Зборник радова Факултета техничких наука, Нови Сад

UDK: 004.633
DOI: https://doi.org/10.24867/28BE12Kureljusic

АЛАТ ЗА ДЕКРИПЦИЈУ ZIP АРХИВА ПРИМЕНОМ NVIDIA CUDA ТЕХНОЛОГИЈЕ

ZIP ARCHIVE PASSWORD RECOVERY TOOL USING NVIDIA CUDA TECHNOLOGY

Лука Курељушић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИЧКО И РАЧУНАРСКО ИНЖЕЊЕРСТВО

Кратак садржај – *У овом раду описан је развој алата за опоравак лозинке ZIP архива WinZip формата, енкриптованих помоћу AES алгоритма. Централна идеја алата обухвата паралелизацију напада грубом силом на ZIP архиву употребом једне или више графичких картица. Структура предложеног алата је модуларна и састоји се од фронтенд дела који обавља поступке парсирања архиве и даљег процесирања улазних информација и бекенд дела, имплементираног помоћу NVIDIA CUDA технологије који се извршава на графичкој картици и обухвата паралелизацију PBKDF2 алгоритма деривације кључа и накнадне процесе верификације добијеног кључа над улазним подацима. Алат остварује задовољавајуће перформансе за нумеричке и краће алфанумеричке шифре, док присуство специјалних карактера значајно отежава успех алата.*

Кључне речи: *ZIP архива, декрипција, опоравак лозинке, паралелизација, CUDA*

Abstract – *This paper presents the development of a password recovery tool for WinZip format ZIP archives encrypted using the AES algorithm. The central idea of the tool involves parallelizing a brute-force attack on a ZIP archive using one or more graphics cards. The structure of the proposed tool is modular and consists of a frontend part that performs archive parsing and further processing of input information, and a backend part, implemented using NVIDIA CUDA technology, which is executed on a graphics card and includes the parallelization of the PBKDF2 key derivation algorithm and the subsequent processes of verification of the obtained key over input data. Tool achieves satisfactory performance for numeric and shorter alphanumeric codes, while the presence of special characters significantly hinders the success of the tool.*

Keywords: *ZIP archive, decryption, password recovery, parallelization, CUDA*

1. УВОД

У ери у којој безбедност података и приватност постају најзначајнији аспекти информационих система, употреба лозинком заштићених архива, тиме и ZIP датотека постаје све чешћа.

Мотивација за развој алата за опоравак лозинке *ZIP* датотеке је двојака. Са једне стране алат налази своју употребну вредност у практичним изазовима са којима се суочавају појединци и организације када изгубе приступ архивама заштићеним лозинком. Са друге стране, идеја алата јесте и допринос пољу дигиталне безбедности, где се алат може употребити за испитивања безбедности делова информационих система који се односе на рад са ZIP архивама.

За потребе рада развијен је алат базиран на *NVIDIA CUDA* технологији, која представља платформу за имплементацију концепта паралелизације заснованог на употреби графичких картица. Сам поступак разбијања шифре подразумева итеративни процес генерисања и потоње провере кандидата шифре методом грубе силе над претходно (кориснички) дефинисаним простором претраге који представља рестрикцију над скупом *ASCII* карактера.

У овом раду најпре ће бити извршен преглед стања у области уз осврт на актуелна најсавременија решења отвореног кода која се баве датом проблематиком. Потом ће бити описана предложена архитектура и спецификација захтева алата након којих ће бити представљени најзначајнији имплементациони детаљи и примењене технике оптимизације. Потом ће бити представљени и дискутовани резултати мерења перформанси алата у односу на друге, савремене алате. У последњем поглављу рада биће представљени закључци рада уз коментар о потенцијалним правцима даљег развоја.

2. ПРЕГЛЕД СТАЊА У ОБЛАСТИ

У одабиру репрезентативних савремених алата везаних за област разбијања лозинке *ZIP* архива може се увести подела у погледу методологије приступа решавању проблема напада грубом силом, где се различита решења могу сврстати у процесорске алате (што представља прву групу) и алате базиране на графичким картицама. Додатно, ови алати могу подржавати употребу у *cloud* окружењу за постизање масовне паралелизације. Сходно томе, као два репрезентативна алата узети су алати отвореног кода *John The Ripper* као превасходно процесорски алат и *Hashcat* као алат базиран на графичким картицама.

2.1. *John The Ripper*

John The Ripper [1] нуди различите прилагодљиве методе напада од којих је напад грубом силом од највећег значаја за овај рад, иако су у погледу рада са *ZIP* архивама свакако интересантни и хибридни и

НАПОМЕНА:
Овај рад је проистекао из мастер рада чији ментор је био др Стеван Гостојић, ред. проф.

mask напади. *John The Ripper* се у стандардним подешавањима увек ослања на *CPU*, док се додатним специфицирањем може дефинисати употреба *GPU*, за које је потребна инсталација *OpenCl.*

Једна од значајних предности *John The Ripper* алата јесте модул за екстракцију битних информација и добијање хеша *ZIP* датотеке - ***zip2John***. Овако добијене информације (које у ширем смислу могу да се назову хеш иако то суштински нису) служе као улаз основном модулу *John The Ripper*. *Hashcat* као очекивани улаз такође очекује хеш специфициран по нешто измењеном [2] излазном формату ***zip2John*** модула, те је овај модул практично неопходан и за употребу *Hashcat,* пошто сличан алат не долази у склопу *Hashcat-a.*

2.2. *Hashcat*

Hashcat [3], као алат који се веома често користи у области дигиталне безбедности, долази као стандардан уз инсталацију неких оперативних система намењеним за употребу у овој области, као што је *Kali Linux.*

Иако старије верзије подржавају и процесорски режим рада, савремене верзије *Hashcat* алата подржавају извршавање искључиво на графичким картицама. Као и у претходном случају, од значаја за овај рад је режим напада грубом силом.

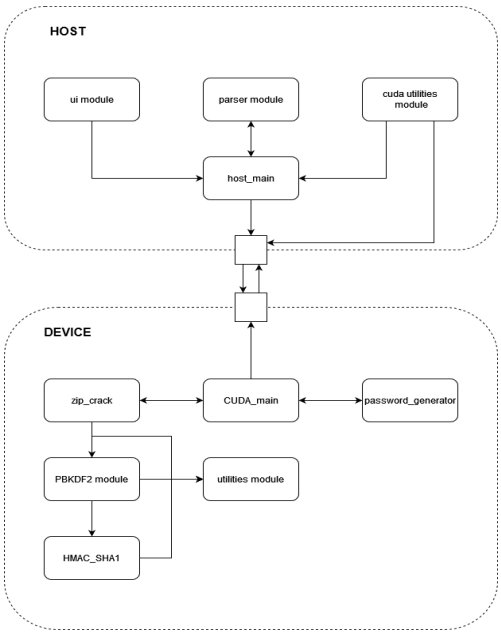
Битно је напоменути да *Hashcat* представља алат за разбијање хеша, те као такав нема могућност директног рада са *ZIP* датотекама, већ се улазни параметри (хеш), потребни за добијање шифре морају прибавити ручно или употребом неког екстерног алата (најчешће поменути *zip2John*).

3. СПЕЦИФИКАЦИЈА

Апликација се према начину извршавања кода може поделити на фронтенд део, који се извршава на процесору и бекенд део, који обухвата функционалности које се извршавају на једној или више графичких картица. На слици 1 приказана је шема предложене архитектуре алата.
Најзначајнији модули фронтенда јесу *host_main* и *парсер* модули. Модул *host_main* представља централно место апликације и задужен је за контролу тока програма, регистрацију парсер и бекенд модула, обраду информација добијених из парсера и креирања одговарајућег окружења на графичкој картици (трансфер података, покретање кернел функција итд.).

Осим тога, ослањајући се на функционалности *UI* и *cuda_utilities* модула, главни модул врши креирање простора претраге шифре на основу улазних података *UI* модула. Парсер модул обухвата функционалности детекције и провере интегритета ZIP архиве, поступке парсирања и екстракције неопходних информација.

У току овог поступка врши се секвенца провера у односу на верзију архиве, тип архиве, тип енкрипције и слично, путем којих се одбацује свака *ZIP* архива која, иако валидног формата, евентуално није подржана од стране алата.



Слика 1 - *Шематски приказ структуре алата*

Бекенд део ослања се на *CUDA_main* као централни модул који посредује између свих осталих елемената бекенда и фронт дела. Модул *password_generator* има улогу постављања иницијалног речника (простора претраге) и креирања и доделе шифре свакој нити. Поред тога, садржи и функцију која као аргумент прима сваку неуспелу шифру на основу које враћа логички следећу чиме је омогућен итеративни рад програма.

Други значајни скуп функционалности обухвата *zip_crack* модул, који енкапсулира све функционалности везане за бекенд рад са *ZIP* архивама, те се као такав може посматрати као плагин за *ZIP*. Овај модул садржи подмодуле за деривацију кључа помоћу *PBKDF2* [4] алгоритма заснованог на *HMAC-SHA1* [5][6] псеудонасумичној функцији. *PBKDF2* ради у конфигурацији од 1000 итерација са дужином добијеног кључа од 528 бита.

4. ИМПЛЕМЕНТАЦИЈА

Како би се дефинисала оптимална конфигурација на графичкој картици (број нити, оптимална употреба регистара и сл.) најпре се на основу улазних аргумената програма врши рачунање укупног простора претраге и евентуално даље партиционисање истог (у случају извршавања алата на више графичких картица или рачунара).

Како би се подржао рад на више уређаја, алату је могуће проследити опционе аргументе који дефинишу број машина које врше обраду *ZIP* архиве и индекс текуће машине. Поред поменутих опционих аргумената, алат региструје све доступне *NVIDIA* графичке картице у оквиру дате машине. Као обавезни улазни аргументи специфицирају се максимална дужина претражене шифре као и речник над којим се претрага врши. Могуће је одабрати један од четири речника: нумеричка шифра, алфанумеричка (само мала слова), алфанумеричка шифра и комплетан речник (алфанумеричка са специјалним карактерима).

Претходно наведене информације користе се за дефинисање укупног простора претраге шифре, даље партиционисање истог и доделу дела посла свакој од регистрованих графичких картица на свим рачунарима. Формула 1 приказује начин рачунања укупног броја шифри које се испитују:

$$S = \sum_{i=1}^{n} x^i = \frac{x * x^{n-1}}{x - 1}$$

Формула 1– *Укупан простор претраге*

У претходној формули *x* представља дужину одабраног речника, а *n* максималну дужину шифре.

За сваку дефинисану графичку картицу отвара се *CUDA context*, на коме се специфицира покретање одређеног броја *stream*-ова. Формула 2 приказује начин на који се рачуна број *stream*-ова:

$$2^{min(\lfloor log_{10} k \rfloor - 6,6)}$$

Формула 2 - *Рачунање броја stream-ова*

Коефицијент *k* представља целобројну вредност добијену као количник укупног простора претраге са бројем рачунара и графичких картица међу којима се врши подела посла и трију константи: број блокова – 6, број нити по блоку – 512 и основни број *stream*-ова – 26. Важно је напоменути да се ова формула примењује само за коефицијент већи од 10 милиона, док се у осталим случајевима примењује вредност 26. Након дефинисања броја *stream*-ова врши се рачунање „укупног посла“ (броја шифри које ће бити испитане) по нити графичке картице (уређаја), те се извршавање пребацује на уређај.

По покретању кернел функције уређаја, свакој нити додељује се јединствени идентификатор, који се користи као механизам за униформно мапирање простора претраге додељеног текућем уређају на конкретне нити. Поступак функционише по принципу креирања класа еквиваленције поделом додељеног простора претраге на укупан број нити, те доделе прве вредности сваке класе еквиваленције одговарајућој нити.

Овакво креирање класа еквиваленције резултује дуалном природом механизма претраге. Са једне стране, обезбеђује се униформно испитивање простора претраге *додељеног уређају* у оквиру текућег *stream*-а, док је, посматрано са глобалног аспекта, претрага ипак локализована, односно секвенцијалне природе на нивоу *stream*-а. Локална природа претраге на нивоу *stream*-а омогућава кориснику који има додатне информације о шифри (на пример познавање једног или више почетних карактера) задавање улазних аргумената који резултују таквим партиционисањем укупног простора претраге који обезбеђује да простор претраге *додељен уређају* садржи шифре које почињу датим карактерима, чиме се простор претраге фактички сужава.

Додатну предност дефинисања поменутих класа еквиваленција представља чињеница да је поступак преласка на наредну шифру (следећа итерација) рачунарски веома једноставан. Наиме, за сваку нит предодређен је број шифри које ће бити испитане, па се поступак преласка на наредну шифру своди на једноставну операцију инкрементирања вредности последњег карактера текуће шифре. Ово је имплементирано једноставном употребом низа чија структура прати ASCII табелу, са померајем у лево. Инкрементирање шифре постиже се читањем вредности овог низа са позиције која одговара целобројној вредности последњег карактера текуће шифре, те заменом старе вредности новом.

Даљи процес рада алата своди се на употребу поменуте *PBKDF2* функције у конфигурацији од 1000 итерација са дужином добијеног кључа од 528 бита, када се по добијању кључа врши провера последњих 2 бајта у односу на верификациону вредност добијену парсирањем архиве. Дужина верификационе вредности узрокује добијање лажно позитивних резултата у $1/2^{16}$ случајева, што представља 1 лажно позитиван резултат на сваких 65,536 шифри [7]. Из овог разлога, шифре за које прођу иницијалну проверу верификационе вредности врши се провера аутентификационе вредности (такође добијене парсирањем архиве). Успешна провера аутентификационе вредности подразумева употребу добијеног хеша као кључа за добијање *hmac-sha1* хеша над енкриптованим подацима архиве. Првих 10 бајтова новодобијеног хеша представља аутентификациону вредност, која би у случају исправног хеша шифре, требало да се подудара са вредношћу прослеђеном из парсера. У случају подударања ове вредности, контрола извршавања се враћа процесору, где се добијена шифра користи за декрипцију архиве и проверава њена исправност.

5. РЕЗУЛТАТИ

Сва мерења извршена су на машини следеће конфигурације:

- Процесор *intel i7-8750H* на *4.00 GHz*
- Графичка *NVIDIA GeForce GTX 1050Ti*
- Интегрисана графичка *(hashcat) Intel UHD 630*
- *RAM 16GB*

John the ripper алат у *CPU* конфигурацији није остварио приближно задовољавајуће перформансе (у односу на *GPU* решења), те је изузет из табеларног приказа. Наиме, у итеративном моду (груба сила), овај алат не даје брзе резултате већ за шифре дужине 4 карактера. Тестиране су шифре: 1234 – 1s; 2209 – 15s; 9909 – 785s; 1xlk – процес прекинут након сат времена рада.

Може се констатовати да алат може успешно да се носи са релативно кратким искључиво нумеричким шифрама, док са повећањем простора претраге осталим карактерима цео процес постаје превише дуг. Табела 1 даје приказ мерења *hashcat* и алата имплементираног у овом раду:

Hashcat							
Тип/дужина	6	7	8	9	10	11	12
нумеричка шиф	2s	16s	119s	19m	~3h10m	~1d7h	~13d
комплетан скуп	~9d12h	~2g180 d	-	-	-	-	-
Алат							
нумеричка шиф	7.8s	79.2s	13m15s	2h12m	~22h	~9d	~90d
комплетан скуп	~58d	~15g	-	-	-	-	-
нум и мала слова	1h19m	~1d22h	~70d	-	-	-	-
алфанумеричка	~2d12h	~112d	-	-	-	-	-

Табела 1- *Приказ мерења перформанси*

Важно је напоменути да *Hashcat* користи и интегрисану поред *NVIDIA GPU*.

У табели је извршено поређење шифре комплетног скупа и нумеричке шифре, јер *Hashcat* алат не нуди директно друге две комбинације за речник. Последња два реда дају мерења и за преостала два речника.

Сва мерења до 2 сата трајања су извршена, док су остала (означена префиксом ~) дата као приближна процена.

Основни и кључни закључак који се може донети на основу наведених вредности јесте да приступ нападом грубом силом никако није ефикасан у већини случајева, осим код веома кратких или строго нумеричких шифри. Експоненцијални раст трајања процеса практично гарантује неуспех за сваку шифру дужине преко 8 која садржи комбинацију алфанумеричких карактера и специјалних карактера.

6. ЗАКЉУЧАК

У овом раду представљен је предлог архитектуре и имплементације алата за опоравак лозинке *ZIP* архиве енкриптоване помоћу *AES-256* алгоритма по *WinZip* стандарду. Имплементација алата заснива се на паралелизацији хеш алгоритма употребом графичке картице, односно *NVIDIA CUDA* технологије.

Оно што овај алат издваја од осталих (што подразумева и поменуте *Hashcat* и *John the ripper* алате) јесте независност од употребе екстерних алата за извршавање процеса. Наиме и *Hashcat* и *John the ripper* ослањају се на *zip2John* модул *John the ripper* алата, који је претходно неопходно прибавити и извршити да би се обезбедио адекватан улаз у алат. Са друге стране, алат предложен у овом раду представља комплетно решење са једноставним корисничким интерфејсом, који од корисника захтева минималне улазне информације.

Као негативне стране алата, свакако се истиче ослањање искључиво на метод грубе силе, који сам по себи није ефикасан и дефинитивно није користан за сложене и дуге шифре.

Ток даљег развоја би могао да се одвија у правцу имплементације ефикаснијих метода генерисања кандидата шифре (у односу на метод грубе силе). Затим додатно унапређење корисничког интерфејса којим би се кориснику обезбедила додатна контрола над комплетним процесом, у циљу смањења простора претраге шифре, као и подршка употребе интегрисаних и осталих не CUDA графичких картица.

7. ЛИТЕРАТУРА

[1] https://github.com/openwall/john (приступљено у новембру 2023.)

[2] https://github.com/hashcat/hashcat/issues/2186 (приступљено у новембру 2023.)

[3] https://hashcat.net/hashcat (приступљено у новембру 2023.)

[4] http://www.faqs.org/rfcs/rfc2898.html (приступљено у новембру 2023.)

[5] https://datatracker.ietf.org/doc/html/rfc2104 (приступљено у новембру 2023.)

[6] https://www.rfc-editor.org/rfc/rfc3174 (приступљено у новембру 2023.)

[7] https://www.winzip.com/en/support/aes-encryption/#key-generation (приступљено у октобру 2023.)

Кратка биографија:

Лука Курељушић рођен је 1999. године у Суботици, Република Србија. Основне академске студије завршио је 2022. го-дине на Факултету техничких наука, на ком брани и мастер рад 2024. године из области Електротехнике и рачунарства смер софтверско инжењерство и инфо-рмационе технологије.
контакт:
lkureljusic@ymail.com