitu medzi týmito dvoma sieťami.
4) Firewall obmedzuje nechcený prístup k službám na chránenej sieti.
5) Firewall nedokáže uchrániť pred spojeniami, ktoré ním neprechádzajú; to sa týka aj škodiacich interných užívateľov.
6) Firewall nedokáže úplne ochrániť pred zatiaľ neznámymi hrozbami, preto je nutné nové hrozby monitorovať a firewall alebo stratégiu bezpečnosti zodpovedajúcim spôsobom upraviť.

Ak má firewall správne pracovať, musí zabraňovať prechodu niektorých dát. Na to sa používa jeden z troch základných spôsobov.

Bastion host

Bastion hostom sa nazýva počítač, ktorý ako jediný (alebo jeden z mála) má prístup aj do vnútornej aj do vonkajšej siete. To ho činí veľmi exponovaným ako cieľ útokov. V súčasnosti sa používa obvykle len v kombinácii s inými spôsobmi zabezpečenia.

Filtrovanie

Pod filtrovaním sa rozumie kontrola prechádzajúcich paketov aktívnym prvkom siete na základe ich hlavičiek a ich následné prepustenie alebo zamietnutie. Účelom filtrovania je poskytnúť užívateľom siete transparentný prístup cez firewall, ako by tam tento ani nebol – s jediným rozdielom, neprípustné pakety sú zakázané.

Filtrovanie je obvykle jednou z funkcií smerovača. Takýto smerovač po aktivácii filtra preskúma každý paket, ktorý chce smerovať, a podľa výsledku sa rozhodne, akú akciu s týmto paketom vykoná. Môže napríklad:

- prepustiť paket, ak vyhovuje bezpečnostnej politike,
- zahodiť paket,
- zahodiť paket a poslať zdroju paketu správu o nedoručiteľnosti (paket TCP s nastaveným príznakom RST alebo správa ICMP „unreachable“),
- zapísať alebo poslať správu o pakete a akcii, ktorú s ním vykonal,
- zmeniť smerovanie paketu,
- zmeniť jedno alebo viacero polí v hlavičke paketu a poslať ho ďalej alebo podrobiť novému skúmaniu (napr. NAT),
- zložiť viacero paketov do jedného (defragmentácia),
- pozdržať paket, kým nebude mať viac informácií na správne rozhodnutie,
- vykonať viacero z týchto činností s tým istým paketom, pokiaľ ich kombinácia dáva zmysel.

Paketový filter sa môže pri rozhodovaní o akcii, ktorú vykoná s paketom, riadiť nasledujúcimi dátami:

- dáta nenachádzajúce sa v pakete, ako napríklad sieťové rozhrania, ktorými prišiel a odíde, aktuálny čas, adresa ďalšieho smerovača...
- dáta z linkovej hlavičky, napr. adresy...
- dáta z hlavičky IP, napr. adresy, typ transportného protokolu, fragmentácia...
- dáta z hlavičky transportného protokolu, napr. zdrojový a cieľový port, príznaky TCP...
- aplikačné dáta, napr. obsah samotného paketu,

– vzťah paketu k ostatným, ktoré smerovačom prešli, napr. ak je súčasťou spojenia cez TCP (takéto paketové filtre sa nazývajú stavovými oproti ostatným – bezstavovým).

Výhodou filtrovania je najmä transparentnosť a schopnosť prepustiť všetky služby, nevýhodou je obmedzená schopnosť analýzy a zmeny dát na aplikačnej úrovni.

Proxy

Proxy je program, ktorý pracuje na aplikačnej úrovni. Skladá sa z rozhodovacej, serverovej a klientskej časti. Klienti sa obracajú so svojimi požiadavkami nie priamo na servery, ale na serverovú časť proxy. Tá požiadavku klienta pošle rozhodovacej časti a keď ju tá schváli a prípadne modifikuje, tak klientskej, ktorá sa spojí priamo s požadovaným serverom. Ak je vyžadovaná odpoveď, tá ide z klientskej do rozhodovacej, odtiaľ do serverovej časti, ktorá ju pošle klientskému počítaču. Proxy server nevyžaduje, aby sa klientské počítače mohli obracať priamo na servery. Spojenie cez proxy môže byť jediným alebo je možné nasadiť smerovače s filtrovaním paketov. Ak sa môžu klienti obracať priamo na servery bez toho, aby prechádzali proxy serverom, firewall neplní svoju funkciu. Ako ale môže klient vedieť, že sa má obrátiť na proxy?

1) Používa sa protokol a aplikácie fungujúce na princípe „ulož a predaj“. Takýmito protokolmi sú napríklad SMTP, NNTP, NTP alebo DNS. Každý server (s istými výnimkami v DNS, ktorých popis prekračuje rozsah tejto práce) pre tieto protokoly totiž má schopnosť požiadavku prijať, uložiť a poslať inému serveru, ku ktorému sa má dostať. Funguje teda v podstate ako proxy. Toto je ideálny prístup, ale funguje len pre obmedzenú množinu protokolov.

2) Modifikuje sa klient. Toto je prípad moderných WWW prehliadačov, ktoré majú medzi voľbami možnosť nastavenia adresy proxy. V takom prípade sa klient spojí s proxy a pošle mu požiadavku modifikovanú pre proxy.

3) Modifikujú sa užívateľské procedúry pri používaní príslušného protokolu. Od tohoto prístupu sa dnes už viac-menej upustilo, pretože vyžaduje, aby sa všetci užívatelia používajúci danú službu naučili zmenenú procedúru.

4) Transparentná proxy je metóda schopná fungovať len pri spolupráci smerovača s paketovým filtrom. Ten presmeruje príslušný druh premávky na proxy pre klienta transparentným spôsobom. V spolupráci so smerovačom si dokáže proxy zistiť adresu a port servera, na ktorý sa klient napojil, takže nie je nutné, aby musel klient používať modifikovaný program alebo užívateľskú procedúru.

Výhodou proxy je schopnosť analýzy, logovania a zmeny dát na aplikačnej úrovni, tiež napríklad schopnosť fungovať ako cache. Nevýhodou je nutnosť použiť pre väčšinu protokolov samostatné proxy servery, úprava klientských programov alebo procedúr, ak sa nepoužije transparentná proxy, a neschopnosť spracovať niektoré služby.

Kombinácie týchto prístupov

Ako vidno, všetky tieto prístupy majú svoje výhody a nevýhody. Firewall by mal byť preto zostavený tak, aby nevýhody jedného z prístupov boli kompenzované výhodami iného, teda tieto prístupy budú kombinované.

2.3.4 Testovanie firewallov

Z predchádzajúceho textu vyplýva, že neoddeliteľnou súčasťou nasadenia bezpečnostných opatrení je aj ich pravidelné testovanie. Nieje tomu inak ani u firewallov.

Obvykle sa firewall testuje najmä z toho hľadiska, či ho nie je možné primäť, aby prepúšťal nelegitímnu premávku. Spôsobov, ktorými môže útočník preniknúť dovnútra, je viacero (napríklad cez bastion host alebo vďaka chybe vo filtri paketov) a testovanie ich musí odhaliť. Rozoznávame dva základné spôsoby testovania.

Testovanie orientované na návrh posudzuje návrh firewallu. Vyžaduje si expertov na jednotlivé komponenty, ktorí dokážu posúdiť, či daný komponent firewallu spĺňa svoju úlohu. Jeho výhodou je, že dokáže odhaliť prakticky všetky nedostatky a chyby v konfigurácii firewallu. Nevýhodou je vysoká náročnosť na čas a financie.

Druhou možnosťou je penetračné testovanie. Je to testovanie bez znalosti štruktúry firewallu takzvanou metódou čiernej skrinky. Testujúci obvykle napodobňuje správanie sa útočníka snažiaceho sa preniknúť do siete, t.j. zameriava sa na viac i menej známe bezpečnostné diery v používaných produktoch, cez ktoré postupne skúša preniknúť. Nevýhodou tohoto systému je, že útočník môže mať znalosti o zraniteľnostiach, ktoré testujúci nemá a ktoré mu uľahčia prienik do systému. Výhodou je, že na rozdiel od predchádzajúceho spôsobu testovania je možné kroky penetračného testovania zautomatizovať. Pri tomto testovaní sa obvykle testuje firewall z vonkajšej i vnútornej siete.

2.4 Analýza existujúcich riešení

2.4.1 Nessus

Nessus je open-source softvérový nástroj na testovanie počítačových systémov komunikujúcich protokolom TCP/IP. Je vytváraný pod licenciou GPL a je voľne dostupný. Nessus používa architektúra klient (užívateľské rozhranie)/server (samotný testovací nástroj), toto riešenie umožňuje centrálny server, ktorý vykonáva všetky testy, zatiaľ čo výsledky sú monitorované a zaznamenané u klienta, ktorý môže bežať na rôznych platformách.

Samotná komunikácia klienta a serveru je zabezpečená šifrovaním.

Nessus podporuje viac platform – server môže bežať na FreeBSD, NetBSD, Sun Solaris, Linux, klient existuje aj pre MS Windows. Existuje aj serverová verzia pre MS Windows, ktorá je však komerčná.

Nessus podporuje plug-in architektúru, ktorá umožňuje ľahké pridávanie testov – každý plug-in vlastne predstavuje jeden konkrétny test. Plug-in pozostáva s jedného súbora, napísaného v jazyku NASL(Nessus Attack Scripting Language). Tento súbor obsahuje vlastný kód pre test, jadro Nessusu tieto súbory de-facto

spúšťa. Samotný test obsahuje svoj krátky popis, ďalej obsahuje informácie do akej kategórie(rodiny) testov patrí a takisto závislosti od predchádzajúcich testov. Výstupom testu, je informácia o zraniteľnosti, ale aj ohodnotenie rizika zraniteľnosti a návod, ako sa tohto rizika zbaviť. Výsledok testov je možné exportovať do HTML. Samotné testy sa pritom nespoliehajú len na detekciu verzií programu, ale naozaj skúšajú „útočiť“. V programe sa ďalej dá nastaviť, aby sa nespúšťali také testy, ktoré by mohli testovanému systému naozaj uškodiť. Ďalšou výhodou programu je veľké množstvo testov, súčasnosti sa ich počet blíži k číslu 10000 (november 2005), a neustále ich pribúda.

Úlohou jadra je na základe už zistených informácii o systéme vybrať tie testy, ktoré sú pre testovaný systém relevantné(napr. ak jeden test zistí, že na serveri beží Apache, potom vykoná ďalšie testy na zistenie jeho verzie alebo zraniteľnosti, ale nebude už napr. vykonávať test pre zistenie verzie IIS). Jadro teda poskytuje akúsi globálnu bázu dát so stromovitou štruktúrou, ktorú testy zdieľajú, a umožňuje výmenu informácií medzi testami.

2.4.2 SAINT

SAINT(Security Administrator's Integrated Network Tool) je ďalším z nástrojov na testovanie systémov komunikujúcich cez TCP/IP. Rovnako ako Nessus podporuje viac platforiem, môže bežať na OS Sun Solaris, FreeBSD, HP-UX, Linux a MAC OS.

SAINT je plne ovládateľný cez WWW prehliadač. Pred samotným testovaním prebehne fáza zbierania informácií, ktoré sa počas fázy testovania využijú na určenie, ktoré testy treba spustiť. Na základe týchto informácii naplánovanie SAINT ďalšie testy, pričom sa podobne ako Nessus nespolieha na verzie programov pri zisťovaní ich zraniteľnosti. Výstupné údaje sú zaznamenané do textového súboru. Spustené testy závisia aj od používateľom nastavenej úrovni testov. Na rozdiel od Nessusu SAINT umožňuje vytváranie nových testov v ľubovoľnom programovacom jazyku.

Dokáže informovať o zraniteľnosti a jej možných dopadoch. Výrobca sleduje zraniteľnosti a vytvára nové testy. Na nekomerčné použitie je SAINT zadarmo.

3 Hrubý návrh riešenia

3.1 Požiadavky na program

1) Rozšíriteľnosť množiny testov. Táto požiadavka vyplýva zo zadania a je jedným z predpokladov využiteľnosti programu; množina zraniteľností sa totiž zo dňa na deň mení a nebolo by udržateľné program príliš často meniť. Splnenie požiadavky bude zabezpečené pomocou externých testov.

2) Umožnenie výberu časti testov podľa istých kritérií. Požiadavka vyplýva z účelu programu, ktorý sa môže využívať na testovanie systémov pred i počas prevádzky. Kritériá by mali byť navrhnuté tak, aby bolo možné testovať zvlášť najdôležitejšie služby a limitovať negatívny vplyv na testovaný systém.

3) Automatické riadenie behu testov. Požiadavka vyplýva z funkcionality programu, niektoré testy totiž vyžadujú pre svoj beh znalosti o systéme, ktoré im môžu poskytnúť iné testy.

4) Príjemné používateľské rozhranie. Je vhodné, aby bol program ovládateľný používateľom bez problémov.

5) Poskytnutie zistených informácií používateľovi. Požiadavka vyplýva priamo zo zadania, bez toho by totiž program nemal zmysel.

3.2 Komponenty programu

1) Testovacie skripty. V tomto projekte bude vytvorená len malá množina testov slúžiaca na demonštráciu funkčnosti alebo skripty poskytujúce informácie, ktoré ostatné skripty využijú. Súčasťou testovacích skriptov budú aj informácie o tom, ako daný skript využíva bázu znalostí, z ktorých jadro určí poradie, v ktorom sa skripty budú spúšťať.

2) Podporná knižnica na beh testovacích skriptov. Bude obsahovať funkcie na komunikáciu testovacích skriptov s jadrom.

3) Testovacie jadro. To bude riadiť beh testovacích skriptov podľa informácií v nich a zároveň na základe ich pokynov modifikovať bázu znalostí. Jadro bude zároveň riadené užívateľským rozhraní, ktorému bude poskytovať informácie o priebehu testovania.

4) Báza znalostí o testovanom systéme. Udržiava ju jadro na základe pokynov testovacích skriptov.

5) Používateľské rozhranie.

3.3 Testovacie skripty

Testovacie skripty budú implementované nezávisle od jadra, čo uľahčuje rozšíriteľnosť programu a adaptáciu na nové bezpečnostné chyby. Testovacie skripty sa budú skladať z dvoch častí: hlavičkového súboru a tela skriptu.

Hlavičkový súbor obsahuje informácie o tom, aké informácie skript z bázy znalostí načítava a aké do nej zapisuje. Budú v ňom uvedené zvlášť vstupné a zvlášť výstupné atribúty, pričom definícia vstupného atribútu bude pozostávať z položiek:

- pôsobnosť atribútu: celý testovaný uzol alebo len jeden jeho port,
- identifikácia atribútu (pre popis jednotlivých atribútov viď informácie v odseku o báze znalostí),
- „povinnosť“ atribútu: či daný atribút musí byť definovaný (v takom prípade musí spĺňať podmienku), môže byť nedefinovaný alebo má byť v prípade jeho neexistencie vyžiadaný od užívateľského rozhrania,
- porovnávacia operácia: pre číselné atribúty je to bežná sada operácií <, >, == a pod., pre číselné rovnosť, nerovnosť a porovnanie s regulárnym výrazom; všetky atribúty môžu byť testované na operáciu „vždy“ (v kombinácii s povinnosťou atribútu sa tak môže zistiť, či je už definovaný) alebo „nikdy“ (v kombinácii s nepovinnosťou sa môže zaistiť vykonávanie skriptu iba ak atribút nie je definovaný),
- hodnota, s ktorou sa má atribút porovnať, ak to má zmysel pre danú operáciu.

Skript môže odmietnuť testovať daný uzol alebo port aj napriek tomu, že v definícii vstupných atribútov uviedol, že má záujem, ale nie naopak.

Definícia výstupného atribútu bude pozostávať z jeho pôsobnosti a identifikácie, ktoré sú obdobné ako u vstupného. Ďalej bude testovací skript v definičnom súbore obsahovať identifikáciu negatívneho vplyvu na testovaný systém, podľa ktorej sa budú členiť do skupín, a iné informácie, ktoré môže jadro na vyžiadanie poskytnúť používateľskému rozhraniu.

Druhou časťou testovacieho skriptu bude samotné jeho telo, ktoré bude mať formu programu vykonateľného na počítači, na ktorom beží jadro. Na komunikáciu s ním bude skript využívať služby podpornej knižnice.

3.4 Podporná knižnica pre beh testovacích skriptov

Bude tvoriť rozhranie medzi testovacími skriptami a jadrom. Bude obsahovať nasledovné funkcie:

- zisťovanie informácií o testovaných uzloch a portoch; jadro na základe informácií z hlavičkového súboru môže testu poskytnúť informácie len o niektorých uzloch alebo portoch z množiny testovaných,
- zisťovanie hodnoty zvolených atribútov a toho, či sú definované,

– zápis hodnoty do zvoleného atribútu; pokiaľ bol atribút predtým nedefinovaný, stane sa definovaným,

– ohlásenie (zapísanie závažnosti a komentára) zisteného bezpečnostného nedostatku uzlu alebo služby bežiacej na danom porte.

3.5 Testovacie jadro

Jadro riadi beh skriptov a zodpovedá za komunikáciu s používateľským rozhraním. Rozhranie jadra a testovacích skriptov bude tvoriť podporná knižnica popísaná v prechádzajúcej kapitole.

Pri riadení behu skriptov bude jadro postupovať nasledovne:

1) identifikácia skriptov, ktoré si používateľ praje spustiť na základe pokynov používateľského rozhrania,

2) načítanie hlavičiek skriptov a identifikácia vstupných a výstupných premenných z bázy znalostí,

3) identifikácia hodnôt, ktoré do bázy znalostí môžu byť poskytnuté používateľom,

4) vyžiadanie týchto hodnôt od používateľského rozhrania; to ich nemusí poskytnúť všetky,

5) vyhľadanie skriptu, ktorý môže bežať za súčasného stavu databázy znalostí bez použitia informácií zadaných používateľom; ak túto podmienku spĺňa viacero skriptov, vyberie sa jeden z nich ľubovoľným spôsobom,

6) ak sa v predchádzajúcom kroku nepodarilo nájsť všetky skripty, zopakuje sa, ale s tým rozdielom, že sa použijú znalosti zadané používateľom,

7) ak sa v 4. alebo 5. kroku podarilo identifikovať skript, ktorý môže bežať, spustí sa, prípadné zmeny, ktoré vykoná, sa uložia do bázy znalostí a pokračuje sa opäť od kroku 4,

8) správa o tom, ktoré skripty nebolo možné spustiť, a o nájdených bezpečnostných nedostatkoch sa predajú používateľskému rozhraniu.

Komunikačné rozhranie medzi testovacím jadrom a používateľským rozhraním bude poskytovať nasledovné funkcie:

- identifikáciu a autentifikáciu používateľa,
- modifikáciu a zobrazenie uzla alebo množiny uzlov, ktoré majú byť testované,
- podávanie informácií o testoch a modifikáciu množiny spustených testov hlavne s dôrazom na ich možný negatívny vplyv na testovaný uzol,
- podávanie informácií o jednotlivých bezpečnostných nedostatkoch,
- informácie o priebehu testovania, najmä počet identifikovaných bezpečnostných problémov a počet prebehnutých testov,

- zobrazenie informácií o výsledkoch testovania.

Informácie o testoch a testovaných bezpečnostných nedostatkoch budú súčasťou testovacích skriptov.

3.6 Báza znalostí

Obsahuje informácie o testovanom systéme, ktoré vypĺňajú testovacie skripty (obvykle špeciálne, ktoré netestujú zraniteľnosť, ale zisťujú informácie) alebo používateľ. Delia sa na znalosti o uzle a znalosti o jeho portoch. Príkladmi znalostí o uzle sú:

- či je dosiahnuteľný (nedosiahnuteľný môže byť napríklad po úspešnom teste vykonávajúcom DoS útok),
- operačný systém a prípadne aj hardware uzlu (ak sa jedná o embedded zariadenie),
- aké testy môžu byť na danom uzle vykonávané z hľadiska negatívneho vplyvu na uzol,

Príkladmi znalostí o porte:

- či na danom porte beží nejaká služba,
- identifikácia RPC služby, ak na danom porte beží,
- typ služby bežiacej na danom porte (HTTP, SMTP a pod.),
- obslužný server,

Testovacie skripty môžu definovať aj iné typy informácie. Za súčasť bázy znalostí sa považujú aj bezpečnostné oznamy, hoci tie nie sú skriptom priamo prístupné.

3.7 Používateľské rozhranie

Používateľské rozhranie bude nezávislé na testovacom jadre a bude s ním komunikovať jedným spojením, aby mohlo bežať aj na inom sieťovom uzle ako jadro.

Používateľské rozhranie musí byť jednoduché, prehľadné ale pritom musí používateľovi poskytovať všetky potrebné informácie a umožniť čo mu najpohodlnejšie testovanie.

Používateľské rozhranie musí poskytovať tieto základné funkcie:

- zadanie testovaného uzlu alebo skupiny uzlov,
- monitorovanie priebehu testovania,
- informovanie o výsledkoch testovania,
- poskytovať informácie o testovacích skriptoch,
- poskytnúť používateľovi možnosť výberu testu alebo skupiny testov ktoré budú testovať určený uzol

V ďalšej časti budú postupne opísané jednotlivé funkcie používateľského rozhrania.

Zadanie testovaného uzlu alebo skupiny uzlov

Používateľ zadá meno alebo IP adresu počítačového uzla ktorý sa bude testovať. Jednotlivé počítačové uzly bude možné zaraďovať do skupín podľa potrieb používateľa. Uzly, alebo celé skupiny uzlov si bude môcť používateľ uložiť a nebude ich musieť ručne zadávať pri každom testovaní.

Monitorovanie priebehu testovania

Počas testovania bude používateľ priebežne informovaný o prebiehajúcom teste – ktorá chyba sa práve testuje, či test prebehol úspešne alebo nie, výsledok testu. Keďže používateľ môže mať naplánovaných veľmi veľa testov, počas testovania sa budú zobrazovať iba najdôležitejšie informácie, pretože výpis veľkého množstva informácií by bolo veľmi neprehľadné.

Používateľ bude mať možnosť monitorovanie priebehu testovania vypnúť, alebo zapnúť podľa potreby.

Informovanie o výsledkoch testovania

Po kompletnom prejdení všetkých naplánovaných testov bude používateľ oboznámený s výsledkom testu.

Výsledky testovania budú obsahovať nasledujúce informácie:

- informácie o testovanom uzly (IP adresa, meno uzlu, meno domény, OS)
- meno alebo id chyby ktorá bola testovaná
- výsledok testovania
- dátum a čas testovania

Ak testy odhalili chybu, bude o nej používateľ informovaný, zobrazí sa stručný popis danej chyby a riešenie ako chybu odstrániť. Celý výsledok testovania bude možné uložiť do súboru vo zvolenom formáte pre neskoršiu analýzu alebo iné potreby používateľa.

4 Použitá literatúra

[1] Dostálek, L., Kabelová, A.: Velký průvodce protokoly TCP/IP a systémem DNS. 3. vyd. Praha: Computer Press, 2002. ISBN 80-7226-675-6.

[2] Chapman, D. B., Zwicky, E. D.: Firewally. Principy budování a udržování. Praha: Computer Press, 1998. ISBN 80-7226-051-0.

[3] Garfunkel, S., Spafford, G.: Bezpečnost v UNIXu a Internetu v praxi. Praha: Computer Press, 1998. ISBN 80-7226-082-0.

[4] Slamka, A.: Testovanie bezpečnostnej brány. Bratislava: Slovenská technická univerzita, Fakulta elektrotechniky a informatiky, 2003. Záverečný projekt.

[5] Semančík, R.: Security management. https://www.sklug.sk/lugcon9/prednasky/2003-11-15_10-05_1/2003-11-15_10-05_1.pdf

[6] Deraison, R.: The Nessus project. http://www.nessus.org

[7] Saint Corporation: SAINT scanning engine. http://www.saintcorporation.com/products/saint_engine.html

[8] Cvečka, M.: Testovanie bezpečnostnej brány. Bratislava: Slovenská technická univerzita, Fakulta elektrotechniky a informatiky, 2002. Záverečný projekt.

[9] SecurityFocus: Vulnerability Archive. http://www.securityfocus.com/bid