Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Študijný odbor: Počítačové Systémy a Siete

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

OBSAH 2

1 ÚVOD 4

2 ANALÝZA PROBLÉMU 5

2.1 Počítačová bezpečnosť 5

2.2 Všeobecné princípy obrany 6

2.3 Sieťová bezpečnosť 8
2.3.1 Riziká v počítačových sieťach 8
2.3.2 Stratégie sieťovej bezpečnosti 9
2.3.3 Firewally 9
2.3.4 Testovanie firewallov 12

2.4 Analýza existujúcich riešení 12
2.4.1 Nessus 12
2.4.2 SAINT 13

3 HRUBÝ NÁVRH RIEŠENIA 14

3.1 Požiadavky na program 14

3.2 Komponenty programu 14

3.3 Testovacie skripty 15

3.4 Podporná knižnica pre beh testovacích skriptov 15

3.5 Testovacie jadro 16

3.6 Báza znalostí 17

3.7 Používateľské rozhranie 17

4 OBSAH VÝUČBOVÉHO PORTÁLU 19

4.1 O penetračnom testovaní 19
4.1.1 Úvod 19
4.1.2 Na čo útočník pri útoku myslí 19
4.1.3 Ciele penetračného testovania 20
4.1.4 Rozsah penetračného testovania 21
4.1.5 Vykonávanie penetračného testu 21
4.1.6 Získavanie informácií 21

4.2 Typy útokov 22
4.2.1 Skenovanie portov 22

4.2.2 Spoofing ... 22
4.2.3 Denial of service (DoS) ... 23
4.2.4 Password attack ... 24
4.2.5 TCP session hijacking ... 24
4.2.6 Únik informácii cez Null Session ... 25
4.2.7 Pretečenie zásobníka v RPC službách ... 25
4.2.8 War dialing ... 25

4.3 Typy nástrojov ... 25
4.3.1 Nessus ... 25
4.3.2 Saint ... 26
4.3.3 Nmap ... 26

5 DOKUMENTÁCIA K IMPLEMENTÁCII ... 28

5.1 Opis fungovania systému ... 28

5.2 Software a hardware využitý na implementáciu ... 28

5.3 Štruktúra databázy ... 29

5.4 Používateľské rozhranie ... 30
5.4.1 Autentifikácia používateľa ... 30
5.4.2 Zoznam dostupných testov ... 30
5.4.3 Nastavenie parametrov testu ... 31
5.4.4 Zobrazenie výsledkov testov ... 31

5.5 Démon na riadenie behu skriptov ... 32

5.6 Implementácia testov ... 33
5.6.1 Zistenie, či je cieľová stanica dosiahnuteľná z penetračného servera ... 33
5.6.2 Zistenie nefiltrovaných portov na cieľovej stanici ... 34
5.6.3 Zistenie verzie služieb bežiacich na cieľovom počítači ... 34
5.6.4 Určenie operačného systému na cieľovom počítači ... 35
5.6.5 Test prítomnosti SMTP servera ... 36
5.6.6 Test prítomnosti HTTP servera ... 36

6 ZHODNOTENIE ... 38

POUŽITÁ LITERATÚRA ... 39

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 tohto 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 tohto 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 komplexný, 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 servery 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 neštandardnou 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 tohto 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 tohto 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 súboru, 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 Obsah výučbového portálu

4.1 O penetračnom testovaní

4.1.1 Úvod

Pravdepodobne ste ten film už videli: hacker, ktorý je ešte príliš mladý na to, aby sa holil sedí o piatej ráno za počítačom, strašne sa potí a láme si hlavu nad počítačom. V miestnosti je bez zjavného dôvodu len tlmené svetlo. V tom sa niekto z kúta miestnosti opýta či to dokáže, veď je to 128 bitová šifra a ostáva už len 5 minút. Hacker sa tvári sebaisto, aj keď je zjavne nervózny. Začína ťukať do klávesnice. Šesť štvorčekov z prehnane veľkými písmenami sa objavuje na obrazovke. Hacker pokračuje v ťukaní do klávesnice za rytmu techno hudby. V štvorčekoch sa pomaly v jednom po druhom začínajú objavovať čísla. Tesne pred tým ako vyprší čas hacker zúfalo buchne ešte raz po klávesnici. V tom sa zjaví na obrazovke posledné číslo. Hacker to zvládol a všetkým sa uľaví. Aj napriek tomu, že je táto scéna zaujímavá je veľmi ďaleko od pravdy. Ak je vaša predstava o penetračnom testovaní práve takáto radšej by ste sa asi mali dať na herectvo. Takže prečo sa táto scéna tak často opakuje vo filmoch? Je to zrejme preto, že je veľmi dramatická a zobrazuje súboj človeka zo strojom.

V skutočnosti je penetračné testovanie asi také zaujímavé ako písanie dizertačnej práce a sociálnej štruktúre mraveniska. Asi by nebolo veľmi zaujímavé sledovať ako hacker strávi cele týždne zbieraním informácií a plánovaním útoku. Takže prečo je toto dôležité? Využitím penetračného testovania sa snažíme nájsť slabiny v zabezpečení počítačovej siete. Čím viac sa bude penetračné testovanie podobať krokom útočníka tým väčšie šance má na úspech.

Na tejto stránke sa pokúsime poskytnúť celkový pohľad na penetračné testovanie a ako by malo prebiehať. Zároveň sa pokúsime priblížiť niektoré typy útokov. Nakoniec sú ešte spomenuté tri už existujúce nástroje, ktoré slúžia na penetračné testovanie.

Dúfame, že tento portál približujúci penetračné testovanie bude slúžiť ako pomôcka na zabezpečenie vlastnej siete komukoľvek kto sem zablúdi

4.1.2 Na čo útočník pri útoku myslí

Útočník môže mať na pokus o preniknutie do siete rôzne dôvody. Aj napriek tomu, že každý útočník pre sieť znamená hrozbu jeho dôvody značne upresňujú cieľ jeho útoku. Pochopením toho, čo by mohlo útočníka motivovať na prienik do konkrétnej siete sa dá lepšie odhadnúť cieľ útoku a teda aj to akým hrozbám daná počítačová sieť čelí. Následne sa dajú lepšie testovať ochranné prvky siete počas penetračného testovania. Niektorých útočníkov môže motivovať aj viacej vecí naraz. V zásade však to čo motivuje útočníka je nasledovné:

Ego a akceptovanie v komunite patria k najčastejším dôvodom útočníkov. Prienikom do počítačových sietí sa snažia upútať na seba pozornosť a získať akceptovanie v elektronickej komunite útočníkov. Prienik do sietí medzinárodných spoločností garantuje útočníkovi publicitu.

Finančný zisk. Útočníci motivovaný finančným ziskom sa dajú v zásade rozdeliť do dvoch skupín. Prvý typ sú tí, ktorý sú motivovaný priamym finančným ziskom. Sú to vlastne obyčajný zlodeji, ktorý sa snažia vykradnúť banku pomocou

svojich technických znalostí. Druhý typ sú tí, ktorý sú motivovaný nepriamym finančným ziskom. Tu sa jedná o získanie informácií (napr. čísla kreditných kariet), ktoré sa neskôr dajú využiť na získanie peňazí.

Výzva. Niektorí útočníci sa pokúšajú o prienik do počítačových sietí iba z toho dôvodu, lebo je to pre nich výzva. Samotný útok môže byť chápaný ako hra šachu, súboj mozgov kombinujúci taktické myslenie, trpezlivosť a mentálnu silu. Na rozdiel od šachu však prieniky do počítačových sietí nemajú svoje pravidlá. Takéto aktivity sú vo väčšine prípadov tiež ilegálne.

Aktivizmus. Útočník môže chcieť preniknúť do siete za cieľom zmeny obsahu web stránky ako člen politického hnutia propagujúci svoje myšlienky. Iné skupiny ľudí zase môžu chcieť získať privátne materiály a následne ich šíriť nelegálnymi cestami. Ďalším cieľom tiež môže byť vytvorenie sofistikovaného denial of service útoku na konkrétnu web stránku za účelom propagácie svojho cieľa.

Odplata. Útočníci motivovaní odplatou sú väčšinou bývalí zamestnanci, ktorí boli buď nespravodlivo vyhodení, alebo majú zlé vzťahy so svojím bývalým zamestnávateľom. Ich nebezpečenstvo spočíva hlavne v tom, že majú dobrú znalosť počítačovej siete, keďže pre danú firmu pracovali. Ich cieľom je väčšinou firme spôsobiť čo najväčšiu škodu.

Špionáž. Nie je neobvyklé, aby sa útočník snažil získať informácie pre tretiu stranu. Príkladom môže byť firma, ktorá sa snaží získať informácie, alebo výskum inej konkurenčnej firmy za cieľom získania vedúceho postavenia na trhu.

Informačný boj. Každá väčšia vojna bola ovplyvnená vznikom nových zbraní. V prvej svetovej vojne to bol samopal a v druhej tank. V dnešnej dobe všetko závisí od informácií a preto je dôležité mať správne informácie čím skôr. V informačnom boji je cieľom podstrčiť nepriateľovi nesprávne informácie a na druhej strane získať taktické informácie pre seba. Takýto informačný boj prebieha hlavne v Spojených štátoch, Číne, Izraeli, Pakistane a Indii. Motiváciou pre útočníkov môže byť práve snaha o úspech v tomto informačnom boji.

4.1.3 Ciele penetračného testovania

Väčšina ľudí vníma výsledok penetračného testovania ako binárnu informáciu. Buď sa do počítačovej siete podarilo preniknúť alebo nie. Nenechajme sa však zviesť touto ilúziou. Úspešnosť penetračného testovania by mala byť hodnotená mierou akou vie prispieť k vyššej bezpečnosti testovanej siete. Treba zhodnotiť zistené nedostatky v bezpečnosti a zhodnotiť ako ich môže útočník zneužiť. Následne je samozrejme potrebné ich odstrániť. To znamená, že nie všetky penetračné testy budú mať rovnaký cieľ a teda sa ani nebudú vykonávať rovnakým spôsobom.

Skôr ako začneme robiť penetračné testovanie je dôležité určiť si jeho cieľ. Cieľom môže byť napríklad ako dlho trvá administrátorovi siete zistiť prítomnosť útočníka. Ciele testovania sú väčšinou nasledovné:

- získanie dôverných informácií
- získanie fyzického prístupu do priestorov alebo k zariadeniu
- získanie administrátorského prístupu do systému alebo systémov
- nechať sa chytiť administrátorom siete
- kompromitovanie aplikácií

- dosiahnutie odmietnutia služby ostatným používateľom
- spôsobenie finančných škôd spoločnosti

4.1.4 Rozsah penetračného testovania

Po tom, čo sme si určili, čo chceme penetračným testovaním dosiahnuť je ešte potrebné určiť si rozsah penetračného testovania. Taktiež nesmieme zabudnúť na časové obdobie v ktorom sa bude testovanie vykonávať. V rozsahu je potrebné si učiť ktoré systémy sa budú testovať, ale ešte dôležitejšie je určiť, ktoré systémy sa testovať nebudú. Pretože medzi penetračným testovaním a počítačovou kriminalitou je veľmi tenká čiara je potrebné si určiť rozsah a pravidlá testovania ešte pred jeho začatím. Je dôležité presne si určiť čo bude počas testovania povolené a čo nie. Minimálne si treba ujasniť odpovede na nasledovné otázky:

- Budú povolené útoky typu denial of dervice?
- Ktoré systémy budú testované a ktoré nie?
- Kedy sa začína obdobie testovania a kedy končí?
- Sú povolené útoky získaním fyzického prístupu?
- Aký prístup do systému má osoba vykonávajúca penetračné testovanie?
- Budú sa testovať reálne alebo testovacie servery?

4.1.5 Vykonávanie penetračného testu

Tak, ako útočník, ani osoba vykonávajúca penetračné testovanie, sa nemôže pokúšať o prienik do počítačovej siete náhodnými útokmi. Bolo by to jednak časovo náročné a zvyšovalo by to pravdepodobnosť odhalenia. Nasledovaním základného postupu testovania sa zvyšujú šance odhalenia bezpečnostných slabín a zároveň sa skracuje čas, ktorý je na to potrebný. Navyše ak použijeme určitý postup je jednoduchšie zaznamenávať výsledky testovania. Nasledujúce kroky poskytujú základnú metodológiu penetračného testovania:

- Určiť najpravdepodobnejší spôsob akým by útočník začal útok
- Lokalizovať slabiny v sieti a aplikáciách
- Určiť akým spôsobom by útočník mohol zneužiť tieto slabiny
- Lokalizovať objekty, ku ktorým sa dá pristupovať, ktoré sa dajú meniť, mazať
- Zistiť, či je možné útočníka odhaliť v procese alebo až po ňom
- Upresniť aká bola charakteristika útoku
- Navrhnúť spôsob prevencie pred skutočným útokom

4.1.6 Získavanie informácií

Na chvíľu zabudnime na svet počítačov a predstavme si bankového lupiča na divokom západe. V tých časoch mohol vojsť do banky, rozbiť trezor a odcválať aj s peniazmi do západu slnka. V dnešnej dobe elektronických alarmov a pancierových trezorov to majú bankový lupiči už ťažšie. Pravdupovediac majú skoro garantované, že ich chytia. Preto sa lupič snaží získať informácie o banke ktorú chce vykradnúť ešte skôr ako sa o to pokúsi. Medzi informácie, ktoré sa pokúsi získať bude určite patriť počet strážcov v banke, pôdorys budovy a únikové cesty. Dôvod je jednoduchý. Čím viac informácií sa mu podarí získať, tým väčšie sú jeho šance na úspech.

Počítačový zlodeji sa správajú rovnako. Pred tým ako sa pokúsia o útok na počítačovú sieť pokúsia sa o nej získať čo najviac informácií. Čím lepšiu predstavu o štruktúre siete majú, tým väčšie sú ich šance na to, že preniknú do siete, spôsobia škodu a zmiznú skôr ako si ich niekto všimne. Takže aké informácie sa pokúša útočník získať?

Konfigurácia systému. Znalosť konfigurácie systému pomáha útočníkovi pripraviť si útok, pretože ak vie, že firma používa napríklad iba Microsoft produkty vie, že sa nemusí obťažovať s útokmi, ktoré fungujú len na operačnom systéme Red Hat.

Reálne používateľské kontá. Znalosť platných používateľských mien je výhodná najme pri brute - force útokoch, kde útočník ušetrí čas tým, že nemusí skúšať lámať heslo neexistujúceho konta.

Kontaktné informácie. Telefónne číslo na spoločnosť, do ktorej sa útočník snaží preniknúť je veľkou výhodou najmä pri War Dialing útoku. Výhodou je taktiež poznať mená zamestnancov a to hlavne v prípade social engineering útokov.

Externé servery a servery vzdialeného prístupu. Servery vzdialeného prístupu umožňujú zamestnancom prístup do firemného intranetu a k využívaniu jeho služieb v prípade, že potrebujú pracovať mimo kancelárie. Využitím týchto serverov sa útočník môže pokúsiť o prienik do firemnej siete.

Obchodný partneri a nedávne spájanie spoločností. V prípade, že sa dve spoločnosti spájajú väčšinou sa spájajú aj ich firemné siete. Je to komplikovaný proces a ak nie je vykonaný dôkladne môže vzniknúť veľké množstvo bezpečnostných slabín, ktoré útočník môže využiť.

Väčšinu týchto informácií dokáže útočník získať z verejne dostupných zdrojov ako napríklad firemných web stránok. Ďalším zdrojom informácií môže byť doménový registrátor. Zároveň nie je problém zistiť verejnú IP spoločnosti. Toto všetko napomáha útočníkovi. Ale ako sa proti tomu brániť? Dokonalá ochrana určite neexituje, ale základným pravidlom je, že verejne dostupné by mali byť iba tie informácie, ktoré naozaj musia byť. Všetko ostatné by malo ostať tajné.

4.2 Typy útokov

4.2.1 Skenovanie portov

Scanovanie portov je jedna z najpopulárnejších techník útočníkov, určená na objavenie bežiacich servisov a ich portov. Všetky počítače spojené do lokálnej počítačovej siete (LAN), alebo Internet majú spustených veľa servisov a majú povolených veľa portov. Scanovanie portov pomáha potenciálnemu útočníkovi nájsť otvorené porty. Scanovanie portov sa skladá zo zasielania správ postupne na všetky porty. Potom podľa druhu odpovede môžme zistiť, ktorý port sa používa a ktorý nie. Na základe týchto informácií môže útočník zamerať svoje útoky na dané porty a servisy využitím známych chýb.

4.2.2 Spoofing

Spoofing je typ útoku, pri ktorom osoba, alebo program úspešne maskuje svoju totožnosť a tvári sa ako druhá osoba. Veľa protokolov používaných v TCP/IP je náchylných na spoofing útoky.

Príklady spoofing útokov sú IP spoofing, ARP spoofing, man-in-the-middle útoky, DNS spoofing, web spoofing - tiež sa označuje ako phishing.

IP Spoofing

IP spoofing je technika, pri ktorej útočník posiela IP pakety, ktorých zdrojová IP adresa je sfalšovaná. Sfalšovaná zdrojová adresa môže vyzerať ako adresa iného stroja.

Ako ochrana pred IP spoofingom môže byť použité filtrovanie paketov. Gateway by mala filtrovať pakety z vonkajšej siete, ale so zdrojovou IP adresou patriacej vnútornej sieti. Toto zabráni útočníkovi z vonkajšej siete falšovať adresy vnútornej siete. V ideálnom prípade by mali byť filtrované aj pakety z vnútornej siete s adresou, ktorá nepatrí vnútornej sieti. Toto zabráni útočníkovi z vnútornej siete uskutočňovať útoky proti vzdialeným strojom.

Niektoré protokoly vyššej vrstvy poskytujú vlastnú ochranu proti IP spoofingu, napr. TCP protokol používa číslovanie pri spojení so vzdialeným strojom na uistenie, že prichádzajúce pakety sú časťou spojenia. Zlá implementácia TCP číslovania v niektorých operačných systémoch umožňuje, že IP spoofing je stále uskutočniteľný.

Útočník je limitovaný faktom, že akákoľvek odpoveď nepríde jemu, ale hostu, ktorého IP adresu sfalšoval. Toto robí IP spoofing útok náročný na realizovanie.

Niektoré nástroje na IP spoofing sú Libnet alebo PacGen.

ARP Spoofing

Princíp ARP spoofingu je v posielaní falošných ARP správ. Tieto správy obsahujú falošnú MAC adresu, čo spôsobí, že sieťové zariadenia ako switche budú preposielať framy nesprávnym hostom. Pomocou ARP spoofingu je možné odchytávať pakety a sledovať komunikáciu na switchovanej sieti.

Predpokladáme, že A chce komunikovať s C a B chce odpočúvať. Pri prvom pokuse o komunikáciu potrebuje stroj A vedieť fyzickú (MAC) adresu stroja C a preto vyšle do siete ARP broadcast s požiadavkou, že nech mu stroj C pošle svoju MAC adresu. Dôležité je, že ide o broadcast rámec a teda ho dostane aj stroj B. A to preto, lebo switch musí poslať broadcast paket do všetkých portov. Tento stroj chce odpočúvať a preto je na situáciu pripravený. Rýchlo vygeneruje ARP odpoveď, kde uvedie IP adresu stroja C, avšak svoju MAC adresu a paket pošle naspäť. Môže sa stať, že táto falošná odpoveď príde do stroja A ešte skôr, než skutočná odpoved od stroja C. Je však už neskoro. Stroj A uložil do cache IP adresu stroja C, ale priradil jej MAC adresu stroja B. Stroj A teraz začína komunikovať so strojom C avšak v skutočnosti všetko ide do stroja B. Stroj B môže teraz všetko zaznamenávať. Na udržanie komunikácie medzi A a C musí stroj B navyše forwardovať všetky rámce medzi A a C, čo však v linuxe nie je problém. Táto situácia je ešte nebezpečnejšia ako pasívne sniffovanie, pretože stroj B môže obsah paketov aj meniť. V tomto prípade sa jedná a tzv. man-in-the-middle útok.

Nástroje pre ARP spoofing sú napr. Cain , Ettercap , alebo Windows ARP spoofer.

MAC Spoofing

Princíp MAC spoofingu spočíva v sfalšovaní svojej fyzickej (MAC) adresy. Tento útok môže útočník uskutočniť ak sa niekde vyžaduje autentifikácia na základe MAC adresy, čo býva celkom bežné napr. u Wifi sietí.

Nástroj na MAC spoofing je MacMakeUp.

4.2.3 Denial of service (DoS)

Ako už naznačuje názov denial of service je typ útoku pri ktorom sa snaží útočník zabrániť používateľovi prístup k službe. Aj napriek tomu, že pod denial of service útok môže spadať aj výbuch sopky alebo zemetrasenie v našom prípade sa jedná čisto o útoky

vykonávané počítačom na iný počítač. Tieto útoky sa dajú rozdeliť to týchto základných kategórií:

- preťaženie alebo iný útok na sieťové spojenie medzi používateľom a serverom
- útok na preťaženie zdrojov, ktorý môže byť spôsobený preťažením CPU alebo zahltením pamäte
- zrušenie služby buď na úrovni aplikácie alebo na úrovni počítača spôsobené napríklad cielenou zmenou konfigurácie
- fyzické útoky

Pri testovaní siete na odolnosť voči tomuto typu útokov treba byť veľmi opatrný. Bezpečnosť sietí sa väčšinou testuje za plnej prevádzky a nešikovne spravený test na denial of service útok by mohol spôsobiť samotné odmietnutie služby. Existujú však metódy na nedeštruktívne testovanie systémov na denial of service útoky.

4.2.4 Password attack

Útoky počas ktorých sa hádajú heslá patria medzi najobľúbenejšie aspekty penetračného testovania. Heslá sa dajú získať rôznymi spôsobmi. Dajú sa uhádnuť, dajú sa nájsť napísané alebo sa dajú získať z operačného systému. Heslá získané od operačného systému sú buď uložené v plain text formáte, alebo sú zakódované dekódovateľnou šifrou. Operačné systémy väčšinou využívajú hash funkcie ktoré zoberú heslo a zakódujú ho nedekódovateľným spôsobom. Pri verifikácii sa porovnáva hash zadaného hesla a uložený hash.

Útoky na hasované heslá sa väčšinou vykonávajú kombináciou dictionary útokov a brute-force metód. Pri dictionary útokoch si útočník pripraví zoznam často používaných hesiel a pomocou nich sa snaží preniknúť do systému. Pri brute-force metóde sa postupne skúšajú všetky možné kombinácie znakov ako heslo. Napríklad aaa , aab , aac a tak ďalej.

Medzi miesta na ktorých sa dajú získať heslá patria aj tieto:

- bat súbory a skripty
- web stránky
- aplikácie a časti operačných systémov, ktoré si pamätajú heslá
- pod klávesnicou a na papierikoch prilepených o monitor
- v excelovskom dokumente “schovanom” na zdieľanom adresári
- v súboroch zabudnutých po inštalácii
- na sieti, hlavne ak systém akceptuje plain text heslá

4.2.5 TCP session hijacking

Útok tohto typu znamená, že útočník sa pripojí do spojenia medzi dvomi počítačmi. Keďže väčšinou sa autentifikácia vyskytuje na začiatku TCP spojenia, je možné sa pripojiť do už vytvoreného spojenia. Známa je metóda presmerovávania IP paketov. To umožňuje útočníkovi z počítača A v počítačovej sieti odpočúvať konverzáciu medzi počítačmi B a C presmerovaním paketov cez jeho systém.

4.2.6 Únik informácii cez Null Session

Null session, alebo aj anonymné prihlásenie je mechanizmus, ktorý umožňuje anonymnému užívateľovi získať niektoré informácie po sieti. Ide napríklad o užívateľské mená a zdieľané adresáre, alebo umožňuje prihlásiť sa bez autentifikácie. Tento prístup je používaný aplikáciami ako explorer.exe na vymenovanie zdieľaných zdrojov na vzdialenom serveri. Na Windows NT, alebo Windows 2000 sa konto SYSTEM, alebo aj LocalSystem používa pre niektoré lokálne systémové služby. Toto konto sa tiež používa pri rôznych kritických operáciách. Ak jeden stroj potrebuje získať dáta z druhého, konto SYSTEM otvorí anonymné spojenie na druhý stroj - prihlási sa. Konto SYSTEM má teoreticky neobmedzené práva a nemá žiadne heslo. Preto sa na toto konto nemôže prihlásiť bežný užívateľ. Útočník však môže použiť mechanizmus vzdialeného anonymného prihlásenia ako NULL a získať tak niektoré informácie.

4.2.7 Pretečenie zásobníka v RPC službách

Remote Procedure Call (RPC) dovoľujú programom na jednom stroji vykonať nejaký program na inom stroji. Často sa používajú hlavne v sieťových službách ako sú NFS zieľanie súborov a NIS . Mnohé chyby v kóde RPC spôsobujú že sú tieto chyby aktívne zneužívané na prieniky. Dá sa ľahko ukázať, že väčšina distribuovaných DoS útokov na prelome rokov 1999 a 2000 bola vykonaná práve vďaka zraniteľnostiam v RPC. Napríklad útok na U.S. military system bol možný kvôli stovkám serverov na ministerstve obrany v USA , ktoré obsahovali RPC chyby.

4.2.8 War dialing

War dialing je technika, ktorá používa počítačový program na automatické volanie na čísla z istého rozsahu za účelom zistiť, či na niektorom čísle je pripojený modem. Program automaticky vytáča definovaný zoznam čísiel a zapisuje si do databázy tie čísla, ktoré sa úspešne pripoja k modemu. Niektoré programy dokážu identifikovať operačný systém bežiaci na danom počítači a tiež môžu vykonávať automatické penetračné testovanie. V takom prípade program vyskúša preddefinovaný zoznam bežných používateľských mien a hesiel za účelom získania prístupu do systému.

Rozšírením dostupnosti pripojenia na Internet sa táto metóda stala zastaranou. Najznámejšie nástroje pre war dialing boli Toneloc pre DOS a ShokDial pre Linux

4.3 Typy nástrojov

4.3.1 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 platforiem, to znamená, že 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 súboru, 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.

Bližšie informácie o tomto nástroji sa dajú nájsť na stránke www.nessus.org

4.3.2 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.

Bližšie informácie o tomto nástroji sa dajú nájsť na stránke www.saintcorporation.com

4.3.3 Nmap

Nmap je voľne šíriteľný nástroj na skenovanie počítačových sietí, ktorý pomáha i pri bezpečnostnom audite. Nmap pracuje so špeciálne zostavenými IP paketmi, pomocou ktorých dokáže zistiť, aké uzly sa na danej sieti vyskytujú, aké služby tieto uzly poskytujú, aké operačné systémy a sieťové aplikácie na nich bežia, aké paketové filtre alebo firewally blokujú prístup k danému uzlu a mnoho iných charakteristík.

Výstupom z Nmapu je zoznam skenovaný uzlov, pričom v závislosti od použitých parametrov ku každému z nich môže uviesť dodatočné informácie. V najbežnejšie používaných módoch, keď skenuje porty na uzloch, je jednou z nich tabuľka „pozoruhodných“ portov. V nej sa pre každý uvedený port nachádza jeho číslo a protokol, meno služby a jeho stav. Stav môže nadobúdať jednu z nasledovných hodnôt:

- open znamená, že na danom porte nejaká aplikácia prijíma spojenie alebo pakety,
- closed znamená, že na danom porte momentálne žiadna aplikácia neprijíma spojenia,
- filtered znamená, že daný port je odfiltrovaný na firewalle alebo inej prekážke,
- unfiltered znamená, že Nmap nevedel rozhodnúť, či je daný port open alebo closed,

• open|filtered a closed|filtered znamená, že Nmap nevedel rozhodnúť, či je daný port open alebo filtered, respektíve closed alebo filtered.

Okrem tejto tabuľky vie Nmap poskytnúť iné informácie, ako napríklad obslužné servery na jednotlivých portoch, operačné systémy uzlov, zariadenia (ak sa nejedná o univerzálne počítače) alebo MAC adresy.

Stránka programu Nmap je www.insecure.org/nmap.

5 Dokumentácia k implementácii

5.1 Opis fungovania systému

Predošlá časť dokumentu bola vytvorená počas prvého semestra. Zaoberala sa hlavne analýzou problematiky a návrhom samotného systému. Počas implementácie však boli niektoré časti systému upravené oproti pôvodnému návrhu. Na úvod dokumentácie k implementácii by sme preto radi uviedli stručný popis spôsobu fungovania výsledného systému.

Systém penetračného testovania môžeme rozdeliť na 4 hlavné časti: databáza, deamon, používateľské rozhranie a samotné testy. Najprv uvedieme, akým spôsobom sú tieto časti poprepájané. Spôsob fungovania jednotlivých častí systému bude uvedený neskôr.

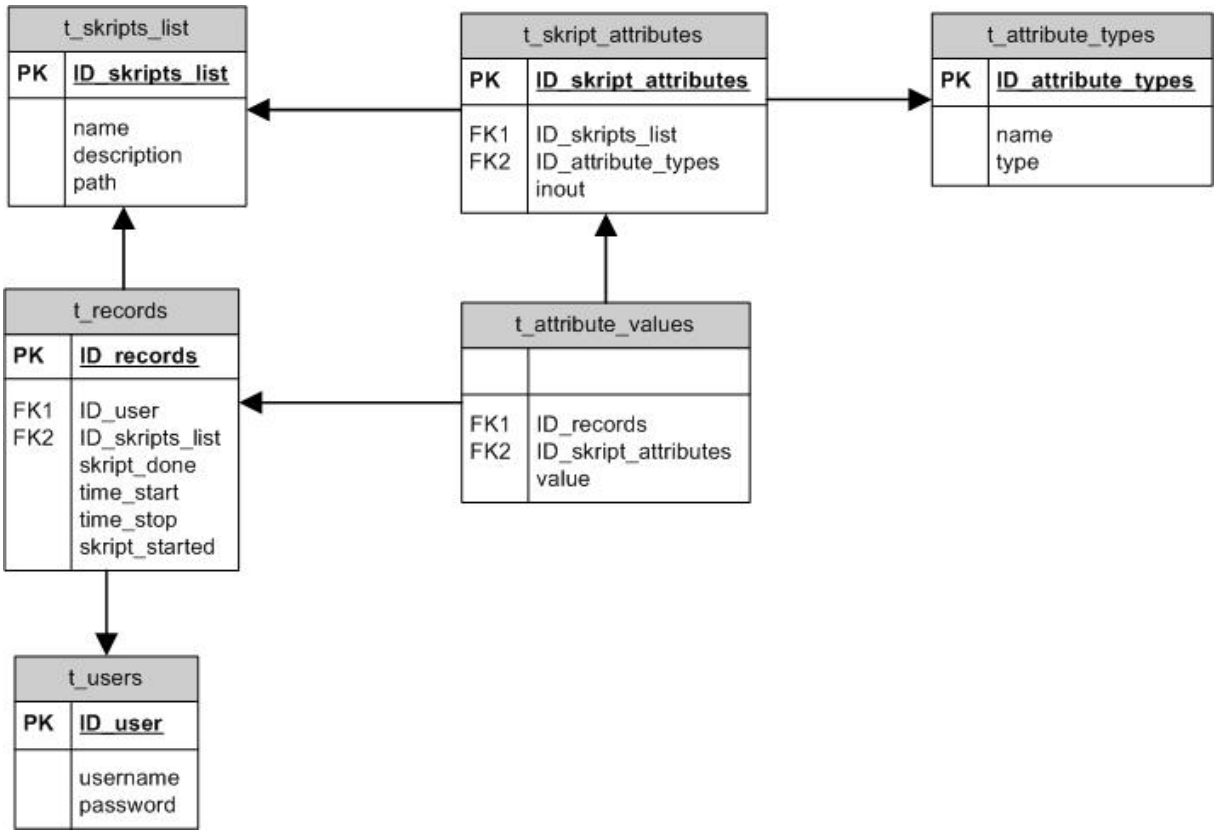
Používateľské rozhranie zabezpečuje interakciu s používateľom. Umožňuje mu jednak pozerať si výsledky predošlých testov a spúšťať nové testy. Táto funkcionalita je zabezpečená čítaním a zapisovaním z/do databázy. Ak chce používateľ spustiť test zadá jeho parametre a používateľské rozhranie vytvorí záznam v databáze, ktorý hovorí o tom aký test s akými parametrami sa má spustiť. Deamon je o pridaní testu modifikovaný. Automaticky si zistí parametre a zavolá konkrétny test s požadovanými parametrami. Každý test je spustiteľný súbor. Test sa vykoná a výsledky vráti deamonovy. Ten výsledky zapíše do databázy a označí test ako vykonaný. Po ukončení testu sa deamon pozrie do databázy, či je potrebné vykonať ďalšie skripty.

Používateľské rozhranie si výsledky testov prečíta z databázy a vráti ich používateľovi. Ten má možnosť pozerať si výsledky všetkých testov, prípadne spúšťať dalšie testy.

5.2 Software a hardware využitý na implementáciu

Navrhnutý systém bol implementovaný na počítači s procesorom Intel(R) Pentium(R) 4 CPU 2.66GHz a 512 Mb RAM. Ako operačný systém bol využitý Linux debian 2.4.27-2-386. Samotné používateľské rozhranie bolo implementované v php a prístupné cez web na adrese stroja 147.175.99.151/~team22. Ako web server bol využitý apache web server. Na databázu bolo využité postgre_sql.

5.3 Štruktúra databázy



Databáza na ukladanie dát je implementovaná v postgre sql. Jej štruktúra je znázornená na predchádzajúcom obraázku. Obsahuje 6 tabuliek v ktorých sú uložené dáta. Všetky tabuľky okrem jednej obsahujú ID typu integer, ktoré sa využíva ako primárny kľúč (PK) v danej tabuľke. Cudzie kľúče sú taktiež typu integer a v obrázku sú označené ako FK.

Tabuľka *t_users* obsahuje informácie o používateľoch ktorý môžu využívať systém. Každý používateľ má priradené meno (username) a heslo (password). Oba atribúty sú typu text.

V tabuľke *t_skripts_list* je zoznam dostupných testov. Obsahuje názov testu (name), popis testu (description) a cestu k spustiteľnému súboru testu (path). Všetky tieto atribúty sú typu text.

Tabuľka *t_attribute_types* obsahuje zoznam možných atribútov pre skripty. Ako príklad je možné uviesť či je atribút IP adresa, alebo port. Je tu obsiahnutá informácia o mene atribútu (name) a type atribútu (type).

Tabuľka *t_skript_attributes* obsahuje zoznam atribútov pre konkrétne skripty. Cudzí kľúč *ID_skripts_lists* hovorí o tom, ktorý skript má aké atribúty. Ďalší cudzí kľúč *ID_attribute_types* poskytuje informáciu o type atribútu. V tejto tabuľke je zároveň informácia o tom, či je atribút vstupný alebo výstupný (inout). Tento atribút je typu bool.

Hlavnou tabuľkou celej databázy je *t_records*. Tu je uložený zoznam všetkých testov, ktoré boli spustené, alebo čakajú na spustenie. Informácie a stave testu je uložená v atribúte *skript_done*. Tabuľka odkazuje na používateľa, ktorý test spustil a zároveň na to, ktorý test

bol spustený. Informácie o čase spustenia (*time_start*) a ukončenia (*time_stop*) testu sú taktiež uložené v tejto tabuľke.

Hodnoty vstupných a výstupných parametrov testu sú uložené v tabuľke *t_attribute_values*. Je to tabuľak cudzích kľúčov, ktorá na základe *ID_records* a *ID_skripts_attributes* priraďuje hodnoty vstupným a výstupným atribútom jednotlivých testov.

5.4 Používateľské rozhranie

Používateľské rozhranie zabezpečuje autentifikáciu používateľa, ukladá požiadavky na testy do bázy dát a prezentuje výsledky testov používateľovi. Je vytvorené v jazyku PHP, dáta sú uložené v SQL databáze.

5.4.1 Autentifikácia používateľa

Autentifikáciu zabezpečuje skript login.php. Autentifikácia je uskutočnená pomocou HTTP protokolu. Pri autentifikácii sa najprv nastaví hodnota serverovej premennej *$_SERVER['PHP_AUTH_USER']*, táto sa porovná s menom a heslom v databáze. Ak sú zhodné, používateľ môže pristúpiť do systému. Používateľ je autentifikovaný pokiaľ nezavrie prehliadač. Autentifikáciu zabezpečuje nasledujúca funkcia

```
function autentifikacia () {
  header("WWW-Authenticate: Basic realm=\"Autentifikujte sa\"");
  header("HTTP/1.0 401 Unauthorized");
  echo "Teba nepoznam";
  die();
}
```

Okrem toho sa v každom ďalšom skripte testuje, či je hodnota premennej *$_SERVER['PHP_AUTH_USER']* nastavená a teda, či je používateľ prihlásený týmo kódom.

```
if(!isset($_SERVER["PHP_AUTH_USER"])) header('Location: login.php');
```

5.4.2 Zoznam dostupných testov

Prezeranie dostupných testov zabezpečuje skript tests.php. Test získa potrebné dáta z databázy týmto príkazom:

```
SELECT ID_skripts_list as id,name,description FROM t_skripts_list
```

Rozhranie poskytne zoznam testov ako linky. Po kliknutí na konkrétny link sa otvorí okno, pomocou ktorého je možne nastaviť parametre testu a daný test spustiť.

5.4.3 Nastavenie parametrov testu

Nastavenie parametrov testu a uloženie požiadavky na test do databázy zabezpečí skript attributes.php. Skript najprv vytiahne z databázy všetky vstupné atribúty testu a zobrazí ich používateľovi. Vytiahnutie atribútov urobí týmto príkazom.

```
SELECT b.id_skript_attributs as attr_id, c.name as type_name
FROM t_skripts_list a,t_skript_attributes b,t_attribute_types c
WHERE a.ID_skripts_list=b.ID_skripts_list AND
        c.ID_attribute_types=b.ID_attribute_types AND
        b.inout=false AND
        a.ID_skripts_list=$id"
```

Používateľ vyplní vstupné atribúty, skript ich hodnoty uloží do databázy, tak isto uloží do tabuľky *t_records* požiadavku na test.

Požiadavku na test uloží týmto príkazom:

```
INSERT INTO t_records (id_user,id_skripts_list,skript_done,time_start) VALUES(
        (SELECT ID_user FROM t_users WHERE
        username='".$_SERVER['PHP_AUTH_USER']."' LIMIT 1),
                ".$id.", 'f' , now())
```

Hodnoty atribútov uloží v cykle týmto príkazom. Mená a hodnoty atribútov sú v poli *$pom, resp. $_GET[$pom]*.

```
INSERT INTO t_attribute_values (id_records,id_skript_attributes,value) VALUES(
        currval('t_records_id_records_seq'),
        '".$pom[$i]."',
        '".$_GET[$pom[$i]]."')
```

5.4.4 Zobrazenie výsledkov testov

Výsledky skončených testov, rovnako aj prebiehajúce testy zobrazuje skript *results.php*. Skript zobrazí všetky testy aktuálneho používateľa. Skript najprv vytiahne všetky testy používateľa a potom, pre každý test vytiahne všetky jeho atribúty.

Všetky testy zistí tento príkaz:

```
SELECT
        c.name as name,
        a.ID_records as id_rec,
        a.skript_done as is_done,
        a.time_start as time_start,
        a.time_stop as time_stop,
```

```
        b.username as username
    FROM t_records a, t_users b, t_skripts_list c
    WHERE
        a.id_skripts_list=c.id_skripts_list AND
        a.ID_user=(SELECT b.ID_user FROM t_users WHERE
            username='".$_SERVER['PHP_AUTH_USER']."' LIMIT 1)
```

Všetky vstupné atribúty zistí nasledujúci príkaz, pre výstupné atribúty sa zmení v príkaze iba jedna hodnota.

```
SELECT
    c.name as name,
    a.value as value
FROM t_attribute_values a, t_skript_attributes b, t_attribute_types c
    WHERE
        a.ID_records=".$data["id_rec"]." AND
        a.ID_skript_attributes=b.ID_skript_attributes AND
        b.ID_attribute_types=c.ID_attribute_types AND
        b.inout=false
```

5.5 Démon na riadenie behu skriptov

Spúšťanie:

```
main [-p <num>] [-c <time>] [-h <host>] [-u <user>] [-w <pass>] [-d <database>] [-D]
  -p    max. parallel checks
  -c    check database at least every <time> seconds
  -h    host with database
  -u    database user
  -w    ...password
  -d    database name
  -b    go to background
  -D    emit debug messages
```

Základom programu je cyklus, v ktorom pomocou funkcie poll() čaká na udalosti od bázy dát (implementovanej ako databáza v PostgreSQL) a bežiacich skriptov. Udalosti a spôsob reakcie na nich:

- vypršanie časového limitu (timeout) – len ak bolo zadané parametrom -c; skontroluje, či beží maximálny počet skriptov (parameter -p), ak nie, pošle báze dát požiadavku na určenie ďalšieho skriptu na spustenie,
- uživateľské rozhranie pomocou bázy dát oznámi, že pridalo novú požiadavku na spustenie skriptu do bázy dát – skontroluje, či beží maximálny počet skriptov (parameter -p), ak nie, pošle báze dát požiadavku na určenie ďalšieho skriptu na spustenie,
- báza dát zaslala odpoveď na požiadavku na určenie ďalšieho skriptu na spustenie (vrátane jeho mena a atribútov) – vytvorí nový proces, nastaví komunikačné kanály a spustí príslušný skript,

- báza dát zaslala odpoveď na zápis atribútu – pošle príslušnému skriptu odpoveď, či sa operácia vydarila,
- niektorý skript poslal dáta komunikačným kanálom – prečíta dáta, uloží ich do pamäťového buffera a skontroluje, či prišiel celý príkaz; ak áno, zašle databáze požiadavku zodpovedajúcu príslušnému príkazu a zmaže buffer,
- niektorý z komunikačných kanálov bol uzavretý – ak sa tak udialo pre ukončenie skriptu, zruší komunikačné kanály a pošle báze dát oznam o ukončení behu skriptu,
- bol zaslaný signál pomocou volania jadra kill() - podľa typu signálu udalosť ignoruje alebo ukončí bežiace skripty, spojenie s bázou dát a nakoniec aj seba samého.

5.6 Implementácia testov

V tejto časti je uvedený spôsob implementácie niektorých penetračných testov. Obsahuje popis jednotlivých testov, ich vstupné a výstupné parametre ako aj zdrojový kód.

5.6.1 Zistenie, či je cieľová stanica dosiahnuteľná z penetračného servera

```
#!/bin/sh
#vstup 2 parametre: IP adresa
#vystup SETP ALIVE 1 - reachable
#vystup SETP ALIVE 0 - unreachable


if ping -c3 -q $2 > /dev/null; then
        echo SETP ALIVE 1
        read OK
else
        echo SETP ALIVE 0
        read OK
fi
exit 0
```

vstupné perametre
formát: PING_final.sh IP <IP adresa cieľového počítača, typ inet>

výstupné perametre

formát: SETP ALIVE <1, typ znak> - stroj je dosiahnuteľný
SETP ALIVE <0, typ znak> - stroj je nedosiahnuteľný

Skript pomocou príkazu ping vyšle k cieľovému uzlu 3 ICMP pakety. Ak ani na jeden nedostane od cieľového počítača odpoveď, skript pošle na svoj štandartný výstup SETP

ALIVE 0. V prípade, že dostane od cieľového stroja odpoveď, pošle na svoj štandartný výstup SETP ALIVE 1. Skript vždy po poslaní správy na svoj štandartný výstup čaká na signál OK od jadra a pokračuje ďalej. Po úspešnom skončení skript vracia hodnotu 0.

5.6.2 Zistenie nefiltrovaných portov na cieľovej stanici

```
#!/bin/sh

date=`date | awk '{print $4}' | sed -e 's/://g'`

jeden=1

nmap -sT $2 | grep ^[0-9] > /tmp/${date}port.log

pocet=`cat /tmp/${date}port.log | wc -l`

while [ ${pocet} -gt ${jeden} ]; do
        echo SETP PORT `head -n ${jeden} /tmp/${date}port.log | tail +${jeden} | awk '{print $1}'` STATUS `head -n ${jeden} /tmp/${date}port.log | tail +${jeden} | awk '{print $2}'`
        read OK
        jeden=`expr $jeden + 1`
done

rm /tmp/${date}port.log

exit 0
```

vstupmé parametre:
formát: PORTY_final.sh IP <IP adresa cieľového počítača, typ inet>

výstupné parametre:
formát: SETP PORT <číslo portu, reťazec> STATUS <názov služby, typ reťazec>

Skript do pomocnej premennej date určí a upravý lokálny čas na penetračnom servery – premenná je potrebná na určenie názvu pomocného súboru, do ktorého sa zapisujú pomocné údaje. Do pomecného súboru /tmp/${date}port.log sa vloží výstup príkazu nmap. Do premennej pocet sa vloží počet zistených portov – tj. počet portov, u ktorých je možné určiť či sú otvorené alebo zavreté. Potom skript vypisuje na svoj štandartný výstup jednotlivé riadky pomocného súboru vo vopred stanovenom formáte.

SETP PORT <číslo portu/protkol, typ reťazec> STATUS <meno služby, typ reťazec>

5.6.3 Zistenie verzie služieb bežiacich na cieľovom počítači

```
#!/bin/sh
date=`date | awk '{print $4}' | sed -e 's/://g'`
```

```
nmap -sV $2 | grep " open " > /tmp/${date}verzia.log
pocet=`cat /tmp/${date}verzia.log | wc -l`
jeden=1

while [ ${pocet} -ge ${jeden} ]; do
    echo SETP PORT `head -n ${jeden} /tmp/${date}verzia.log | tail +${jeden} | awk
'{print $1}'` NAME `head -n ${jeden} /tmp/${date}verzia.log | tail  +${jeden} | tail -c +24`
    read OK
    jeden=`expr $jeden + 1`
done
rm /tmp/${date}verzia.log
exit 0
```

vstupmé parametre:
formát: VERZIE_final.sh IP <IP adresa cieľového počítača, typ inet>

výstupné parametre:
formát: SETP PORT <číslo portu, reťazec> NAME <názov služby, typ reťazec>

Skript do pomocnej premennej date určí a upravý lokálny čas na penetračnom servery – premenná je potrebná na určenie názvu pomocného súboru, do ktorého sa zapisujú pomocné údaje. Do pomecného súboru /tmp/${date}port.log sa vloží výstup príkazu nmap. Do premennej pocet sa vloží počet otvorených portov – tj. počet portov, u ktorých je možné určiť verziu služby. Potom skript vypisuje na svoj štandartný výstup jednotlivé riadky pomocného súboru vo vopred stanovenom formáte.

SETP PORT <číslo portu, reťazec> NAME <názov služby, typ reťazec>

5.6.4 Určenie operačného systému na cieľovom počítači

```
#!/bin/sh
date=`date | awk '{print $4}' | sed -e 's/://g'`
nmap -O --osscan_limit $2 > /tmp/${date}OS.log
if grep ^Running: /tmp/${date}OS.log > /dev/null; then
    echo SETP OS `grep ^Running /tmp/${date}OS.log | tail -c +9`
    read OK
else
    echo SETP OS undefined
    read OK
fi
rm /tmp/${date}OS.log
exit 0
```

vstupmé parametre:
formát: OS_final.sh IP <IP adresa cieľového počítača, typ inet>

výstupné parametre:
v prípade, že sa podarilo zistiť meno OS
formát: SETP OS <meno operačného systému, reťazec>

v prípade, že sa podarilo zistiť meno OS
formát: SETP OS undefined

Skript do pomocnej premennej date určí a upravý lokálny čas na penetračnom servery – premenná je potrebná na určenie názvu pomocného súboru, do ktorého sa zapisujú pomocné údaje. Do pomecného súboru /tmp/${date}OS.log sa vloží výstup príkazu nmap. V prípade, že sa nmap-u podarilo určiť verziu OS na cieľovom stroji, pošle na svoj výstup riadok Running: OS. Toto využíva skript, v prípade že vo výstupe nmap-u objavý reťazec "Running:", podarilo sa určiť OS. Skript vypíše na svoj výstup meno OS v stanovenom formáte.

Poznámka: tento skript musí spúšťať root. Tenot problém ešte potrebujeme vyriešiť.

5.6.5 Test prítomnosti SMTP servera

```
#!/bin/sh
if nmap -sV $2 | grep ^[0-9] | grep " open " | grep ^"25/"; then
        echo SETP SMTP_SERV 1
        read OK
else
        echo SETP SMTP_SERV 0
        read OK
fi
exit 0
```

vstupmé parametre:
formát: DETEKCIA_SMTP_SERVERA.sh IP <IP adresa cieľového počítača, typ inet>

výstupné parametre:
formát: SETP SMTP_SERV <1, znak> - Na cieľovom stroji je spustený SMTP server
SETP SMTP_SERV<0, znak> - Na cieľovom stroji nieje spustený SMTP server

Skript si pomocou služby nmap zistí stav portu 25 na ktorom počúva SMTP server a jeho verziu. V prípade, že je port 25 otvorený, hodnota výstupného parametra SMTP_SERV sa nastavý na 1, v opačnom prípade na 0.

5.6.6 Test prítomnosti HTTP servera

```
#!/bin/sh
if nmap -sV $2 | grep ^[0-9] | grep " open " | grep ^"80/"; then
        echo SETP HTTP_SERV 1
        read OK
else
```

```
        echo SETP HTTP_SERV 0
        read OK
    fi
    exit 0
```

vstupmé parametre:
formát: DETEKCIA_HTTP_SERVERA.sh IP <IP adresa cieľového počítača, typ inet>

výstupné parametre:
formát: SETP HTTP_SERV <1, znak> - Na cieľovom stroji je spustený HTTP server
SETP HTTP_SERV<0, znak> - Na cieľovom stroji nieje spustený HTTP server

Skript si pomocou služby nmap zistí stav portu 80 na ktorom počúva HTTP server a jeho verziu. V prípade, že je port 80 otvorený, hodnota výstupného parametra HTTP_SERV sa nastavý na 1, v opačnom prípade na 0.

6 Zhodnotenie

Prvý semester bol zameraný hlavne na zbieranie informácií ohľadne penetračného testovania. Bolo zozbierané množstvo informácií o danej problematike, pričom výsledkom práce je portál obsahujúci informácie o penetračnom testovaní. Ten poskytuje nezasväteným osobám bližší pohľad na bezpečnosť počítačových sietí ako aj na spôsob prenikov do nich. Ďalším, ale nie menej dôležitým výsledkom prvého semestra bol návrh spôsobu implementácie portálu pre penetračné testovanie.

Počas druhého semestra celý tím vynaložil veľké úsilie na vytvorenie samotného portálu umožňujúceho penetračné testovanie. Práca bola zameraná na vytvorenie deamona spúšťajúceho penetračné testy a na používateľské rozhranie, ale aj na samotné testy. Výsledkom je funkčný portál umožňujúci penetračné testovanie.

Musíme však priznať, že táto prvá verzia portálu penetračného testovania ešte stále obsahuje drobné chybičky, ktoré plánujeme do konca semestra odstrániť. Na to, aby bol portál použiteľný je samozrejme nutné naprogramovať ďalšie testy.

Ďalšou funkciou portálu, na ktorej sa stále pracuje je umožnenie spúšťania na sebe závislých testov. Tu je hlavnou myšlienkou to, že systém bude na základe predchádzajúcich testov vedieť aké ďalšie testy je ešte vhodné spustiť. Ak sa napríklad zistí, že testovaný systém počúva na porte 80, systém bude vedieť, že je vhodné ešte zistiť aký typ http servera beží na danom stroji. Podľa verzie servera bude následne možné testovať konkrétny server na bezpečnostné slabiny.

Neoddeliteľnou súčasťou projektu bude samozrejme aj používateľská príručka. Tá bude zverejnená zároveň s výslednou verziou portálu pre penetračné testovanie.

Na záver treba len dodať, že všetci členovia tímu na projekte tvrdo pracovali a myslíme si, že aj napriek drobným nedostatkom je priebežný stav projektu je na veľmi vysokej úrovni.

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