Jurnal TRANSFORMATIKA
Vol.20, No.2, Januari 2023, pp. 105-112
P-ISSN: 1693-3656, E-ISSN: 2460-6731
https://journals.usm.ac.id/index.php/transformatika/

Distributed MD5 Brute Force Using Message Passing Interface (MPI) on ARM Architecture

M. Bagus Saputra[1], Chaerul Umam[2]
[1]Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro
Jl. Imam Bonjol No.207, Pendrikan Kidul, Semarang Tengah, Kota Semarang, Jawa Tengah 50131, e-mail: 111201710233@mhs.dinus.ac.id
[2]Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro
Jl. Imam Bonjol No.207, Pendrikan Kidul, Semarang Tengah, Kota Semarang, Jawa Tengah 50131, e-mail: chaerul@dsn.dinus.ac.id

ARTICLE INFO

History of the article :
Received 29 Agustus 2022
Received in revised form 29 Desember 2022
Accepted 5 Januari 2023
Available online 30 Januari 2023

Keywords:

Brute-force attack, MD5, Message Passing Interface, ARM

*** Correspondece:**

Telepon:
-

E-mail:
chaerul@dsn.dinus.ac.id

ABSTRACT

Over the years, there has been a computing paradigm shift towards smart devices. However, the security aspect of this type of device is questionable. In this research, the writer provides a design, implementation, and testing of a distributed MD5 brute-force attack method. Testing is performed using two systems. The first system is an ARM Cluster consisting of four single-board computers. The second system is a conventional server with two Xeon processors. From this research, the writer wants to answer the question of whether distributed brute-force attacks using IoT nodes can be realized in the real world by doing a comparison between increases of ARM Cluster performance with node addition to conventional server performance. From the test result, Xeon model performance is equivalent to 8 ARM Clusters and around 30 ARM nodes are required to match Xeon model performance.

1. INTRODUCTION

Seiring dengan perkembangan perangkat benam terhubung yang berupa objek sehari-hari dan memiliki perangkat elektronik tambahan berupa sensor, prosesor, memori dan konektivitas yang memungkinkan perangkat tersebut untuk mengirimkan informasi, menerima perintah bahkan memproses data melalui media internet [1]. Perangkat benam terhubung mencakup beberapa macam jenis sesuai kegunaanya. Misalnya Smart Home yang terdiri dari TV pintar, kulkas pintar, pengatur suhu, kamera keamanan, kunci pintu, lampu pintar dan sistem deteksi lokasi yang dapat mengirimkan informasi dan menerima perintah dari internet, dengan meningkatnya jumlah perangkat maka semakin banyak pula data yang dihasilkan dan diperlukan pula kemampuan komputasi yang lebih besar. Seiring dengan meningkatnya jumlah perangkat benam terhubung dan meningkatnya kemampuan komputasi perangkat, meningkat pula resiko akan pembajakan pada

perangkat tersebut, mulai dari konfigurasi keamanan yang kurang baik sehingga penggunaan password default [2] membuat perangkat tersebut menjadi sasaran bagi aktor jahat. Pada beberapa kasus, perangkat yang dibajak akan diubah menjadi botnet [3] dan digunakan dalam serangan DDoS, selain itu sebuah penelitian menunjukkan bahwa botnet memungkinkan untuk digunakan sebagai Cluster dalam memecahkan password [4].

Serangan brute-force atau dapat dikenal juga sebagai exhaustive key search merupakan salah satu metode pencarian kunci kriptografi dengan cara mencoba tiap kemungkinan kunci satu persatu hingga kunci yang tepat ditemukan. Metode ini memerlukan sumber daya yang besar karena tingkat kesulitan pencarian berupa eksponensial seiring dengan banyaknya jumlah karakter pada kunci [5]. Selain itu ada pula metode pencarian menggunakan daftar password yang telah diketahui, metode ini disebut dictionary attack dan biasanya memerlukan lebih sedikit sumber daya bila dibandingkan dengan exhaustive key search. Meskipun metode brute-force merupakan proses yang memerlukan sumber daya besar, ada beberapa cara untuk mengurangi penggunaan sumber daya yang digunakan. Misalnya metode kriptanalisis yang mencari kelemahan implementasi algoritma kriptografi yang dapat mengurangi waktu brute-force [6]. Selain itu seiring dengan perkembangan teknologi berupa peningkatan pemrosesan instruksi dan multi-core pada prosesor modern menjadikan beberapa algoritma kriptografi rentan terhadap serangan brute-force.

Salah satu algoritma kriptografi yang rentan terhadap serangan adalah Message Digest 5 atau MD5. Peneliti menemukan bahwa MD5 rentan terhadap serangan seperti brute-force attack, rainbow attack, birthday attack, dictionary attack, dan masih banyak lagi [7]. Meski MD5 memiliki banyak kelemahan dan rentan terhadap berbagai macam serangan, MD5 masih sering digunakan dalam berbagai aplikasi. Berdasarkan penelitian gabungan yang dipublikasikan pada 32nd European Conference on Object-Oriented Programming (ECOOP 2018), menyatakan bahwa hasil evaluasi 10.000 aplikasi Android, 43% memiliki kesalahan implementasi keamanan, dan kebanyakan kesalahan implementasi tersebut disebabkan oleh penggunaan MD5 dan SHA-1 dalam aplikasi [8]. Pada awalnya peneliti menggunakan superkomputer dan multi GPU Cluster untuk melakukan brute-force atau mencari hash collision [9], namun sistem itu memerlukan sumber daya besar yang sulit dicapai bagi kebanyakan orang. Maka dari itu tingkat keamanan MD5 perlu diuji dengan sistem yang lebih praktis.

RESEARCH METHODS

Pada penelitian ini akan dibuat satu buah model perangkat lunak yang dapat melakukan brute-force terdistribusi dan menggunakan seluruh CPU Core pada sistem tujuan dan dua buah model perangkat keras yang terdiri dari sebuah Cluster ARM sebagai representasi perangkat IoT yang dicluster dan sebuah sistem dengan prosesor Xeon sebagai representasi server konvensional. Metode yang digunakan penulis pada penelitian ini adalah metode eksperimen dengan melakukan perbandingan performa penyelesaian brute-force yang direpresentasikan dengan jumlah masalah dibagi dengan waktu penyelesaian yang akan menghasilkan variabel berupa hash per second. Kemudian data dari Cluster ARM akan dikumpulkan mulai dari satu node hingga empat node dan akan dibandingkan dengan data dari sistem Xeon untuk menentukan sejauh atau sedekat apa perbandingan performa Cluster ARM dengan sistem Xeon. Tiap pengambilan sampel akan dilakukan sebanyak 5 kali dan akan dirata - rata.

1. Pengumpulan Data

Perangkat pertama yaitu model Cluster ARM yang terdiri dari empat buah OrangePI Zero terhubung dengan switch 100 Mbps dengan spesifikasi masing – masing:

Tabel 1. Spesifikasi perangkat Cluster ARM

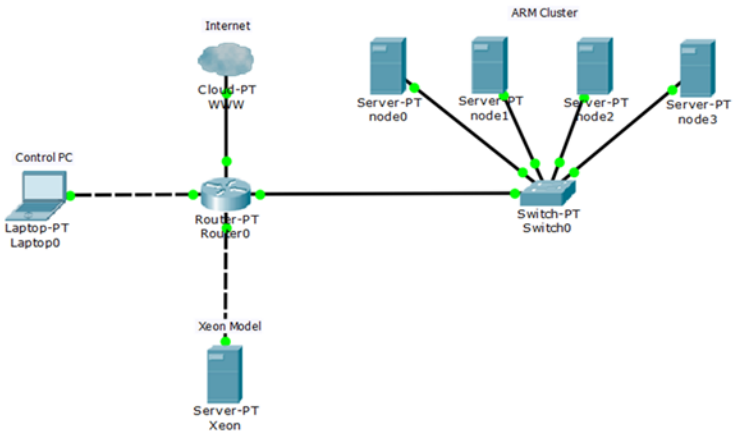
Prosesor	Allwinner H2+ Quad Core 1.2GHz
Board	OrangePI Zero
RAM	512MB DDR3
Drive	8GB MicroSD
NIC	Integrated 100Mbps
OS	Armbian 5.35 (Ubuntu 16.04)

Cluster ARM terdiri dari satu buah Master node dan tiga buah Worker node. Master node bertugas untuk melakukan penugasan pada Worker node dan berfungsi sebagai antarmuka utama. Perangkat kedua yaitu model server Xeon dengan spesifikasi:

Tabel 2. Spesifikasi perangkat *Xeon*

Prosesor	2x Intel Xeon X5670 Six Core 2.9GHz
Board	ASUS Z8NA-D6
RAM	12GB DDR3 ECC
Drive	128GB SSD
NIC	Intel 82574L 1Gbps
OS	Ubuntu 20.04

Selain itu diperlukan pula perangkat tambahan yang berupa satu buah router dan satu buah komputer kendali. Seluruh perangkat dirangkai sesuai diagram dibawah kemudian dilakukan konfigurasi dan installasi perangkat lunak dan library pendukung seperti compiler C, libmpich-dev, libssl-dev dan nfs-common.



Gambar 1. Diagram Cluster dan komponen jaringan

2. Pengembangan Model Perangkat Lunak

Pengembangan perangkat lunak dimulai dari pembuatan prototipe konsep untuk mempelajari bagaimana reaksi sistem terhadap beberapa implementasi perangkat lunak sekaligus menjadi dasar untuk optimalisasi pada perangkat lunak akhir. Pada tahapan pengembangan prototipe ini, penulis menggunakan metode Rapid Application Development yang memungkinkan penulis untuk membuat berbagai macam prototipe sekaligus melakukan uji untuk mendapatkan

DOI :10.26623/transformatika.v20i2.5630

hasil terbaik. Proses ini diulang sehingga didapatkan perangkat lunak yang sesuai dengan kebutuhan. Berikut adalah diagram alur perangkat lunak:

Gambar 2. Diagram alur program Brute-force MD5

Program dimulai dengan melakukan inisalisasi MPI, kemdudian master node melakukan pemanggilan dan uji terhadap tiap CPU Core pada masing – masing node yang didefinisikan pada hostfile. Jika tiap node menjawab panggilan dengan benar, maka sistem melanjutkan dengan melakukan print timestamp dan menjalankan fungsi brute-force scheduler. Variabel utama yang digunakan pada main function adalah data atau hash yang akan diuji, rank yang berupa indeks n per core, size yang digunakan untuk menyimpan jumlah maksimal core dan reply yang digunakan sebagai variabel komunikasi antar core.

Pada fungsi brute-force scheduler, sistem melakukan inisialisasi memori dan variabel yang diperlukan. Kemudian melakukan inisialisasi variabel progress dan index progress dengan memberikan value berupa \0 dan -1. Selanjutnya sistem melakukan iterasi brute-force sekaligus melakukan komparasi apabila hasil hash pada progress saat ini sama dengan variabel test. Jika hasil hash sama dengan variabel test, maka sistem melakukan pemanggilan MPI_Abort untuk memberi tahu pada node lain bahwa hash telah ditemukan diikuti dengan melakukan output hasil hash dan timestamp. Jika hasil hash tidak sama dengan variabel test, maka sistem melakukan iterasi kepada variabel progress.

Variabel loop merupakan iterasi brute-force utama pada tiap core. Variabel loop menggunakan tipe data long dan dilakukan inisialisasi sesuai dengan rank-1, kemudian dilakukan iterasi sesuai dengan variabel size. Dengan demikian tiap core memiliki task masing - masing.

Variabel idxProgress digunakan sebagai penyimpan hasil iterasi brute-force. Variabel idxProgress terdiri dari array integer dengan panjang sesuai dengan variabel length. Secara desain, penulis menggabungkan variabel idxProgress[0],[1],[2] dengan variabel loop. Hal ini bertujuan untuk memastikan bahwa sistem mampu menangani jumlah core yang besar dengan perhitungan berupa pemangkatan pada panjang karakter set, namun penulis belum dapat memastikan apakah sistem mampu menangani core dengan kapasitas maksimal atau sistem akan melakukan abort dan menyatakan hash tidak ditemukan.

Variabel string progress terdiri dari array karakter dictionary dengan panjang sesuai dengan variabel length atau jumlah maksimal karakter yang akan di brute-force. Variabel progress digunakan untuk menyimpan karakter hasil dari konversi idxProgress.

Selain itu brute-force scheduler memerlukan beberapa fungsi penunjang diantaranya adalah fungsi indexToText yang digunakan untuk mengkonversi hasil increment brute-force pada idxProgress menjadi string pada variabel progress. Kemudian fungsi progressLen digunakan untuk menghitung jumlah karakter pada variabel progress. Sedangkan fungsi compareResult digunakan untuk membandingkan hasil MD5 dari variabel progress terhadap variabel test.

RESULTS

Pengujian dilakukan untuk mencari selisih waktu dengan melakukan eksekusi program dan mencatat waktu mulai dan waktu selesai pada Cluster ARM dan Model Xeon. Kemudian dilanjutkan dengan tahap analisa yang dilakukan untuk mengetahui perbandingan dan skalabilitas performa sistem dengan melakukan perbandingan hashrate. Pengujian pada Cluster ARM dilakukan pada satu node dengan empat core, dua node dengan 8 core, 3 node dengan 12 core dan 4 node dengan 16 core, selain itu pengujian pada Model Xeon dilakukan dengan 24 thread. Masing - masing uji dilakukan 5 kali pengambilan sampel untuk selanjutnya dirata - rata.

Setelah didapatkan hasil rata - rata selisih waktu dari tiap sistem, tahap selanjutnya adalah perhitungan hashrate tiap sistem dengan membagi jumlah hash yang diproses dengan waktu penyelesaian. Untuk menemukan jumlah hash keseluruhan yang diproses oleh sistem dapat dihitung dengan menambahkan hasil pemangkatan charset (x) dengan indeks (i), kemudian dilakukan penjumlahan hingga jumlah maksimal karakter yang akan di brute-force (n).

DOI :10.26623/transformatika.v20i2.5630

$$\sum_{i=1}^{n} x^{i} \quad (1)$$

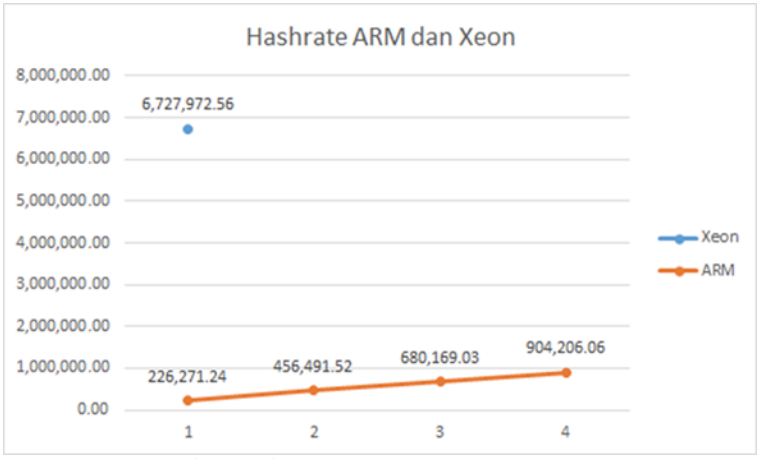
DISCUSSION

Pada penelitian ini penulis melakukan brute-force mulai dari satu digit 0 hingga 5 digit menggunakan 62 karakter set, dengan demikian jumlah hash keseluruhan yang diproses oleh sistem sejumlah 931151402 hash. Berikut adalah hasil hashrate dari masing - masing hasil pengujian.

Tabel 4. Hasil Hashrate tiap sistem

Sistem	Rata - Rata Waktu	Hashrate
1 node ARM 4 core	4115.2	226,271.24
2 node ARM 8 core	2039.8	456,491.52
3 node ARM 12 core	1369	680,169.03
4 node ARM 16 core	1029.8	904,206.06
1 node Xeon 24 thread	138.4	6,727,972.56

Untuk memudahkan dalam melakukan visualisasi perbandingan dan skalabilitas sistem, penulis melakukan plotting data kedalam grafik. Berikut adalah grafik skalabilitas performa sistem Cluster ARM dengan berbagai konfigurasi node.



Gambar 3. Grafik performa antara Cluster ARM dengan Xeon

Dari grafik diatas ditunjukkan bahwa peningkatan performa Cluster ARM dari satu node hingga empat node mengalami trend yang relatif linear, dengan demikian implementasi perangkat lunak dan perangkat keras Cluster ARM mampu membagi task dan menggunakan seluruh core secara rata, namun performa yang ditunjukkan berupa performa relatif berdasarkan perangkat lunak yang diuji. Maka dari itu untuk melihat performa relatif perangkat secara nyata, dibutuhkan performa model Xeon sebagai representasi sistem konvensional.

Grafik diatas juga menunjukkan bahwa performa model Xeon terlampau jauh dari performa Cluster ARM, namun berapa Cluster ARM yang dibutuhkan untuk menyamai satu model Xeon? Untuk menjawab pertanyaan ini, penulis melakukan perhitungan performa relatif (pR) yang didapatkan dengan melakukan pembagian hashrate model Xeon (pX) dengan hashrate Cluster ARM (pA).

$$pR = \frac{pX}{pA}$$
$$pR = \frac{6{,}727{,}972.56}{904{,}206.06} \quad (2)$$
$$pR = 7.44075145$$

Dari perhitungan diatas didapatkan performa relatif antara model Xeon dengan Cluster ARM sebanyak 7.44 yang berarti, dibutuhkan sekitar 8 buah Cluster ARM untuk menyamai satu model Xeon. Kemudian untuk menjawab pertanyaan utama dari penelitian ini tentang praktikalitas dan berapa besar ancaman yang dihasilkan dari perangkat Cluster ARM, penulis melakukan perhitungan relatif (pR1) dengan melakukan pembagian hashrate model Xeon (pX) dengan hashrate satu node ARM (pA1).

$$pR1 = \frac{pX}{pA1}$$
$$pR1 = \frac{6{,}727{,}972.56}{226{,}271.24} \quad (3)$$
$$pR1 = 29.73410405$$

Dari hasil perhitungan tersebut, didapatkan performa relatif sebesar 29.73 atau dibutuhkan sekitar 30 node ARM untuk menyamai model Xeon yang digunakan pada penelitian ini.

CONCLUSIONS AND RECOMMENDATIONS

Penelitian tentang implementasi MD5 brute-force terdistribusi menggunakan library MPI pada arsitektur ARM menghasilkan kesimpulan berupa penerapan algoritma brute-force terdistribusi dapat membagi task dan menggunakan seluruh CPU Core secara merata. Hal ini ditunjukkan oleh trend hashrate yang relatif linear dari satu node hingga empat node, perbandingan performa Cluster ARM dengan model Xeon terlampau cukup jauh. Hal ini ditunjukkan oleh hasil performa relatif antara Cluster ARM dengan model Xeon yang menujukkan bahwa butuh 8 Cluster ARM untuk menyamai model Xeon, praktikalitas serangan brute-force MD5 menggunakan arsitektur ARM dapat direalisasikan. Hal ini ditunjukkan dengan hasil performa relatif antara satu node ARM dengan satu model Xeon. Dari perhitungan ini menunjukkan bahwa butuh sekitar 30 node ARM untuk menyamai satu model Xeon.

Pengembangan penelitian ini dapat dilakukan pada penelitian – penelitian selanjutnya seperti pengujian lebih lanjut untuk mengetahui bagaimana respon algoritma brute-force dengan jumlah core yang lebih besar, pengujian algoritma kriptografi lain seperti SHA3, AES atau bahkan RSA, penelitian lebih lanjut untuk menentukan apakah sistem distribusi serupa dapat digunakan pada beban kerja selain kriptografi.

REFERENCES

[1] Sinung, S., & I Putu, A. E. (2015). Wireless Sensor Network. Bandung: Informatika. Vervier, P.-A. (2018). Before Toasters Rise Up: A View into the Emerging IoT Threat Landscape. Research in Attacks, Intrusions, and Defenses (pp. 556--576). Cham: Springer International Publishing.

DOI :10.26623/transformatika.v20i2.5630

[2] Dave, M., Wei, G., & Charles, D. (2020, September 17). A New Botnet Attack Just Mozied Into Town. Retrieved from Security Intelligence - Cybersecurity Analysis & Insight: https://securityintelligence.com/posts/botnet-attack-mozi-mozied-into-town/
[3] S. Salamatian, W. H. (2019). Why Botnets Work: Distributed Brute-Force Attacks Need No Synchronization. IEEE Transactions on Information Forensics and Security, 2288-2299.
[4] Kaspersky Lab. (2018, April 26). Brute Force Attack: Definition and Examples. Retrieved from Kaspersky: https://www.kaspersky.com/resource-center/definitions/brute-force-attack
[5] OWASP Foundation. (2021, May 8). Cryptanalysis | OWASP Foundation. Retrieved from OWASP Foundation: https://owasp.org/www-community/attacks/Cryptanalysis
[6] Asmin Bhandari, M. B. (2017). Enhancement of MD5 Algorithm for Secured Web Development. Journal of Software.
[7] S. Krüger, J. S. (2019). CrySL: An Extensible Approach to Validating the Correct Usage of Cryptographic APIs. IEEE Transactions on Software Engineering, 1-1.
[8] V. Chiriaco, A. F. (2016). Finding partial hash collisions by brute force parallel programming. 2016 IEEE 37th Sarnoff Symposium.
[9] Ali, Z. M., & Ahmad, A. (2016). IMPLEMENTATION OF PARALLEL ALGORITHM FOR LUC CRYPTOSYSTEMS BASED ON ADDITION CHAIN BY A MESSAGE PASSING INTERFACE. Journal of Theoretical & Applied Information Technology, 92, 1.
[10] Asmin Bhandari, M. B. (2017). Enhancement of MD5 Algorithm for Secured Web Development. Journal of Software.
[11] Auth0 Inc. (2021, November 27). How to Hash Passwords: One-Way Road to Enhanced Security. Retrieved from Auth0 - Blog: https://auth0.com/blog/hashing-passwords-one-way-road-to-security/
[12] Bruce, S. (2015). Applied Cryptography: Protocols, Algorithms and Source Code in C, 20th Anniversary Edition. Wiley.
[13] Dean, T. (2012). Network+ guide to networks. Cengage Learning.
[14] Ghafouri, M. R. (2017). On a Novel Grid Computing-Based Distributed Brute-force Attack Scheme (GCDBF) By Exploiting Botnets. International Journal of Computer Network and Information Security, 21-29.