Útoky na informační systémy vedené hrubou silou.

The brutal force attacks against information systems

Pavel Trefil

Bakalářská práce
2008

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2007/2008

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Pavel TREFIL**
Studijní program: **B 3902 Inženýrská informatika**
Studijní obor: **Informační technologie**

Téma práce: **Útoky na informační systémy vedené hrubou silou.**

Zásady pro vypracování:

1. **Vyhledejte vhodné zdroje řešící problematiku útoků hrubou silou na informační systémy.**
2. **Analyzujte současné možnosti útoků z pohledu jejich rizikovosti.**
3. **Realizujte útoky v bezpečném prostředí a navrhněte možnosti jejich eliminace na úrovni současných poznatků.**
4. **Vyhodnoťte úspěšnost řešení a definujte silná a slabá místa zvolených řešení.**

Rozsah práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Stephen Prata, 2001, Mistrovství v C++, Praha, Comuter Press.**
2. **Andrew Troelsen,2006, C-sharp a .NET 2.0 Profesionálně, Brno, Zoner Press.**
3. **Herbert Schild,2001, Nauč se sám C++, Praha, SoftPress.**
4. **Libor Dostálek a kolektiv,2001, Velký průvodce protokoly TCP/IP: Bezpečnost, Praha, Comuter Press.**
5. **Carl Endorf,Hacking – detekce a prevence počítačového útoku, 2003,Praha, Grada.**
6. **Joel Scambray,Hacking bez tajemství, 3. aktualizované vydání, 2005,Praha, Comuter Press.**
7. **Josef Pirkl, Síťové programování pod Windows a programování Internetu, 2001, české Budějovice, Kopp.**

Vedoucí bakalářské práce: **doc. Mgr. Roman Jašek, Ph.D.**
Ústav informatiky a statistiky

Datum zadání bakalářské práce: **20. února 2008**

Termín odevzdání bakalářské práce: **5. května 2008**

Ve Zlíně dne 20. února 2008

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Má bakalářská práce popisuje problematiku útoků vedených hrubou silou na informační systémy. Práce seznámí s vlivem abeced a délky hesla na časovou náročnost. V praktické části jsou rozebrány konkrétní útoky na operační systémy Windows XP a Windows Vista.

Klíčová slova: útok hrubou silou, heslo, bezpečnost

ABSTRACT

My bachelor thesis describes the problems of brutal force attacks on information systems. This work gets acquainted with the effects of the alphabets and passwords lengths on time-consuming process. There are analysed some concrete attacks on operation systems Windows XP and Windows Vista in the practical part.

Keywords: brutal force attack, password, security

Úvodem bych chtěl poděkovat doc. Mgr. Romanu Jaškovi za vedení a připomínky při psaní bakalářské práce. Dále bych chtěl poděkovat všem vyučujícím UTB, kteří mne po dobu studia obohacovali o další poznatky v oblasti informatiky.

Prohlašuji, že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně dne 8.5.2008 …….

Pavel Trefil

OBSAH

ÚVOD

V dnešní době, která je charakterizována jako období, kdy se informační technologie rozvíjí nebývalým tempem, se neubráníme otázce ochraně informací. V minulosti kdy byly informace ukládány zejména v „papírové“ podobě, spočívala ochrana informací „pouze“ v jejich fyzickém zabezpečení. A to jistě znamenalo značnou pozornost a úsilí. Zároveň je však nutno podotknout, že díky nutnosti uložení fyzického záznamu informace (papíru), byla tato informace velmi komplikovaně dostupná. Cesta ze Zlína do centrály Interpolu ve francouzkém Lyonu je jistě krásným výletem, nicméně absolvujeme-li ho pouze z důvodu přečtení jedné A4 strany, jeví se tato informace jako poněkud nákladná.

Vzhledem k současnému stavu, kdy se většina informací ukládá v elektronické podobě je, nutné se o tuto devizu náležitě starat. Na těchto informacích je často závislá činnost nejenom firem, ale bez nadsázky snad i fungování celé společnost. Ačkoliv si tento stav uvědomujeme, ne vždy se podle toho ke svým informacím chováme. Mám na mysli hlavně jejich ukládání a zabezpečení. Fyzické ukládání dnes probíhá na různé typy nosičů jako jsou pevné disky, CD a DVD média, DAT pásky, flash paměti atd. Tyto informace potřebujeme nejenom ukládat, ale také sdílet, posílat a modifikovat. Při práci s daty je téměř vždy nutné definovat bezpečnostní politiku, kterou budeme používat.

Musíme si uvědomit, jakou má informace obrovskou cenu a zároveň může být snadno zneužitelná. Dnes se prakticky všechny záznamy ukládají v elektronické podobě. Snad jenom pro dokreslení uvádím příklady z běžného života. Bankovní transakce, veškerá obchodní činnost firmy, záznamy o studiu na universitách, daňová evidence, bakalářské práce, zdravotní dokumentace, telefonní komunikace, burza cenných papírů, katastrální záznamy.

To vše nás vede k nutnosti zajistit následující úkony při práci s informacemi:

- Dostupnost informace
- Modifikovatelnost informace
- Kopírovatelnost informace
- Možnost ukládat informace
- Zasílat informace

Všechny tyto dílčí činnosti musíme zajistit s dostatečnou mírou bezpečnosti.

Snad každému z nás je jasné že např. na Internetu někdo informace ukládá (poskytuje) a většina z nás jsou pouze čitateli. V některých případech nemáme k těmto informacím přístup vůbec. Např. elektronické bankovnictví je přístupné jen majitelům bankovních účtů. Na tomto jednoduchém příkladě si uvědomíme, že ne vždy je žádoucí, aby naše informace byla veřejná. Jednou z nejpoužívanějších bezpečnostních politik pro přístup k systému je autentizace uživatele za pomocí hesla a případně uživatelského jména. Každý z nás zná a používá PIN u mobilních telefonů, PIN u platebních karet, heslo pro přístup do sítě, heslo pro přístup k elektronické poště. Je nám tedy naprosto jasné že ten, kdo zná tyto autentizační údaje, může využívat daný prostředek. Záhy nás přitom napadne, jestli tyto údaje nemůže zjistit někdo jiný. Pokud si PIN mobilního telefonu napíšeme na baterii, tak stačí, aby někdo odklopil kryt a má vše potřebné. Druhou možností je zkoušet různé varianty čísel a pokusit se tento PIN uhádnout. Zde se dostáváme k podstatě útoků „hrubou silou“.

I. TEORETICKÁ ČÁST

1 ÚTOKY VEDENÉ HRUBOU SILOU

1.1 Princip vedení útoků hrubou silou

Jak již bylo uvedeno dříve, základní princip vedení „útoků hrubou silou“ spočívá ve zkoušení kombinací znaků při autentizaci do systému, ke kterému nemáme přístupové právo. Druhou možností je použití rozsáhlého slovníku, ve kterém jsou již hesla vygenerovaná. V neposlední řadě můžeme vyzkoušet kombinaci slovníku společně s kombinací dalších znaků.

1.2 Použití kombinací znaků

Tato metoda se opírá o vyzkoušení všech možných kombinací znaků. Základním problémem je výběr množiny znaků (abecedy) a délky zkoušeného hesla.

1.2.1 Vliv velikosti použité abecedy a délky hesla

Snad nejsnadnější použitou abecedou pro heslo můžou být číslice 0 až 9. Nyní se podívejme na možné počty kombinací. Např. pro dvou-znakové heslo použijeme následující kombinace 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11 až 99. Z toho vyplývá jednoduchý vztah pro výpočet počtu kombinací.

$10^2 = 100$ kombinací při použití abecedy 0 až 9 a délce hesla dva znaky.

Obecně zavedeme následující proměnné.

A – počet znaků v abecedě

x - délka hesla

N – počet kombinací

Pro zobecnění platí

$$N = A^x$$

Z výše uvedeného vztahu je patrné že se jedná o exponenciální funkci. Vše platí pouze za předpokladu, že známe délku hesla a počet znaků v abecedě. Nyní se zaměříme na vliv velikosti abecedy. Následující tabulka ukazuje nárůst počtu kombinací při použité abecedě.

velikost abecedy	1	2	3	4	5	6	7
10 znaků	10	100	1 000	10 000	100 000	1 000 000	10 000 000
26 znaků	26	676	17 576	456 976	11 881 376	308 915 776	8 031 810 176
36 znaků	36	1 296	46 656	1 679 616	60 466 176	2 176 782 336	78 364 164 096
52 znaků	52	2 704	140 608	7 311 616	380 204 032	19 770 609 664	1 028 071 702 528
62 znaků	62	3 844	238 328	14 776 336	916 132 832	56 800 235 584	3 521 614 606 208
86 znaků	86	7 396	636 056	54 700 816	4 704 270 176	404 567 235 136	34 792 782 221 696
96 znaků	96	9 216	884 736	84 934 656	8 153 726 976	782 757 789 696	75 144 747 810 816

Tab.1: Vliv jednotlivých abeced na maximální počet hesel

Jenom pro úplnost uvádím příklady používaných abeced. Samozřejmě že skutečná abeceda může být libovolnou kombinací níže uvedených. Otázkou zůstává, nakolik můžeme

používat diakritické znaky české abecedy (ěščřžýáíéůúťďň), případně znaky specifické pro jiné země. Většina systémů podporuje pro zadání hesla i znaky regionálních abeced. Z praxe ale víme, že to není příliš používané. Problémy s přepínáním klávesnic, případně zadání ESC sekvence je málo komfortní. Bez ohledu na to, že se uživatel, může dostat do těžké situace, kdy heslo není schopen zadat.

	Příklad abecedy	Popis
10 znaků	0123456789	čísla
26 znaků	ABCDEFGHIJKLMNOPQR STUVWXYZ	malá nebo velká písmena
36 znaků	ABCDEFGHIJKLMNOPQR STUVWXYZ0123456789	malá nebo velká písmena a čísla
52 znaků	AaBbCcDdEeFfGgHhIiJjKk LlMmNnOoPpQqRrSsTtUuV vWwXxYyZz	malá a velká písmena
62 znaků	AaBbCcDdEeFfGgHhIiJjKk LlMmNnOoPpQqRrSsTtUuV vWwXxYyZz0123456789	malá, velká písmena a čísla
86 znaků	AaBbCcDdEeFfGgHhIiJjKk LlMmNnOoPpQqRrSsTtUuV vWwXxYyZz<SP>!"#$%&'( )*+,-./:;<=>?@[\]^_`{|}~	malá, velká písmena a speciální znaky
96 znaků	AaBbCcDdEeFfGgHhIiJjKk LlMmNnOoPpQqRrSsTtUuV vWwXxYyZz<SP>!"#$%&'( )*+,-./:;<=>?@[\]^_`{|}~	malá, velká písmena, speciální znaky a čísla

Tab.2: Příklady možných abeced

Na příklad použijeme-li malá písmena plus číslice, tak při maximální délce hesla 5 znaků musíme vyzkoušet 60 466 176 kombinací. Používáme abecedu s počtem 36-ti znaků. Počet pokusů je maximální, za předpokladu že známe délku hesla. Pokud délku hesla neznáme, musíme ještě přičíst počty kombinací 4, 3, 2, 1 znakového hesla.

$$N_{sum} = A^{x_{\max}} + A^{x_{(\max -1)}} + A^{x_{(\max -2)}} + \ldots + A^{x_{(0)}}$$

x_{max} – maximální délka hesla

N_{sum} – maximální celkový počet pokusů

Tento celkový počet pokusů je důležitý zejména při výpočtu strojového času, který budeme potřebovat k úspěšnému nalezení hesla.

1.3 Použití slovníkového útoku

Tato metoda předpokládá, že máme k dispozici velkou databázi slov-hesel, které budeme používat při útoku. Na Internetu je takových slovníků k dispozici několik. Zpravidla jsou rozděleny do souborů podle různé délky slova. Zde se paradoxně stává náš mateřský jazyk výhodou. V několika z nich jsem zkoušel vyhledat mnou užívaná hesla, a byl jsem mile překvapen. Slova jako „miluska“, „slivovice“, „burcak“ atd. tam opravdu nenajdeme. Možná že by nám pomohl třeba Ottův slovník naučný. Dalším možným zdrojem může být například portál wikipedia. Slovníky jsou zpravidla rozděleny do zájmových oblastí. Někdy bývají členěny podle délky slova. Nalezneme je například na následující adrese http://mozektevidi.com/wordlisty.html. Vzhledem k tomu že úspěšnost této metody nelze přesně vyhodnotit, nebudu se tímto typem dále zabývat. Je to úkol spíše pro etymology nebo lingvisty. Posledním možným způsobem tvorby hesla je kombinace slov ze slovníku s dalšími přidanými znaky. Možná by stálo za úvahu využívat slovníky v informačních systémech jako prevenci při tvorbě hesla. Jednoduše neumožnit uživateli systému vytvořit heslo, které je ve slovníku. Je opět diskutabilní, které slovníky používat. Na druhou stranu pokud se k těmto slovníkům dostane útočník, může je vyloučit ze svých pokusů. Tím se samozřejmě zvýší účinnost útoku.

1.4 Hlavní překážkou je čas

Jak již bylo uvedeno výše, pokud známe abecedu a délku hesla není problém vypočítat maximální možný počet pokusů. Podstatně důležitější je čas, který budeme potřebovat na ověření jediného pokusu.

$$t_{sum} = t \times N_{sum}$$

t_{sum} – maximální čas

t – čas pro ověření jednoho pokusu

Vše platí za předpokladu, že systém, který ověřuje heslo se chová zcela konstantně při každém pokusu. V praxi to ovšem nemusí vždy platit. Těžko donutíme např. jakýkoli operační systém, aby použil vždy stejný počet taktů procesoru. Je nutné si uvědomit, že systém ve chvíli ověřování našeho hesla může vykonávat jiný proces a budeme tedy čekat. Pokud chceme náš výpočet času upřesnit je nutné naměřené hodnoty času zprůměrovat např. za 1000 pokusů. Pozdržení ověření hesla je rovněž silnou zbraní proti útokům hrubou silou. V praxi to může vypadat následovně. Při každém dalším neúspěšném pokusu systém dobu odpovědi zdvojnásobí. Tímto způsobem můžeme útočníkovi velmi zkomplikovat vlastní atak. Jsme- li na straně útočníka, budeme se vždy snažit najít takovou metodu, která nám úspěšnost ověření hesla vrací co nejrychleji. Samozřejmě že velkou roli zde hraje vlastní výkon systému. Výkon procesorů se neustále zvyšuje a tím roste i výkon programů. Tedy i výkon našeho atakovacího programu. Další možností zrychlení útoku je jeho paralelizace. To samozřejmě musí umožňovat vlastní atakovaný systém. Jde-li o online systémy, např. serverový OS můžeme útok vést z několika PC současně. Tímto způsobem se dá útok urychlit. Pokud atakujeme off-line systém, např. při hádání hesla v excelovém souboru, můžeme tento soubor zkopírovat na několik systémů a pracovat paralelně. Při paralelním útoku musíme zajistit součinnost jednotlivých útočníků. Rozdělit jednotlivým vláknům skupiny hesel. Případně může jedno vlákno systému hesla pouze generovat do zásobníku a jednotlivé systémy si je postupně vyzvedávají a zkouší.

Je těžké definovat, kdy budeme považovat útok za úspěšný. Někdy nám postačí heslo znát třeba za týden, jindy již po hodině pokusů považujeme útok za neúspěšný. Čím delší čas budeme potřebovat na vlastní atak, tím je větší pravděpodobnost, že budeme odhaleni nebo že si uživatel mezi tím heslo změní. U off-line systémů (zaheslovaný soubor v

excelu) nehrozí žádné riziko našeho odhalení. Ovšem jen v případě kdy si soubor můžeme zkopírovat do našeho systému, kde budeme provádět atak.

2 ÚTOK HRUBOU SILOU Z POHLEDU ÚTOČNÍKA

2.1 Proč se zabývat možnou strategií útočníka?

Na případný útok se můžeme lépe připravit, čím více budeme vědět o útočníkovi. Je důležité znát praktiky útočníka. Jsme sice v obrovské nevýhodě, protože útočník je vždy o krok napřed. V následujících kapitolách jsem shrnul, jak by potenciální útočník mohl postupovat.

2.2 Analýza, příprava půdy pro útok

Budu-li útočníkem, rozhodně se na daný útok co nejvíce připravím. Jedna z prvních věcí, které budu zjišťovat je délka hesla. Připomeňme skutečnost, že ve většině systémů se při psaní hesla objevují náhradní znaky typu „* “. Počet hvězdiček je totožný s délkou hesla. To znamená, mám-li možnost svou „oběť“ sledovat při zapisování hesla určitě se budu snažit všimnout si délky hesla. Pokud ne, nezbývá než začít zkoušením jednoznakových hesel a pokračovat dále. Druhým významným faktorem, který ovlivňuje počet pokusů je použitá abeceda. Zde platí podobně jako u délky hesla. Co nejvíce informací odpozorovat. Vidím-li že „oběť“ nepoužívá klávesy Num-Lock mám hned o 10 znaků menší abecedu. Naprosto stejně s diakritickými znaky, případně s klávesou Shift pro psaní velkých znaků abecedy. Budu-li se pokoušet o atak systému, ke kterému mám přístup jako běžný uživatel typu „User, Host“. A snažím-li se při útoku zjistit heslo např. „Administratora“, pak je dobré zkoušet si postupně měnit svoje vlastní hesla. A tak zjistit celou možnou abecedu, kterou systém podporuje pro zadání hesla. Znám-li abecedu a délku hesla již můžu vypočítat maximální možný počet pokusů.

2.3 Výběr metody útoku, čas

Snad nejprimitivnější metodou pro útok hrubou silou je zkoušet při přihlášení hesla zapisovat ručně z klávesnice. Pokud budu takto postupovat, nejsem útočník, ale zapomětlivý uživatel. Určitě se ještě vyplatí otočit klávesnici a podívat se zespodu, jestli tam není lístek s používanými hesly. Někdy bývají umístěny i na monitoru. Průzkumy v tomto směru jsou z hlediska bezpečnosti naprosto děsivé.

Ale zpět k „seriózním“ útokům. V každém případě se snažím najít metodu, která mi co nejdříve heslo ověří. Zde se vyplatí použít obvyklé programovací nástroje. Jestliže znám systém, který budu atakovat, vyzkouším např. 1000 pokusů a zjistím průměrnou dobu

jednoho ověření. Ve chvíli kdy znám maximální počet pokusů a čas potřebný k jednomu ověření. Potom vypočítám čas, který budu potřebovat k úspěšnému nalezení hesla.

2.4 Vlastní útok

Vlastní útok je již jednoduchou záležitostí. Spustím program, který se snaží heslo nalézt, a pak čekám. Když se útok podaří a zjistím tak heslo daného uživatele, nesmím zapomenout na zametení stop. Určitě se podívám do logů, jestli nebyly zaznamenávány neúspěšné pokusy o přihlášení. Dále již nechci spekulovat, nikdy jsem v pozici útočníka nebyl. Pokud ovšem nepočítáme tuto práci.

3 OBRANA PROTI ÚTOKŮM HRUBOU SILOU

3.1 Použití silných hesel

Téměř každá literatura, věnující se bezpečnosti informačních systémů doporučuje použití tzv. silných hesel. Co dnes můžeme považovat za silné heslo, nelze jednoznačně určit. Vždy bude záležet na konkrétním systému a dalším zabezpečení. Silná hesla stojí zpravidla na dvou pilířích. Jedním z nich je délka hesla a druhým je zvolená abeceda. A tak pokud použijeme pro tvorbu hesla kombinaci malých a velkých písmen společně s čísly a speciálními znaky. Dostaneme abecedu o 96-ti znacích. Již při délce hesla 5 znaků jde vytvořit 8 miliard kombinací. Pokud vybereme abecedu s malými písmeny, musíme mít délku hesla 7 znaků, abychom dosáhli stejného počtu kombinací.

3.2 Omezit počet neúspěšných pokusů

Největším nepřítelem pro útočníka se stává čas. Proto je dalším způsobem jak zabezpečit systém proti útokům hrubou silou prodlužovat odpověď na špatné heslo. Například první dvě odpovědi na špatné heslo akceptujeme v běžně možné odezvě systému. Na každé další neúspěšné přihlášení reagujeme záměrně v delším čase. Případně tuto dobu můžeme po každém neúspěšném pokusu například zdvojnásobit. Toto je velmi účinná metoda a jistě útočníka odradí. Práci běžného uživatele to nijak neovlivní. Dalším krokem je alespoň dočasné zneplatnění uživatelského účtu. Takto zneplatněný účet může zpravidla zpět povolit jenom správce (administrátor) systému. Použití tohoto opatření je však velmi diskutabilní. Budu-li na straně útočníka, můžu takto opakovaně uživateli znepříjemňovat život. I když se mi nepodaří heslo zjistit, můžu mu takto jeho účet soustavně zneplatňovat. Kdo je v tuto chvíli vítězem lze jen těžko určit.

Dalším rizikem může být paralelizace. V dnešní době, kdy jsou systémy tvořeny někdy i celou farmou serverů, může ověření uživatele často provádět kterýkoli z nich. Tato skutečnost sice přispívá k velkému výkonu, ale ten se v případě útoku stává naopak slabou stránkou. Určitě se vyplatí zamyslet, zda by ověření uživatele neměl v danou chvíli provádět pokaždé pouze jeden systém.

3.3 Záznam neúspěšných pokusů

Pro každého správce informačního systému je důležité vědět, co se po dobu běhu informačního systému děje. Pro tyto účely zpravidla slouží záznamy událostí. Z pohledu bezpečnosti se jedná o další silný nástroj, jak zabezpečit systém. Pokud zaznamenáváme neúspěšné pokusy přihlášení, lze tak velmi přesně odhalit útočníka. V každém záznamu událostí uvidíme čas případně i ostatní informace. Pokud zde uvidíme neúspěšné pokusy, které se opakují každých 5ms, je nám jasné že se nejedná o zdatnou písařku, ale o útok! Určitě se vyplatí dále sledovat, odkud je útok veden, případně jaký proces útok provádí atd. Následné zabezpečení systému již záleží na správci a možnostech systému.

4 CO LZE ATAKOVAT HRUBOU SILOU?

4.1 Některé příklady systémů

Jenom z důvodu nebezpečnosti útoků uvádím následující příklady.

- PIN bankovní karty, zde platí jednoduché zabezpečení. Po třetím pokusu přicházíme o kartu. Je nutné si uvědomit, že se používá jako abeceda pouze čísla. Délka hesla je jen 4 znaky. To znamená, že stačí 1000 pokusů. Proto tak striktní omezení.
- PIN mobilního telefonu, velmi podobná situace. Po třetím pokusu se mobil zablokuje.
- Zabezpečovací systémy při vstupu do objektů. Zkoušel jsem v našem bytovém domě. Na jiné zabezpečení než délka kódu jsem nenarazil. Prostě stačí vyzkoušet 1 000 000 pokusů a můžete jít dál. Určitě ale záleží na kvalitě zabezpečovacího zařízení.
- Obecně informační systémy. Tady musíme zařadit operační systémy, Internetové aplikace, CRM systémy. Zjednodušeně řečeno každá aplikace, která vyžaduje ověření jménem a heslem.
- Dokumenty v elektronické podobě, které jsou proti čtení případně zápisu zabezpečeny vstupním heslem. Zde bych jenom rád upozornil na možnost paralelizace útoku.
- Instalační sériová čísla. Asi každý z nás již někdy instaloval nějakou aplikaci, která na začátku instalace vyžaduje vstupní kód (product key). Při tomto útoku platí, že výsledkem je celá množina kódů.
- Jistě existuje ještě mnoho dalších možností, kde lze využít útok hrubou silou. Dnešní přetechnizovaná doba je k tomu jako stvořená.

II. PRAKTICKÁ ČÁST

5 ÚTOK NA OPERAČNÍ SYSTÉMY FIRMY MICROSOFT

5.1 Proč právě Microsoft?

Ať se nám to líbí nebo ne, je dnes Microsoft nejrozšířenější platformou operačních systémů. Svůj výzkum budu provádět na OS Microsoft Windows XP CZ Professional a Microsoft Windows Vista Business CZ. Tyto dva operační systémy patří v podnikové sféře mezi nejrozšířenější. Testy jsem prováděl v roce 2007 a začátkem roku 2008. Vždy byly nainstalovány veškeré možné bezpečnostní aktualizace. Vzhledem k tomu že další aktualizace a servis packy neustále vznikají, můžou být výsledky mých pokusů za několik měsíců zcela jiné. Zároveň chci upozornit, že se jedná pouze o jednu z mnoha možných metod jak provádět útok na zjištění hesla.

5.2 Použitá metodika

Dnes není v silách žádného jednotlivce vyzkoušet veškeré přístupy do operačních systémů, musel jsem zvolit jednoduchý model, na kterém jsem začal systémy atakovat. Musíme si uvědomit, že v operačních systémech běží řada služeb (procesů), které je možné atakovat prostřednictvím sítě, jiných procesů atd. Předpokládám, že do každého systému máme přístup jako uživatel typu „host“. Tedy uživatel, který má minimální oprávnění cokoli nastavovat, upravovat registry, měnit nastavení zabezpečení. Vždy známe jméno uživatele, jehož heslo se snažíme zjistit útokem hrubou silou. To znamená, budeme se snažit přihlásit jako administrátor a postupně zkoušet hesla. Postupně budeme nastavovat bezpečnostní politiku tak abychom zabránili útokům hrubou silou. Veškeré testy jsou vztaženy k výkonnosti systému. To znamená, že nejsou zcela objektivní. Na jiném systému, který používá odlišný hardware budou výsledky jiné. Pro úplnost uvádím konfiguraci zmíněných systémů. V dalším textu budu oba systémy zjednodušeně označovat Windows XP a Windows Vista.

	Windows XP	Windows Vista
Operační systém	Verze 5.1.2600	Verze 6.0.6000
Operační paměť	512 MB	2GB
Procesor	Intel 1,6 GHz	Intel 1,8 GHz Core 2 T7100
Typ OS	32bitový	32bitový
Typ HW	Acer TravelMate 2414WLMI	DELL Latitude D830

Tab.3: Hardware použitých systémů

5.3 Jak provádět útok hrubou silou v systémech Windows?

Nejdříve napíšeme jednoduchý program nejlépe v jazyce C++ nebo v některém z jazyků nad platformou .NET. Program musí co nejrychleji ověřovat uživatelské jméno a heslo. Následující ukázka kódu ověřuje uživatele „admin“ z domény „nazevdomeny“. Z důvodu možného zneužití uvádím pouze použití číselné abecedy o 10 znacích. Test bude ukončen 1000 pokusem. Program je napsán v jazyce C#. Tento příklad nám postačí jako základ pro zjištění odolnosti operačních systémů. Komentáře k programu jsou vloženy přímo v kódu.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Runtime.InteropServices;
using System.Security.Principal;


namespace logon
{
    class Program
    {
/* deklarace funkcí z API*/
[DllImport("advapi32.dll", SetLastError = true)]
        private static extern int LogonUser(string username, string
domena, string heslo, int logontype, int logonProvider, out int phToken);

[DllImport("kernel32.dll")]
        private static extern int FormatMessage(int dwFlags, string
lpSource, int dwMessageId, int dwLanguageId, StringBuilder lpBuffer, int
nSize, string[] Arguments);
/* konstatnty z API*/

        private const int LOGON32_LOGON_NETWORK_CLEARTEXT = 3;
        private const int LOGON32_PROVIDER_DEFAULT = 0;
```

```
        private const int FORMAT_MESSAGE_FROM_SYSTEM = 0x1000;

        private static WindowsImpersonationContext
winImpersonationContext = null;
        public static void ImpersonateUser(string domena, string
username, string heslo)
        {
            int userToken = 0;
            bool login =
(LogonUser(username,domena,heslo,LOGON32_LOGON_NETWORK_CLEARTEXT,LOGON32_
PROVIDER_DEFAULT,out userToken)!=0);
            if (login == false)
            {
                //Console.WriteLine("špatné heslo");

            }
            if (login == true)
            {
                Console.WriteLine("Heslo uzivatele {1} je....
{0}",heslo,username);


            }

        }

        static void Main(string[] args)
        {
            string domena = "nazevdomeny";
            string user = "admin";
            string heslo = "0";
            double pocet= 10000;

            Console.WriteLine("Start");
/* vlastní útok, použití pouze čísel, před ověřením převod na string*/

            for (double i = 0; i < pocet; i++)
            {
                heslo = Convert.ToString(i);
                ImpersonateUser(domena, user, heslo);
            }
            Console.WriteLine("Konec");
            Console.ReadLine();
```

V tomto programu byly použity části kódu z knihy [1] .

Jistě je možné program napsat tak, aby odhlásil uživatele a znovu přihlásil. Tyto metody jsou ale velmi zdlouhavé. Ptal jsem se na technické podpoře Microsoftu, zda je možné identifikaci uživatele ověřit rychlejším způsobem. Odpověď byla, že výše uvedená metoda je dostatečně rychlá pro jakoukoli aplikaci. Jedná se o funkci LogonUser z API Windows.

5.4 Výsledky měření času

Jako vzorek jsem použil vždy 1 000 000 pokusů. Pro Windows Vista jsem dosahoval rychlosti 1 mil pokusů za 4,5 minut. Za jednu hodinu můžeme ověřit 11,5 milionů hesel. Je nutné si uvědomit, že se jedná vždy o maximální hodnotu. Klidně se to může podařit za polovinu času. Vždy záleží, jak poskládáme abecedu znaků. Pokud například uživatel má heslo „admin“. A náš útok začne heslem „aaaaa.. aaaab.. aaaac..“ Máme prostě štěstí. V tomto případě se jedná v podstatě o hádání 4 znakového hesla. Samozřejmě vždy platí teorie o použití abecedy a délky hesla. Při jednotlivých měřeních nebyly na operačních systémech provozovány žádné další aplikace. Zajímavostí bylo pouze vytížení procesoru na 98%. U vícejádrových procesorů se vyplatí z hlediska úspory strojového času, hádání hesel rozdělit na více aplikací případně na více vláken přímo v programu. Následující tabulka uvádí, jak rychle se podaří heslo rozluštit. Nezapomínejme, že se jedná o běžný desktopový systém (notebook DELL). Na výkonných serverech můžou být hodnoty násobně rychlejší. Jak vidíme při použití standardní abecedy s malými písmeny (26 znaků) uhádneme 5 znakové heslo za cca 1 hodinu. O abecedách s menším počtem znaků se snad ani bavit nemusíme. Stejnému testu rychlosti jsem podrobil i systém Windows XP. Naměřené hodnoty byly téměř totožné s Windows Vista. Zajímavé je že systém s Windows XP je nainstalován na míň výkonném hardwaru. Z čistě technického hlediska by měl test rychlosti trvat déle. Tento jev si vysvětluji snad jenom tím, že volání API fcí z Windows Vista je pomalejší. Microsoft tento poznatek opět odmítl komentovat, s tím že volání „LogonUser“ vyhovuje všem aplikacím. Je možné, že po vydání service packu bude výsledek o něco lepší. V tabulce jsou tedy shrnuty hodnoty pro oba operační systémy. Čas je uváděn v hodinách.

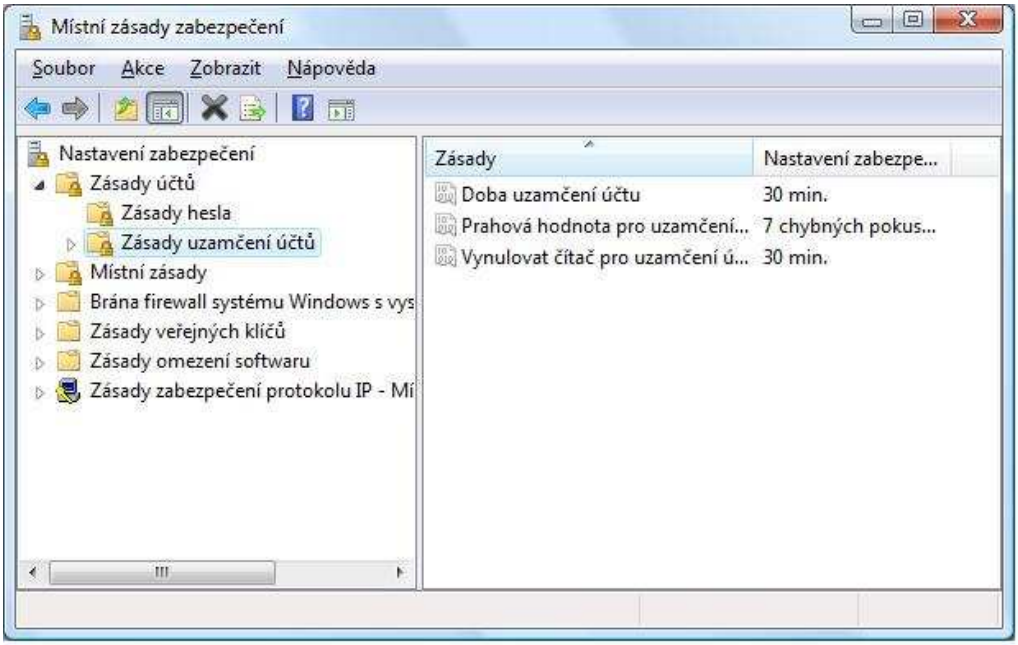
Čas potřebný k nalezení hesla při rychlosti						**11 500 000**	**za hodinu**
délka hesla/ velikost abecedy	2	3	4	5	6	7	
10 znaků	0	0	0	0	0	1	
26 znaků	0	0	0	1	27	698	
36 znaků	0	0	0	5	189	6 814	
52 znaků	0	0	1	33	1 719	89 398	
62 znaků	0	0	1	80	4 939	306 227	
86 znaků	0	0	5	409	35 180	3 025 459	
96 znaků	0	0	7	709	68 066	6 534 326	

Tab.4: Výsledky naměřeného času při útoku hrubou silou

5.5 Zabezpečení operačních systémů Microsoft

Microsoft Windows Vista Business i Windows XP jsou proti útokům hrubou silou zabezpečeny stejným způsobem. Musíme otevřít okno „Mistní zásady zabezpečení“. Ve Windows XP i Windows Vista ho najedeme postupně Ovládací panely – Nástroje pro správu – Místní zásady zabezpečení. V menu Zásady účtů najdeme následující položky.

Zásady hesla a Zásady uzamčení účtu



Obr.1: Zabezpečení Windows Vista

5.5.1 Heslo musí splňovat požadavky na složitost

Nesmí obsahovat název účtu uživatele ani části celého jména uživatele přesahující dva sousední znaky.

Musí obsahovat alespoň 6 znaků.

Musí obsahovat znaky ze tří z následujících čtyř kategorií:

velká písmena anglické abecedy (A až Z),

malá písmena anglické abecedy (a až z),

základních 10 číslic (0 až 9),

jiné než alfanumerické znaky (například !, $, #, %).

Požadavky na složitost jsou prosazovány při změně nebo vytvoření hesla.

Z našeho pohledu se jedná o abecedy s počtem minimálně 62 znaků. Při délce 6 znaků se dostáváme na 57 miliard kombinací. Budeme-li heslo hádat naší metodou,pak potřebujeme 4 939 hodin. To je téměř 7 měsíců.

5.5.2 Maximální stáří hesla

Toto nastavení zabezpečení určuje dobu (ve dnech), po kterou může být heslo používáno, než systém požádá uživatele o jeho změnu. Zadáním hodnoty v rozsahu 1 až 999 určíte počet dnů, po jejichž uplynutí má vypršet platnost hesla, zadáním hodnoty 0 určíte trvalou platnost hesla. Pokud je maximální stáří hesla mezi 1 a 999 dny, musí být minimální stáří hesla menší než tato hodnota. Pokud je maximální stáří hesla nastaveno na hodnotu 0, lze minimální stáří hesla nastavit na libovolnou hodnotu mezi 0 a 998 dny. Výchozí hodnota: 42.

5.5.3 Minimální délka hesla

Toto nastavení zabezpečení určuje nejmenší počet znaků, které musí heslo uživatele obsahovat. Lze nastavit hodnotu mezi 1 a 14 znaky, případně nastavením hodnoty 0 určit, že není požadováno žádné heslo.

5.5.4 Minimální stáří hesla

Toto nastavení zabezpečení určuje dobu (ve dnech), po kterou musí být heslo používáno, než je uživatel může změnit. Můžete nastavit hodnotu mezi 1 a 998 dny, nastavením hodnoty 0 umožníte bezprostřední změnu hesla.

Minimální stáří hesla musí být menší než maximální stáří hesla, pokud není maximální stáří hesla nastaveno na hodnotu 0, která značí trvalou platnost hesel. Pokud je maximální stáří hesla nastaveno na hodnotu 0, lze minimální stáří hesla nastavit na libovolnou hodnotu mezi 0 a 998.

5.5.5 Ukládat hesla pomocí reverzibilního šifrování

Toto nastavení zabezpečení určuje, zda operační systém ukládá hesla pomocí reverzibilního šifrování.

5.5.6 Vynutit použití historie hesel

Toto nastavení zabezpečení určuje počet jedinečných nových hesel, která musí být přidružena k uživatelskému účtu, než lze znovu použít některé staré heslo. Hodnota musí být mezi 0 a 24 hesly.

Tato zásada umožňuje správcům zvýšit zabezpečení tím, že zabraňuje opětovnému používání starých hesel.

5.5.7 Doba uzamčení účtu

Toto nastavení zabezpečení určuje počet minut, po které zůstává uzamčený účet uzamčen, než je opět automaticky odemčen. Lze nastavit dobu v rozsahu 0 až 99 999 minut. Při nastavení hodnoty 0 bude účet uzamčen tak dlouho, dokud jej správce explicitně neodemkne.

Pokud je definována prahová hodnota pro uzamčení účtu, musí být doba uzamčení účtu větší nebo stejná jako doba obnovení.

5.5.8 Prahová hodnota pro uzamknutí účtu

Toto nastavení zabezpečení určuje počet neúspěšných pokusů o přihlášení, po kterém je uživatelský účet uzamčen. Uzamčený účet není možné použít, dokud ho správce neodemkne nebo dokud nevyprší doba trvání uzamčení účtu. Můžete nastavit hodnotu mezi 0 a 999 neúspěšnými pokusy o přihlášení. Pokud nastavíte hodnotu 0, účet nebude uzamčen nikdy.

Jako neúspěšné pokusy o přihlášení se počítají i neúspěšné pokusy o zadání hesla na pracovních stanicích či členských serverech, které byly uzamčeny stisknutím kombinace kláves Ctrl+Alt+Delete nebo spořičem obrazovky chráněným heslem.

5.5.9 Vynulovat čítač uzamčení účtu po

Toto nastavení zabezpečení určuje počet minut, které musí po neúspěšném pokusu o přihlášení uplynout, než se čítač neúspěšných pokusů o přihlášení znovu nastaví na hodnotu 0. Možný rozsah je od 1 do 99 999 minut.

Tato doba vynulování musí být menší nebo rovna prahové hodnotě pro uzamknutí účtu, je-li definována.

5.6 Doporučené zabezpečení

Když použijeme zásadu 5.5.1 (složitost hesla) společně se zásadou 5.5.8 (uzamknutí účtu při neúspěšných pokusech). Potom se stává Váš operační téměř nedobytný útokem hrubou silou. Navíc je nutné definovat dobu uzamčení účtu. Vše v závislosti na neúspěšných pokusech.

5.7 Doba uzamčení účtu je obrana i útok

Vše vysvětlím na následujícím příkladu. S operačním systémem pracuje administrátor i běžný uživatel. Náš administrátor bude precizní a nastaví prahovou hodnotu uzamčení účtu např. na 100 pokusů. Zároveň nastaví dobu uzamčení účtu na 60 minut. Běžný uživatel se rozhodne účet administrátora atakovat hrubou silou. Po 100 pokusech sice heslo neuhádne, ale zamkne účet administrátora na 60 minut. Kdo se stává v tomto případě vítězem a kdo poraženým nechám na posouzení každého správce systému. Uvědomme si prosím, že tímto způsobem můžeme výrazně obtěžovat jak jiné uživatele systému, tak i administrátory. Z toho důvodu nevím, nakolik je moudré používat zamčení účtu při neúspěšném přihlášení.

6 JEDNODUCHÝ ÚTOK V PROSTŘEDÍ MICROSOFT

6.1 Při použití Active Directory

„… pracujete ve firmě jako správce sítě. Jako server používáte Windows 2003 Server. Ctíte doporučenou bezpečnostní politiku firmy Microsoft. Máte tedy nastavená silná hesla o délce minimálně 6 znaků. Dále nastavíme zamčení uživatelského účtu po 20-ti neúspěšných pokusech zadání hesla. Snažíme se tímto zabránit „útokům hrubou silou“. Dle doporučení používáte síť s doménou a Active Directory. Ve firmě máme útočníka, který napíše jednoduchý program. Tento program bude atakovat účet administrátora. Po 20-ti neúspěšných pokusech se účet administrátora zamkne. Heslo sice nezjistil, ale administrátor se do sítě nepřihlásí dřív než za nastavenou dobu. Což může být např. 10 minut. Není nic jednoduššího než takový program nechat spuštěný. Ani administrátor se v tuto chvíli nepřihlásí. O co horší bude, pokud si útočník zjistí uživatelské jména všech uživatelů v Active Directory (viz příloha PI) a daný útok spustí na všechny účty. V tuto chvíli ochromí činnost celé sítě. Následky jsou katastrofální …“

Není snad i toto útok HRUBOU SILOU!!! A to vše jako běžný uživatel.

ZÁVĚR

Útoky hrubou silou jsou pravděpodobně nejsnadnější metodou, jak získat přístup do systému. Přesto je jejich hrozba vysoce aktuální. Musíme si uvědomit, že se výpočetní výkon našich systému neustále zvyšuje. Přidáme-li možnost paralelizace ať formou vícejádrových procesorů nebo spuštěním na více systémech. Stává se útok hrubou silou skutečně reálným. Když náš systém žádným způsobem nezabezpečíme, vystavujeme ho velkému riziku. Uživatelé, kteří používají pouze 5 znakové hesla, skutečně riskují odhalení hesla. Používat silná hesla je tedy víc než nutné.

Pokud se týká praktického výzkumu Windows XP a Windows Vista, musím konstatovat, že firma Microsoft v tomto směru žádný pokrok neudělala. Zabezpečení na úrovni zamčení účtu po několika neúspěšných pokusech není dostatečné. A může být zneužito.

ZÁVĚR V ANGLIČTINĚ

The brutal force attacks are probably the easiest method how to get access to the information system. Although their dangeris too current now. We have to realize that our systems computing power is continually raised and if we addthe paralelization possibility either by multi-core processors form or by running an application on some more systems, then the brutal force attack becomes real. If we do not provide access to our system in any way, it is exposed to a high risk.Users who use only five-sign password take a high risk of its find out. Using of special characterised passwords is more than neceséry. As far as practical research of Windows XP and Windows Vista ,I have to claim that Microsoft company did not make any progess in this area.The protection ,at the locking account level after some unsuccesful attempts, is not sufficient and can be misused.

SEZNAM POUŽITÉ LITERATURY

[1] JURGEN ,BAYER, C# 2005 Velká kniha řešení Nakl. Computer Press, a, s. Brno, 2007. ISBN 978-80-251-1620-3

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

Active Directory	Adresářová služba systému Windows, která ukládá informace o prostředcích (např. tiskárnách, uživatelích, počítačích a skupinách) a poskytuje uživatelům a jiným počítačům přístup k těmto informacím.
.NET	Vývojová platforma firmy Microsoft.
C#	Programovací jazyk pro .NET platformu vycházející z C++.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: PROGRAM PRO VÝPIS UŽIVAZTELŮ – C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.DirectoryServices;

namespace user1


        {
    public class PrintChildren{
   public static void Main(String[] args)
   {
      DirectoryEntry objDE;

      String strPath = "WinNT://nazev-pc-nebo-domeny";

       if(args.Length>0)strPath=args[1];

      // Create a new DirectoryEntry with the given path.
      objDE=new DirectoryEntry(strPath);

      foreach (DirectoryEntry objChildDE in objDE.Children)
      {

          if (objChildDE.SchemaClassName == "User")
Console.WriteLine("Název..{0} Typ.. {1}",
objChildDE.Name,objChildDE.SchemaClassName);
      }
          Console.ReadLine();
          Console.WriteLine("Konec");



          Console.ReadLine();
   }


    }
```