Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Útoky hrubou silou na Truecrypt

Diplomová Práca

2016 **Martin Strapko**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Útoky hrubou silou na Truecrypt

Diplomová Práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra Informatiky
Vedúci práce:	RNDr. Richard Ostertág PhD.

2016 **Martin Strapko**

77900676

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Martin Strapko
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský

Názov: Systém automatickej distribúcie softvéru na pracovné stanice s OS Windows 7 / *Automatic Software Distribution System for Windows 7 Workstations*

Cieľ: Cieľom práce je analyzovať možnosti automatizovanej inštalácie softvéru, konfigurácie a vykonávania definovaných úloh na pracovných staniciach s Windows 7 a navrhnúť a implementovať systém, ktorý umožní automatickú inštaláciu softvéru na základe vzorovej inštalácie na jednom počítači. Systém má umožniť inštalovať softvér "na požiadanie", ako aj automaticky po štarte počítača. Zároveň má umožniť centrálne nastavovanie konfiguračných parametrov a vykonávanie definovaných úloh. Zmyslom výsledného diela je automatizácia práce administrátora veľkého počtu pracovných staníc. Pri návrhu a implementácii systému je potrebné zohľadniť aj požiadavky na bezpečnosť.

Vedúci: RNDr. Jaroslav Janáček, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: doc. RNDr. Daniel Olejár, PhD.

Dátum zadania: 19.10.2012

Dátum schválenia: 21.10.2013 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie

Abstrakt

Cieľom tejto práce je implementovať jednoduchú aplikáciu na automatizáciu hromadného inštalovania staníc s operačným systémom Windows 7. Snahou je taktiež implementovať funkcionalitu inštalácie balíku na vyžiadanie, pre používateľa bez potreby administrátorských práv.

KĽÚČOVÉ SLOVÁ: Windows, .NET framework, HTTPS, registre, počúvanie udalostí

Abstract

The main purpose of this thesis is to implement a simple application that automate mass installation on stations with Windows 7. We also attempt to implement a function for on-demand package installation, to be available even for a user without privileged access rights.

KEYWORDS: Windows, .NET framework, HTTPS, registry, event listening

Obsah

Zoznam listingov

Zoznam obrázkov

Úvod

V dnešnom svete fungujúcom na elektronických dátach, ktoré pre nás majú obrovskú cenu, sa ľudia snažia udržať ich čo najviac v tajnosti. Za týmto účelom vznikli mnohé programy slúžiace na zašifrovanie dát pomocou používateľského hesla. Vzhľadom na rýchly vzrast výpočtovej sily počítačov tieto používateľské heslá sa začali predlžovať a komplikovať. Zatiaľ čo vďaka tomuto trendu sa zvýšili bezpečnosť dát, heslá sa stali na toľko komplikované, že boli ťažko zapamätateľné. Zatiaľ čo spoločnosti poskytujúce služby využívajúce zaheslované dáta dokážu potenciálne vyresetovať používateľovi heslo aby mal možnosť zvoliť si nové, zašifrovaný disk na domácom počítači takúto možnosť nemá. Práve tento problém sa snažíme adresovať v tejto práci.

KAPITOLA 1

TrueCrypt

TrueCrypt je šifrovací program poskytujúci používateľovi možnosť zašifrovať ľubovoľnú časť disku v počítači pomocou používateľom zvoleného hesla. Vývoj tohto programu bol ukončený roku 2014 a podľa autorov nie je bezpečný, nakoľko jeho implementácia môže obsahovať bezpečnostné chyby. Cieľom tejto práce nie je odhalenie týchto chýb, nakoľko sa nesnažíme zlomiť toto šifrovanie. Našim cieľom je implementácia algoritmu schopného zistiť stratené používateľské heslo za účelom odšifrovania dát ich majiteľom. Aj napriek tomuto cieľu budeme naše pokusy o nájdenie heslá nazývať útoky na šifrovanie. V nasledujúcich kapitolách práce bližšie popíšeme algoritmus šifrovania disku pomocou programu TrueCrypt a následne podrobne rozoberieme možnosti útokov na používateľské heslá. V druhej polovici práci sa budeme venovať nami implementovanému algoritmu a jeho fungovaniu. Prácu ukončíme odsekom popisujúcim nami získane výsledky.

Ako sme spomínali v úvode program TrueCrypt slúži na zašifrovanie dát na používateľskom disku pomocou zvoleného hesla. Tento program má implementované viaceré šifrovacie algoritmy, avšak predpokladáme, že poznáme algoritmus zvolený na zašifrovanie cieľového disku a na základe toho ďalej v práci budeme pracovať s algoritmom AES-256. Zašifrovanie disku pomocou tohto programu prebieha v dvoch fázach. Najskôr vygeneruje náhodný kľúč a pomocou neho zašifruje disk aj s dátami. Následne zapíše konfiguráciu a vygenerované kľúče do hlavičky zašifrovaného disku a tú následne zašifruje kľúčom vygenerovaným pomocou používateľského hesla. Dôležitou informáciou o programe TrueCrypt je že zdrojové kódy tohto programu sú voľne dostupné na internete.

KAPITOLA 2

Útoky na šifrovanie

V nasledovnej kapitole popíšeme možnosti útokov na používateľské heslá. Všetky nižšie vymenované typy útokov patria spoločne do kategórie útokov hrubou silou, keďže ich cieľom je nájsť používateľské heslo pomocou prehľadávania všetkých možností.

2.1 Inkrementálna metóda

Inkrementálna metóda je pravdepodobne najintuitívnejší útok hrubou silou. Pri tomto type útoku útočník generuje všetky reťazce vstupnej abecedy kratšie ako stanovená maximálna dĺžka hľadaného hesla. Ak poznáme túto maximálnu dĺžku hľadaného hesla, tak tento prístup ma 100 percentnú úspešnosť keďže prehľadá celý priestor možných hesiel. Avšak týchto hesiel je <strasne vela, treba dosadit vzorec>, čo by pri skúšaní <pekna konstanta> hesiel za sekundu trvalo dokopy <ohurujuce cislo> rokov.

2.2 Slovníkový útok

Iná možnosť ako skúšať všetky možné kombinácie znakov je skúšať heslá, ktoré sú často používane alebo majú nejaký konkrétny význam pre používateľa. Z týchto slov vystaviame slovník pomocou ktorého následne skúšame či sa nám podarilo nájsť správne heslo. Keďže slovník použitý pri útoku sami pripravíme vyskúšanie všetky hesiel, ktoré obsahuje, bude uskutočniteľné v rozumne krátkom čase. Vďaka tomuto patrí medzi najpopulárnejšie metódy útoku. Avšak úspešnosť tohto útoku je závislá od hesiel, ktoré pridáme do slovníku. V tomto probléme nám môže pomôcť fakt, že hľadáme stratené heslo, čiže používateľ dokáže poskytnúť veľké množstvo potenciálnych hesiel, ktoré majú vysokú pravdepodobnosť úspechu.

2.2.1 Prekrúcanie slov

Avšak samotný používateľ nemusí vedieť hľadané heslo, ktoré by sa mohlo líšiť od niektorého v slovníku len v jednom znaku, napríklad vo veľkom alebo malom písmene. Preto sa často so slovníkovým útokom používa takzvané prekrúcanie slov, ktoré pomocou predom definovaných pravidiel skúsi znetvoriť postupne každé heslo zo slovníka. Týmto výrazne zväčší veľkosť slovníka čím zvýši šancu na úspech a zmizne potreba ručne generovať stovky tisíc hesiel.

2.3 Hybridný útok

Metóda, ktorou sa zaoberáme v tejto práci a ktorú sme implementovali je hybrid vytvorený zo všetkých vyššie uvedených. Našim hlavným cieľom je nájsť hľadané heslo v konečnom čase. Preto sa budeme snažiť vytvoriť algoritmus, ktorý vygeneruje všetky možné reťazce kratšie ako zadaná maximálna dĺžka, avšak aby sme tento čas minimalizovali tak pomocou slovníka, obsahujúceho známe často používané heslá a heslá o ktorých si používateľ myslí, že by mohli byť hľadaným heslom. Naštudujeme si štruktúru hesiel aké používateľ používa. Z týchto znalostí si vytvoríme pravidlá pomocou ktorých budeme generovať naše finálne pokusy na odšifrovanie cieľového disku.

KAPITOLA 3

Učenie

Ako sme spomínali vyššie v texte, náš program bude generovať heslá na základe nejakých znalostí. Tento proces učenia sa, sa dá implementovať pomocou viacerých známych metodík medzi, ktoré patria napríklad neurónové siete alebo pravdepodobnostné gramatiky.

3.1 Neuronové siete

3.2 Pravdepodobnostné bezkontextové gramatiky

Bezkontextové gramatiky používajú pravidlá pri ktorých sa neterminál môže zmeniť na ľubovoľnú vetnú formu bez ohľadu na kontext v ktorom sa nachádza. Pravdepodobnostné gramatiky vzniknú keď každému pravidlu priradíme číslo v rozmedzí 0 a 1. Toto číslo vyjadruje pravdepodobnosť použitia daného pravidla pre jeho neterminál, súčet pravdepodobností jedného neterminálu by mal byť 1. Následne pravdepodobnosť terminálnej vetnej formy je rovná súčinu pravdepodobností pravidiel použitých v jej odvodení. V gramatikách je možné a častokrát až bežné aby jedna terminálna vetná forma mala viacero stromov odvodenia, poradí použitia pravidiel. My sme sa tejto vlastnosti snažili vyhnúť, pretože by sme zbytočne generovali jedno heslo viac krát.

KAPITOLA 4

Implementácia

4.1 Tvorba gramatiky

Pri tvorbe gramatiky sme potrebovali zaistiť aby gramatika spĺňala určité podmienky. Prvou z nich je schopnosť gramatiky vygenerovať všetky reťazce zo vstupnej abecedy kratšie ako používateľom zadaná maximálna dĺžka. Druhou podmienkou je aby každé terminálne slovo, ktoré gramatika generuje malo práve jeden strom odvodenia.

Jednodché neterminály Prvý typ neterminálov, ktoré budeme nazývať jednoduché, obsahuje pravidlá na zterminalnenie generovaného slova. Keďže naša vstupná abeceda obsahuje okolo 70 znakov, medzi ne patria veľke a malé písmena, cifry a niektoré často používané symboly, rozhodli sme sa ich rozdeliť do jednotlivých skupín. Pre každú z týchto skupín sme vytvorili neterminál, ktorý bude reprezentovať sekvenciu pevnej dĺžky zloženú zo znakov danej skupiny. V gramatike tieto neterminály vyjadrujeme pomocou prvého písmena anglického názvu danej skupiny.

- U - upper case - veľké písmena
- L - lower case - malé písmena
- D - digit - cifry
- S - symbol - symboly

Každý jednoduchý neterminál sa teda skladá z písmena vyjadrujúceho skupinu znakov, ktoré generuje, a čísla popisujúceho dĺžku sekvencie na pravej strane pravidiel tohto neterminálu. Keďže generovanie všetkých variacií veľkostí k pri n prvkoch môže byť obrovské množstvo rozhodli sme sa zadefinovať maximálnu dĺžku sekvencie generovanej jednoduchým neterminálom.

Zložené neterminály Jednoduché neterminály nám pomáhajú vyjadrovať sekvenciu znakov práve jedného z vyššie vymenovaných typov. Aby sme boli schopný popísať ľubovolný reťazec tvorený znakmi vstupnej abecedy, budeme tieto jednoduché neterminály skladať do skupín, zložených neterminálov. Tieto neterminály vyjadrujú vzdy jeden možný predpis pre terminálne slovo. Napríklad neterminál *U1L3D4* vyjadruje všetky terminálne slová začínajúce na veľké písmeno nasledované tromi malými písmena, ukončené štvoricou cifier.

Ďalej taktiež nedovoľujeme aby sa vyskytovali 2 jednoduché neterminály rovnakého typu za sebou. V prípade, že potrebujeme popísať sekvenciu terminálnych znakov jedného typu dlhšiu ako povolené maximum (popísane vyššie), rozdelíme túto sekvenciu do viacerých jednoduchých neterminálov pažravým algoritmom, čiže každý z týchto neterminálov zobere maximálný možný počet znakov sekvencie. Tento spôsob nám zaručí, že nevzniknú dva rôzne zložené neterminály vyjadrujúce ten istý predpis terminálneho slova.

Počiatočný neterminál gramatiky Z bude obsahovať pravidlá prepisujúce tento neterminál na niektorý zo zložených alebo jednoduchých neterminálov. Tento spôsob generovania gramatiky spĺňa obe pravidlá, ktoré sme popisovali v úvode tejto kapitoly.

4.2 Počítanie pravdepodobností

Ako sme spomínali v úvode textu, nami generované pokusy o nájdenie hesla chceme prispôsobit potrebám jednotlivých používateľom, ktorí sa snažia získať svoje stratené heslo. Aby sme vedeli čo najlepsie vyhovieť týmto používateľom, potrebujeme upraviť našu gramatiku. Tu prichádzajú do pozornosti pravdepodobností jednotlivých pravidiel našej gramatiky. Našim cieľom je nastaviť našu gramatiku tak aby generovala heslá podľa pravdepodobnosti použitia daným používateľom. Úspešnosť tohto učenia gramatiky bude drastický záležať od kvality vstupných dát.

Vzhľadom na to, že v dnešnom svete používatelia používajú rôzne služby, ktoré každá odporúča mať jedinečné heslo, používatelia používajú niekoľko hesiel naraz. Tieto heslá by si radi všetky pamätali a preto si často vytvoria pre seba charakteristický spôsob tvorby a zapamätania si týchto hesiel. V ideálnom prípade by sme chceli aby naše vstupné dáta pozostávali z čo najväčšieho počtu hesiel vytvorených pomocou tohto charakteristického spôsobu, keďže každé upresnenie informácií o hľadanom hesle nám zvýši rýchlosť nájdenia tohto hesla.

Keďže cieľom našej práce je nájsť heslo so 100% pravdepodobnosťou, čo v najhoršom prípade znamená vygenerovať všetky možné reťazce kratšie ako zadaná maximálna

dĺžka hesla, tak pravidla našej gramatiky sa budú meniť len pri zmene maximálnej dĺžky hesla. V ostatných prípadoch sa budú meniť len ich pravdepodobností. Pravdepodobností terminálných sekvencií budeme rátať ako percento výskytov danej terminálnej sekvencie spomedzi všetkých sekvencií spadajúcich pod tento neterminál. Práve kvôli tomuto spôsobu sme pridali v implementácií možnosť napísať do vstupného slovníku počty výskytov jednotlivých hesiel, aby mal používateľ možnosť zdôrazniť dôležitosť hesla. Pri počítaní pravdepodobností zložených neterminálov máme viacero možností ako postupovať.

Priamo zo vstupného slovníka Prvý spôsob ako postupovať bol identický s tým pre jednoduché neterminály. Pre každe pravidlo gramatiky prepisujúce neterminál Z na zvolený zložený neterminál vypočítame jeho pravdepodobnosť ako pomer počtu výskytov tohto neterminálu a všetkých výskytov. Tento spôsob môže mať ešte 2 varianty.

- Do výskytov počítame len výskyty hesiel ktoré sú presne reprezentované daným neterminálom
- Do výskytov započítame aj výskyty kedy je zvolený neterminál podreťazcom iného neterminálu

Rekurzívne Ďalší spôsob spočíva v tom, že zo vstupného slovníka vypočítame pravdepodobností len pre jednoduché netermiály. Následne pre zložené neterminály počítame pravdepodobnosti ako súčin pravedpodobností jednoduchých neterminálov, ktoré daný neterminál obsahuje.

Všetky vyššie spomenuté metódy na počítanie pravdepodobností pravidiel gramatiky sme implementovali. Ich vzájomne porovnanie ako aj porovnanie s inými bežne používanými programami je vidieť v kapitole Výsledky.

KAPITOLA 5

Testy

Cieľom tejto kapitoly je podrobne popisať testy, ktoré slúžia k vyhodnoteniu efektivity a správanosti nášho algoritmu voči iným bežne dostupným riešeniam.

5.1 Časové testy

V teste budeme rátať čas behu jednotlivých programov od začiatku spracovania vstupného slovníka, až po vygenerovanie predom stanoveného počtu hesiel. Pri jednorázovom generovaní určitého počtu hesiel by naše riešenie mohlo byť trochu pomalšie ako iné bežne používané riešenie. Avšak v prípade, že budeme generovať heslá s rovankými parametrami rozdelené do viacerých osobitných požiadaviek, náš program by mohol byť trochu rýchlejší vďaka tomu, že nebude musieť spracovávať opäť vstupné dáta.

Vo viacerých spusteniach tohto testu sme menili premenné *maximálna dĺžka hesla*, *počet generovaných hesiel*, *počet opakovaných generovaní hesiel*, *maximálna dĺžka neterminálu v gramatike*. Taktiež budeme každý test spúšťať viac krát a vo výsledkoch uvedieme aritmetický priemer týchto hodnôt. Výsledne tabuľky a grafy sa nachádzajú v sekcií výsledky.

5.2 Test správnosti

Hlavným cieľom našej práce je najsť zabudnuté používateľské heslo v čo najkratšom čase. Toto sa dá dosiahnuť dvoma spôsobmi. Jeden z nich je optimalizácia kódu zodpovedného za overovanie či sme našli správne heslo. Druhý, ktorému sme sa v tejto práci venovali, je optimalizácia poradia generovania hesiel, ktoré chceme vyskúšať. Práve preto nám záleži aby nami sme minimalizovali počet hesiel, ktoré musíme vyskúšať než nájdeme to správne. Avšak toto môže veľmi záležať od prípadu k prípadu, preto bu-

deme náš program porovnávať s ostatnými bežne dostupnými riešeniami a ich spôsobmi generovania hesiel.

Vrámci testu necháme všetky programy vygenerovať predom stanovené množstvo hesiel zo zadaného slovníka. Tento slovník bude pozostávať z hesiel zoradených podľa počtu použití a bude identický pre všetky testované programy. Následne výsledne zoznamy hesiel porovnáme oproti pôvodnému slovníku. Ako jednu z hlavných metrík pri rozhodovaní o kvalite zoznamu hesiel budeme brať *štandardnú odchýlku* indexov hesiel oproti pôvodnému slovníku. Taktiež budeme sledovať iné miery kvality «TODO».

KAPITOLA 6

Výsledky

V tejto časti budú prezentované výsledky z testov popísaných v kapitole Testy. Ku každému testu bude tabuľka, graf a krátky popis vysvetľuúci čo sa z daných dát dá pochopiť. Zatiaľ bohužial kvôli problémom, ktoré sa naskytli pri spúšťaní testov je to tu prázdne

Literatúra

[1] Matt Davis. Návod na inštaláciu služby - http://stackoverflow.com/questions/1195478, Január 2014.

[2] Microsoft. Trieda použitá na detekovanie zmien v systéme - http://msdn.microsoft.com/en-us/library/system.io.filesystemwatcher.aspx, November 2013.

[3] Microsoft. Windows registry informácie - http://support.microsoft.com/kb/310516, Január 2014.

[4] Networkworld. Porovnanie existujúcich riešení - http://www.networkworld.com/reviews/2010/112210-netresults.html, Marec 2014.

[5] Michael Potter. Algoritmus použitý na diferenciu súborov - http://www.codeproject.com/Articles/6943/, Január 2014.

[6] John Vekal. Chyba so side-by-side konfiguráciou - http://www.codeproject.com/Articles/43681/, Máj 2014.