Desain *Infrastructure as Code* untuk Automasi Pengamanan Jaringan: *Hardening Router* Berbasis MikroTik

Infrastructure as Code (IaC) Design for Network Security Automation: Hardening MikroTik-Based Router

Arief Indriarto Haris[1)], Rd. Angga Ferianda[2)]

[1,2]Pusat Riset Sains Data dan Informasi, Badan Riset dan Inovasi Nasional (BRIN), Bandung
[1,2]Jl. Sangkuriang, Dago, Kecamatan Coblong Kota Bandung 40135, Indonesia

arie046@brin.go.id[1)], rdan002@brin.go.id[2)]

Diterima : 14 November 2022 || Revisi : 1 Januari 2023 || Disetujui: 28 April 2023

Abstrak – *Router* berperan penting dalam mengatur lalu lintas paket data didalam infrastruktur jaringan. Terganggunya fungsi *Router* oleh serangan siber akan berdampak secara langsung pada kualitas layanan Teknologi Informasi (TI) di dalam jaringan secara keseluruhan. Oleh sebab itu, diperlukan adanya pengamanan (*Hardening*) terhadap *Router* dari serangan siber. Namun, proses *Hardening Router* kerap menemui beberapa kendala dan tantangan, seperti kesalahan konfigurasi, proses konfigurasi yang berulang-ulang dan cenderung menghabiskan banyak waktu serta energi, terutama jika perangkat yang dikonfigurasi tersebut berjumlah banyak. Dengan menggunakan metode PPDIOO, penelitian ini bertujuan untuk mendesain *Infrastructure as Code* (IaC) yang berfokus pada *Hardening Router* berbasis MikroTik melalui proses automasi. Hasil yang diperoleh adalah seluruh desain IaC berhasil diimplementasi melalui proses automasi dan tidak ditemui adanya eror. Total durasi pelaksanaan *hardening* melalui automasi adalah 4 menit 28 detik. Dari hasil uji keamanan sistem diperoleh hasil bahwa *Router* berhasil terlindungi dan tidak ditemui adanya celah keamanan.

Kata Kunci: *Infrastructure as Code*, *Hardening Router*, Automasi

Abstract *– The router plays an important role in managing data packet traffic in the network infrastructure. Disruption of Router functions by cyber-attacks will have a direct impact on the quality of Information Technology (IT) services in the network as a whole. Therefore, it is necessary to harden the router to protect it from cyber-attacks. However, securing (Hardening) Router often encounters several obstacles and challenges, such as configuration errors, configuration processes that are repeated and tend to consume a lot of time and energy, especially if there are a lot of devices configured. By using the PPDIOO method, this study aims to design Infrastructure as Code (IaC) which focuses on MikroTik-based Router Hardening through an automation process. The results obtained were that all IaC designs had been successfully implemented through the automation process and no errors had been encountered. The total duration of hardening through automation was 4 minutes 28 seconds. The results of the system security test showed that the router was successfully protected and no vulnerabilities were encountered.*

Keywords: *Infrastructure as Code, Hardening Router, Automation*

PENDAHULUAN

Berdasarkan laporan hasil pemantauan insiden siber di Indonesia pada tahun 2021, sepanjang tahun 2021 tercatat Indonesia menjadi negara sumber serangan siber terbanyak dengan catatan 190.076.960 serangan, sekaligus sebagai destinasi serangan siber terbanyak dengan 1.000.941.603 serangan (Jeni Rahman et al., 2022). Hal ini mengindikasikan bahwa serangan siber merupakan suatu ancaman serius dan nyata yang dapat terjadi kepada siapa saja, dari tingkat individu hingga global.

Berdasarkan data yang bersumber dari Shodan dan dihimpun pada tahun 2019, Indonesia merupakan pengguna perangkat MikroTik nomor 5 terbanyak di dunia, dimana salah satu perangkat berbasis MikroTik tersebut adalah *Router* (Ceron et al., 2020). Sebagai *gateway* dan *backbone* di dalam jaringan, *Router* berperan penting dalam mengatur lalu lintas paket data, sehingga perannya menjadi sentral dan krusial, serta akan berdampak langsung terhadap layanan Teknologi Informasi (TI) di dalam jaringan (Pambudi & Muslim, 2017).

doi:10.56873/jpkm.v8i1.4936

Dikarenakan cakupan fungsi yang luas dan perannya tersebut, *Router* menjadi salah satu target utama *Attacker* didalam tujuannya untuk melumpuhkan layanan TI. Oleh karena itu, diperlukan adanya pengamanan (*Hardening*) pada *Router*, sebagai upaya menanggulangi serangan siber. Dengan menerapkan *hardening*, maka *Router* menjadi lebih sulit untuk dipenetrasi dan diserang (Akin, 2002).

Namun disisi lain, terdapat beberapa tantangan atau kendala yang kerap ditemui dalam melakukan *hardening* pada *Router*, diantaranya seperti proses yang berulang dan banyaknya jumlah perangkat yang perlu dikonfigurasi. Hal ini cenderung akan menghabiskan waktu dan energi lebih banyak. Selain itu, diperlukan keahlian dan kecermatan dari administrator agar tidak terjadi kesalahan konfigurasi yang dapat menyebabkan munculnya celah keamanan baru pada sistem (Swastika & Atitama, 2017).

Infrastructure as Code (IaC) menawarkan solusi untuk menjawab permasalahan tersebut. Dengan menerapkan IaC, infrastruktur TI didefinisikan dalam bentuk kode sumber terbuka yang bertujuan untuk menyediakan konfigurasi, penyiapan, penyebaran, manajemen, hingga pengembangan, tanpa mengabaikan aspek keamanan informasi (Agus & Pratama, 2021). IaC memungkinkan semua proses tersebut dilakukan dengan cepat dan berkelanjutan melalui infrastruktur berbasis *machine-readable files*, sehingga dapat dieksekusi melalui automasi dan meminimalisir konfigurasi fisik (Dalla Palma, Di Nucci, Palomba, et al., 2020). Ansible menjadi salah satu perangkat manajemen konfigurasi yang memungkinkan *script* IaC diimplementasikan melalui proses automasi, sehingga dapat meningkatkan efisiensi dan mengeliminasi dependensi pada suatu sistem spesifik ataupun operator (Kokuryo et al., 2020).

Pada penelitian sebelumnya, automasi telah banyak diterapkan untuk menjawab beberapa permasalahan, diantaranya seperti meminimalisir celah keamanan pada *Virtual Machine Images* (VMI) (Spichkova et al., 2020), hingga penanganan insiden keamanan informasi yang terbukti mampu mengurangi biaya dan sangat membantu administrator dalam mengambil keputusan, serta meminimalisir waktu dan kesalahan (Khumaidi, 2021). Sedangkan pada bidang jaringan, pembangunan 100 interkoneksi jaringan antara *node* IPv6 dan IPv4 dapat dilakukan kurang dari 30 detik melalui proses automasi (Bahnasse et al., 2019).

Adapun studi tentang penerapan automasi menggunakan Ansible telah cukup banyak dilakukan, diantaranya seperti automasi pembuatan *Virtual Private Server* (VPS) beserta dengan izin akses, *container*, dan manajemen sumber daya didalamnya (Hariyadi & Marzuki, 2020), automasi manajemen *container* (Pratama & Hariyadi, 2021), dan automasi penyediaan web server (Rifki Afandi et al., 2020). Disamping itu, Ansible telah terbukti mampu dan berhasil untuk melakukan konfigurasi pada *Router* melalui proses automasi, seperti pada studi (Islami et al., 2020; Mohd Fuzi et al., 2021; Rifki Afandi et al., 2020; Swastika & Atitama, 2017). Namun, penelitian tersebut tidak memfokuskan studi dari sisi keamanan, melainkan hanya terbatas pada konfigurasi fungsional *Router* secara umum, seperti *routing*, pengalamatan jaringan, izin akses sistem, hingga pengelolaan *bandwidth.*

Berbeda dengan (Perera et al., 2021) yang melakukan automasi hardening terhadap *Router* berbasis Cisco dengan perangkat automasi bernama NetBot berbasis Python. Penelitian ini bertujuan untuk merancang *script* IaC yang berfokus pada *hardening Router* berbasis MikroTik melalui proses automasi menggunakan Ansible. Script IaC dirancang dalam format YAML file berbentuk Ansible playbook. Adapun Ansible dipilih karena bersifat *open source*, memiliki cakupan penggunaan yang luas dan komunitas pengguna yang besar, serta keamanan komunikasi yang baik, menggunakan *Secure Shell* (SSH) dan menyimpan kunci RSA pada mesin lokal (Dalla Palma, Di Nucci, & Tamburri, 2020). Penelitian ini dilakukan pada *virtual environment* dengan menggunakan metode PPDIOO. Pengujian keamanan sistem dilakukan pada tahap akhir studi ini, dengan maksud untuk mengetahui tingkat keberhasilan pengamanan *Router* melalui metode ini.

METODOLOGI PENELITIAN

Penelitian ini menggunakan metode PPDIOO, yang mana metode ini dirancang oleh Cisco dan berbentuk suatu siklus hidup yang berisikan beberapa tahapan, yaitu *Prepare*, *Plan*, *Design*, *Implement*, *Operate*, dan *Optimize*. Metode ini dinilai telah teruji dan umum digunakan untuk membangun, mengelola, dan mengembangkan jaringan, sistem, hingga arsitektur teknologi informasi secara berkesinambungan, sistematis, dan berkelanjutan (Christanto & Suprayogi, 2017; Tantoni et al., 2020;

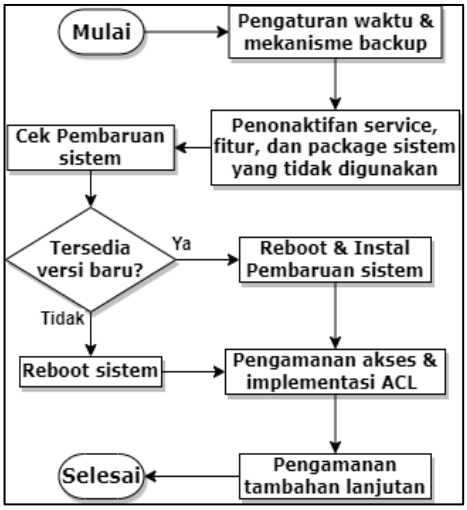
Wilkins, 2011). Adapun penerapan PPDIOO pada penelitian ini adalah sebagai berikut:

a) *Prepare:* Pada tahap ini proses yang dilakukan antara lain menginventaris, mengidentifikasi, dan menyiapkan sistem. Perangkat yang digunakan pada penelitian ini dijalankan di atas teknologi virtualisasi. Adapun spesifikasi perangkat yang digunakan terdapat pada Tabel 1. Komponen lain yang disiapkan adalah koneksi antara *Router*, Ansible *Control Node*, dan *File Server*. Koneksi tersebut dibangun menggunakan SSH, sedangkan *Secure Copy* (SCP) digunakan untuk mendistribusikan file hasil pencadangan konfigurasi dari *Router* ke *File Server*. Pada keduanya (SSH dan SCP) menerapkan enkripsi data menggunakan *public key*.

Tabel 1 Daftar dan Spesifikasi Perangkat

Perangkat	Spesifikasi	Keterangan
Router MikroTik (5 Unit)	OS: CHR Stable v.6.45.1 CPU: 1 Core @1,8GHz *Memory*: 256 MB *Storage*: 64MB	Sebagai *managed node* (target hardening melalui automasi).
Ansible *Control Node*	OS: CentOS 7 CPU: 1 Core @1,8GHz *Memory*: 2 GB *Storage*: 12 GB *Tools*: Ansible 2.9.25 Modul: *Ansible Community Network*	*Node* yang berfungsi untuk melakukan *push* perintah automasi.
Hacking Tools	OS: Kali Linux CPU: 2 Core @1,8GHz *Memory*: 3 GB *Storage*: 32 GB *Tools*: Nmap, Hydra, Hping3, Nessus	*Node* yang berfungsi untuk melakukan pengujian keamanan sistem.
File Server	OS: CentOS 7 CPU: 1 Core @1,8GHz *Memory*: 2 GB *Storage*: 12 GB	*Node* yang berfungsi untuk menyimpan file *backup Router*.

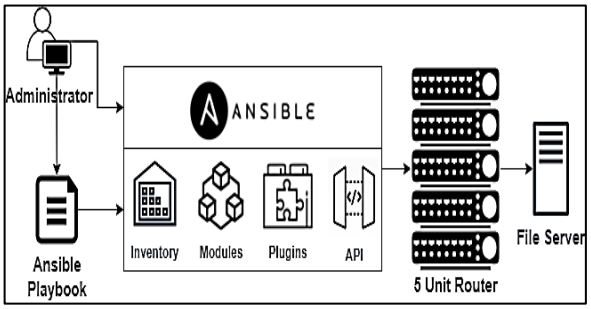
b) *Plan:* Perencanaan alur tindakan *hardening* pada *Router* mengacu kepada tren keamanan siber terkini, dokumentasi teknis resmi, dan rekomendasi praktik terbaik (CISA, 2020; MikroTik, 2019). Adapun fokus automasi hardening dikelompokkan menjadi beberapa bagian dengan alur proses pada Gambar 1.



Gambar 1 Alur Automasi Hardening

c) *Design*: Pada tahap ini dilakukan perancangan IaC berbasis Ansible Playbook berbentuk YAML *file* dan tambahan *file* var yang berisi kumpulan variabel yang dikonfigurasi sesuai dengan kebutuhan *hardening*. Masing-masing playbook dikelompokkan berdasarkan pada fokus dan tahapan proses *hardening* pada *Router*.

d) *Implement*: Tahapan implementasi dilakukan dengan mengeksekusi Ansible playbook yang telah dirancang, sehingga Ansible *Control Node* akan melakukan *push* setiap *task* automasi ke masing-masing *Router*. Adapun arsitektur implementasi automasi *hardening* terdapat pada Gambar 2.



Gambar 2 Arsitektur Automasi Hardening

e) *Operate*: Berikutnya dilakukan verifikasi dan pengujian terhadap konfigurasi *Router* yang sudah melalui proses automasi. Pengujian dilakukan melalui serangkaian uji keamanan sistem, diantaranya meliputi port scanning, serangan *brute force*, serangan *Denial of Services* (DoS), eksploitasi celah keamanan sistem berdasarkan CVE-2018-14847, dan *Vulnerability Assessment Test* (VA Test) (Haeruddin, 2021; Haris et al., 2022).

f) *Optimize*: Pada tahap akhir, dilakukan analisis dan evaluasi terkait keberhasilan *hardening Router* melalui proses automasi, sehingga dapat dikembangkan dan dioptimasi lebih lanjut dimasa mendatang.

HASIL DAN PEMBAHASAN

a) Desain IaC

Perancangan IaC untuk automasi *hardening Router* dikelompokkan menjadi beberapa bagian dalam bentuk Ansible Playbook sesuai dengan fokus dari masing-masing *task* automasi yang dijalankan. Vars-file berisi variable-variabel yang dideklarasikan, sehingga dapat dipanggil dan digunakan pada saat mengeksekusi Ansible playbook (Tabel 2).

Tabel 2 Algoritma Vars-file

Vars-file		
1:	*available_from*	← IP *address* yang diizinkan untuk mengakses *Router*
2:	*unused_services*	← servis yang tidak digunakan dan perlu dinonaktifkan
3:	*used_service*	← servis *Router* yang digunakan dan perlu diamankan
4:	*users*	← akun pengguna yang perlu diamankan
5:	*unused_packages*	← paket sistem yang tidak digunakan dan perlu dinonaktifkan
6:	*auth_pass_snmp*	← *authentication password* untuk mengamankan *service Simple Network Management Protocol* (SNMP)
7:	*encr_pass_snmp*	← *encryption password* untuk mengamankan servis SNMP
8:	*primary_ntp*	← IP *address* dari *Simple Network Time Protocol* (SNTP) server primer yang akan digunakan *Router*
9:	*secondary_ntp*	← IP *address* dari SNTP server sekunder yang akan digunakan *Router*

Playbook 1 berfokus pada pengaturan waktu dan mekanisme pencadangan konfigurasi pada *Router* (Tabel 3). Pengaturan waktu bertujuan agar setiap aktivitas atau kejadian yang terjadi pada *Router* dapat tercatat di log sistem dengan *timestamp* yang tepat. Pengaturan waktu menggunakan fitur *Simple Network Time Protocol* (SNTP) *client*. Proses *scripting* dan *scheduler* digunakan untuk pengaturan dan penerapan mekanisme pencadangan konfigurasi *Router*. Pencadangan konfigurasi terdiri dalam dua sesi dengan hasil berupa file dengan format (.backup) dan format (.rsc). Pada masing-masing sesi diberikan penundaan waktu beberapa detik agar *Router* dapat menyelesaikan masing-masing sesinya dengan sempurna, sehingga setiap task tidak tumpang tindih dan dapat menghasilkan file cadangan konfigurasi yang utuh serta tidak rusak. Berikutnya *Router* mengirim file tersebut ke file server via SSH *File Transfer Protocol* (SFTP). Penggunaan SFTP bertujuan agar keamanan data saat proses pengiriman terjaga. Adapun scheduler berfungsi untuk menjalankan *script* pencadangan konfigurasi secara periodik.

Tabel 3 Algoritma Playbook 1

PB-1.yaml	
1:	---
2:	***name:*** *Pengaturan waktu dan mekanisme backup*
3:	*target_routers*
4:	*vars_file*
5:	***task:***
6:	***name:*** *task-1 # Set waktu & aktifkan SNTP client*
7:	*commands:*
8:	*/set time*
9:	*/set NTP client*
10:	***name:*** *task-2 # Tambah script mekanisme pencadangan konfigurasi*
11:	*commands:*
12:	*/set script backup file .backup*
13:	*/set script backup file .rsc*
14:	***name:*** *task-3 # Eksekusi script pencadangan konfigurasi*
15:	*commands:*
16:	*/run script backup file .backup*
17:	*/run script backup file .rsc*
18:	***name:*** *task-4 # Tambah scheduler*
19:	*commands:*
20:	*/set scheduler script backup file .backup*
21:	*/set scheduler script backup file .rsc*
22:	…

Playbook 2 berisi *task* yang berfokus pada penonaktifan servis, fitur, dan paket dari sistem yang tidak digunakan, serta mengaktifkan fitur keamanan tambahan pada sistem, sehingga dapat memperkecil celah keamanan siber, seperti memvalidasi *routing* dan perlindungan dari serangan DoS (Tabel 4). *Router* perlu untuk memuat ulang sistem setelah proses penonaktifan paket sistem, agar paket sistem dapat dinonaktifkan secara sempurna.

Tabel 4 Algoritma Playbook 2

PB-2.yaml	
1:	---
2:	***name:*** *Penonaktifan fitur atau service yang tidak digunakan untuk memperkecil attack surface*
3:	*target_routers*
4:	*vars_file*
5:	***task:***
6:	***name:*** *task-1 # Menonaktifkan services*
7:	*commands:*
8:	*/disable services*
9:	***name:*** *task-2 # Menonaktifkan fitur bandwidth test server*
10:	*commands:*
11:	*/disable btest*
12:	***name:*** *task-3 # Menonaktifkan MAC server, MAC Winbox, dan MAC Ping*
13:	*commands:*
14:	*/disable fitur MAC*
15:	***name:*** *task-4 # Menonaktifkan IP Cloud, DNS Server, NDP, Socks, dan Proxy*
16:	*commands:*
17:	*/disable fitur IP Cloud, DNS Server, NDP, Socks, dan Proxy*
18:	***name:*** *task-5 # Menonaktifkan fitur Watchdog*
19:	*commands:*
20:	*/disable watchdog*
21:	***name:*** *task-6 # Menonaktifkan packages*
22:	*commands:*
23:	*/disable packages*
24:	***name:*** *task-7 # Menonaktifkan fitur IP forwarding & mengaktifkan fitur keamanan*
25:	*commands:*
26:	*/disable IP forwarding*
27:	*/enable routing filtering*
28:	*/enable tcp-syn cookies*
29:	*...*

Playbook 3 berfokus pada pengecekan ketersediaan pembaruan sistem (Tabel 5). Keluaran yang dihasilkan dari playbook ini berupa notifikasi terkait status kebaruan sistem. Jika hasil yang diperoleh menunjukkan tersedianya versi terbaru, maka Playbook 4 perlu dieksekusi, agar *Router* dapat melakukan pembaruan sistem. Namun jika hasilnya tidak ditemui adanya versi terbaru, maka hal ini menunjukkan bahwa sistem yang digunakan oleh *Router* saat ini adalah versi yang terbaru, sehingga Playbook 4 (Tabel 6) tidak perlu dieksekusi dan lanjut ke tahap automasi berikutnya yaitu mengeksekusi Playbook 5, (Tabel 7) agar *Router* dapat memuat ulang sistem untuk menonaktifkan paket sistem secara sempurna.

Tabel 5 Algoritma Playbook 3

PB-3.yaml	
1:	---
2:	***name:*** *Pengecekan keterbaruan sistem*
3:	*target_routers*
4:	*vars_file*
5:	***task:***
6:	***name:*** *task-1 # Cek kebaruan sistem*
7:	*commands:*
8:	*/cek updates*
9:	*/set NTP client*
10:	***name:*** *task-2 # Menampilkan status*
11:	*commands:*
12:	*/show status*
13:	*...*

Playbook 4 berfokus pada pembaruan sistem yang bertujuan untuk menutup celah keamanan pada sistem operasi terdahulu (Tabel 6). Playbook ini berisi *task* yang menjalankan tindakan berupa mengunduh file sistem versi terbaru, kemudian memuat ulang sistem, dan di tahap akhir menginstal pembaruan sistem. Adapun Playbook 5 hanya berisi *task* untuk memuat ulang sistem pada *Router* (Tabel 7). Memuat ulang sistem diperlukan jika tidak ditemukan adanya pembaruan sistem, yang mana sebelumnya dilakukan pada task Playbook 3 (Tabel 5).

Tabel 6 Algoritma Playbook 4

PB-4.yaml	
1:	---
2:	***name:*** *Instal pembaruan sistem*
3:	*target_routers*
4:	*vars_file*
5:	***task:***
6:	***name:*** *task-1 # Instal sistem*
7:	*commands:*
8:	*/instal sistem*
9:	*...*

Tabel 7 Algoritma Playbook 5

PB-5.yaml	
1:	---
2:	***name:*** *Memuat ulang sistem untuk instal pembaruan sistem*
3:	*target_routers*
4:	*vars_file*
5:	***task:***
6:	***name:*** *task-1 # Memuat ulang sistem*
7:	*commands:*
8:	*/Memuat ulang sistem*
9:	*...*

Task pada Playbook 6 berfokus pada pengamanan akses yang masuk kedalam *Router* (Tabel 8). Pengamanan akses dibutuhkan untuk melindungi *Router* dari serangkaian upaya dan tindakan

penyerangan terhadap akses secara ilegal. Adapun pengamanan akses yang dimaksud meliputi pendaftaran IP *address* yang diizinkan untuk mengakses *Router* (*Whitelist* IP *address*) via servis yang aktif dan akun pengguna sistem, sehingga *Router* dapat menolak akses yang berasal dari selain IP *address* tersebut. Berikutnya adalah pengamanan akses SSH dengan menerapkan fitur *strong-crypto*, agar SSH menggunakan algoritma enkripsi yang lebih kuat. Bagian terakhir pada task ini adalah Pengamanan *Simple Network Management Protocol* (SNMP). SNMP umum digunakan untuk memantau perangkat jaringan, maka protokol ini perlu diamankan. Diantaranya adalah dengan menerapkan *whitelist* IP *address*, autentikasi, otorisasi, enkripsi, hingga penggunaan SNMP versi terbaru.

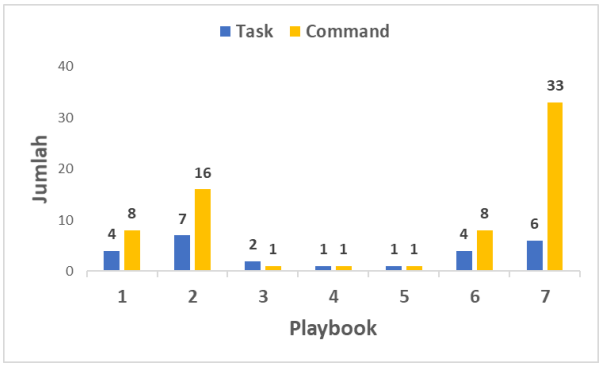
Tabel 8 Algoritma Playbook 6

	PB-6.yaml
1:	---
2:	***name:*** *Pengamanan akses*
3:	*target_routers*
4:	*vars_file*
5:	***task:***
6:	***name:*** *task-1 # Whitelist IP address pada service*
7:	*commands:*
8:	*/set whitelist IP on services*
9:	***name:*** *task-2 # Pengamanan akses SSH*
10:	*commands:*
11:	*/set strong-crypto*
12:	***name:*** *task-3 # Whitelist IP address pada akun pengguna*
13:	*commands:*
14:	*/set whitelist IP on user*
15:	***name:*** *task-4 # Pengamanan SNMP*
16:	*commands:*
17:	*/set secure SNMP*
18:	*/set whitelist IP on SNMP*
19:	…

Playbook 7 berisi *task* yang berfokus pada tindakan pengamanan tambahan menggunakan fitur *firewall* pada *Router* berbasis *Filter Rules* dan Raw (Tabel 9). Hal ini bertujuan untuk melindungi *Router* dari beberapa serangan siber yang kerap terjadi, diantaranya perlindungan dari *port scanning*, *bruteforce*, *Denial of Service* (DoS), pencurian data *user* dan *password* akses *Router* dengan mengeksploitasi celah keamanan WinboxExploit, dan penyebaran *malware* melalui beberapa *port*, seperti *ransomware* wannacry. Adapun rekapitulasi terkait jumlah *task* dan *command* pada masing-masing Playbook diperlihatkan pada Gambar 3.

Tabel 9 Algoritma Playbook 7

	PB-7.yaml
1:	---
2:	***name:*** *Pengamanan tambahan dengan fitur firewall Router berbasis Filter Rules dan RAW*
3:	*target_routers*
4:	*vars_file*
5:	***task:***
6:	***name:*** *task-1 # Memblokir Port Scanning*
7:	*commands:*
8:	*/set raw-rule block port-scanning*
9:	***name:*** *task-2 # Memblokir SSH Bruteforce*
10:	*commands:*
11:	*/set filter-rule block ssh-bruteforce*
12:	***name:*** *task-3 # Memblokir Winbox Bruteforce*
13:	*commands:*
14:	*/set filter-rule block winbox bruteforce*
15:	***name:*** *task-4 # Memblokir DoS*
16:	*commands:*
17:	*/set raw-rule block dos*
18:	***name:*** *task-5 # Memblokir Winbox Exploit*
19:	*commands:*
20:	*/set raw-rule block winbox-exploit*
21:	***name:*** *task-6 # Memblokir Malicious Port*
22:	*commands:*
23:	*/set raw-rule block malicious-port*
24:	…



Gambar 3 Jumlah Task dan Command pada Playbook

b) Implementasi Desain IaC

Masing-masing Ansible playbook diimplementasi secara bertahap dan berurutan pada sisi Ansible *Control Node*. Setiap task automasi pada Playbook 1 berjalan dengan sukses, tanpa ditemukan adanya eror. Hasil yang diperoleh antara lain: 1) IP *address* NTP server berhasil ditambahkan dan fitur SNTP *client* berhasil diaktifkan, serta zona waktu berhasil disesuaikan. 2) *Script* pencadangan konfigurasi berhasil ditambahkan. 3) Script pencadangan konfigurasi berhasil dijalankan dan *Router* mampu untuk melakukan pencadangan konfigurasi dalam dua format pencadangan (.backup dan .rsc), serta *Router* berhasil mengirim pencadangan tersebur ke file server melalui SFTP. File pencadangan konfigurasi sudah diuji melalui pengetesan pemulihan konfigurasi. Hasil

yang diperleh menunjukkan bahwa file tersebut berhasil dipulihkan dengan sempurna, hal ini menunjukkan bahwa kondisi file tersebut dalam keadaan utuh dan tidak rusak. 4) *Scheduler* berhasil dibuat. Durasi yang dihabiskan untuk mengeksekusi Playbook 1 adalah 1 menit 16 detik, dengan *task* yang paling banyak menghabiskan waktu adalah pada *task* ke-3 yaitu proses *backup* konfigurasi dan kirim ke *file server*. Rekap seluruh aktivitas Playbook 1 terlihat pada *log Router.*

Implementasi Playbook 2 berhasil dilakukan, tanpa adanya kegagalan. Hal ini ditunjukkan dari: 1) Servis yang tidak digunakan berhasil dinonaktifkan. 2) Fitur yang tidak digunakan (*Bandwidth Test Server*, MAC server, MAC Winbox, MAC Ping, IP *Cloud*, DNS Server, NDP, Socks, Proxy, Watchdog, dan IP Forwarding) berhasil dinonaktifkan. 3) Fitur keamanan berhasil diaktifkan, fitur yang dimaksud antara lain: RP-Filter untuk memvalidasi *routing* dan TCP-SYN *Cookies* untuk memproteksi dari serangan DoS bertipe SYN *Flooding.* 4) Paket sistem yang tidak digunakan berhasil dinonaktifkan dan *Router* perlu memuat ulang sistem untuk menyempurnakannya. Adapun seluruh aktivitas Playbook ini tercatat juga pada *log Router*. Durasi yang dihabiskan dalam mengeksekusi Playbook 2 adalah selama 50 detik.

Playbook 3 berhasil diimplementasi tanpa adanya eror. Status pengecekan ketersediaan pembaruan sistem berhasil ditampilkan, sehingga memungkinkan untuk menginstal pembaruan sistem dengan mengimplementasi Playbook 4. Playbook ini berhasil dieksekusi dengan durasi selama 29 detik.

Playbook 4 berhasil diimplementasi dengan durasi sekitar 20 detik. Proses unduh dan instal pembaruan sistem menghabiskan waktu sekitar 4 menit (bergantung pada kecepatan koneksi pada saat mengunduh). *Router* berhasil mengunduh sistem terbaru, lalu memuat ulang sistem, dan menginstal pembaruan sistem dari versi awal *Router*OS 6.45.1 menjadi 6.49.5. Hal ini terlihat dari log sistem dan *resource Router*.

Playbook 5 berhasil diimplementasi dengan durasi 16 detik. *Router* berhasil melakukan memuat ulang sistem dan paket sistem yang tidak digunakan berhasil dinonaktifkan dengan sempurna selama 10-15 detik.

Playbook 6 berhasil diimplementasi tanpa adanya eror, dalam kurun waktu 38 detik. Semua task berhasil dijalankan dan dibuktikan dari: 1) *Whitelist* IP *address* berhasil diset pada servis yang aktif. 2) Pengamanan SSH berhasil dilakukan dengan mengaktifkan fitur kriptografi yang kuat dan menolak akses SSH tanpa kriptografi. 3) *Whitelist* IP *address* berhasil diset pada akun pengguna yang aktif. 4) Pengamanan SNMP berhasil dilakukan dengan menerapkan SNMP versi 3, autentikasi, otorisasi, dan enkripsi, serta *whitelist* IP *address* yang diizinkan mengakses SNMP. Setiap aktivitas ditampilkan pada log sistem.

Adapun pada Playbook 7, semua task berhasil diimplementasi. Hal ini terlihat dari rule berbasis *Filter rule* dan Raw *rule* berhasil diset. Seluruh aktivitas dari setiap *task* tercatat pada *log Router* dan waktu yang dibutuhkan untuk mengimplementasi Playbook ini adalah selama 39 detik.

c) Pengujian Desain IaC

Pengujian keamanan sistem dilakukan untuk mengetahui tingkat keberhasilan dan keefektifan dari *hardening* tersebut. Adapun pengujian ini dilakukan dengan enam tes. Tes pertama adalah *Port Scanning*, yang bertujuan untuk mengumpulkan informasi dari target terkait *port*, servis, dan informasi informasi seputar Sistem Operasi, sehingga target dapat dieksploitasi lebih lanjut. Perangkat yang digunakan adalah Nmap. Hasil yang diperoleh menunjukkan bahwa serangan ini berhasil dimitigasi. Hal ini dibuktikan dari Nmap yang tidak berhasil memperoleh informasi apapun dari target.

Tes kedua adalah *Illegal access* dengan menggunakan metode *Bruteforce*. Bertujuan melakukan percobaan berulang untuk masuk kedalam sistem melalui servis SSH hingga mendapatkan akses yang sesuai. Perangkat yang digunakan adalah Hydra. Hasil yang diperoleh menunjukkan bahwa serangan ini berhasil diblokir. Tindakan ini berhasil dilakukan mulai dari tahap awal, saat *Bruteforce* melakukan kegagalan masuk ke sistem pada lima percobaan pertama. Pada tahapan akhir, IP *address* yang menjadi sumber serangan dimasukkan kedalam *blacklist*.

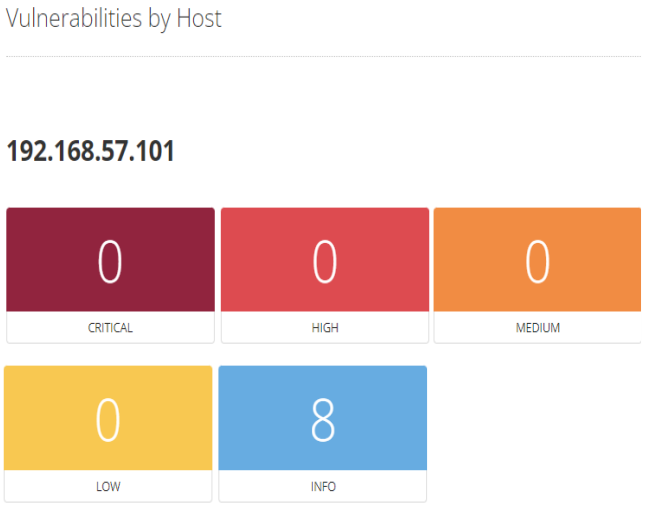
Tes ketiga adalah *Recursive* DNS dengan menggunakan Nmap. Tes ini bertujuan untuk mendeteksi kerentanan pada DNS Server. Kerentanan ini memanfaatkan DNS *Server* sebagai proksi untuk melakukan serangan DoS. Hasil yang diperoleh menunjukkan bahwa kerentanan ini tidak ditemukan.

Tes keempat adalah WinboxExploit, bertujuan untuk mengekploitasi kerentanan CVE-2018-14847 yang bersifat kritikal. Kerentanan ini menargetkan *arbitrary file* (user.dat) yang berisikan informasi dari akses untuk masuk ke sistem. Pengujian ini

menggunakan *script* Python untuk membaca *arbitrary file* dalam bentuk *clear text*. Hasil dari pengujian ini menunjukkan bahwa kerentanan ini tidak ditemukan, karena *arbitrary file* tidak berhasil terbaca.

Tes kelima adalah DoS *Attack*, untuk membanjiri target dengan paket data selama 5 menit. DoS *Attack* menggunakan perangkat Hping3, dengan tipe TCP *Flood* dan SYN *Flood*. Hasil pengujian menunjukkan bahwa serangan DoS berhasil diminimalisir, terlihat dari konsumsi CPU turun hingga 32%.

Tes keenam adalah *Vulnerability Assessment* (VA), bertujuan untuk melakukan pemindaian celah keamanan sistem maupun miskonfigurasi pada *Router*. Adapun *tools* yang digunakan adalah Nessus. Hasil yang diperoleh menunjukkan bahwa tidak ditemukan celah keamanan pada *Router*, baik pada tingkat *Critical*, *High*, *Medium*, maupun *Low*. (Gambar 4). Adapun rekapitulasi dari semua rangkaian pengujian keamanan tersedia pada Tabel 10.



Gambar 4 Hasil *Vulnerability Assessment Test*

Tabel 10 Rekapitulasi Hasil Uji Keamanan Sistem

Pengujian	Hasil
Port Scanning	Berhasil terblokir
Bruteforce	Berhasil terblokir
Recursive DNS	Berhasil terblokir
WinboxExploit, CVE-2018-14847	Berhasil terblokir
DoS *Attack*	Berhasil terblokir
Vulnerability Assessment	Tidak ditemukan celah keamanan

KESIMPULAN

Seluruh desain IaC dari *hardening Router* melalui automasi berbasis Ansible Playbook berhasil diimplementasi pada lima unit *Router* dan tanpa ditemui adanya eror. Adapun total durasi pelaksanaan automasi *hardening* adalah 4 menit 28 detik. Pada pengujian keamanan sistem dengan serangkaian serangan siber diperoleh hasil bahwa keamanan sistem pada *Router* berhasil terbangun, dibuktikan dengan keberhasilan *Router* untuk merespon dan menanggulangi setiap serangan siber yang diujikan dengan optimal.

DAFTAR PUSTAKA

Agus, I. P., & Pratama, E. (2021). Infrastructure as Code (IaC) Menggunakan OpenStack untuk Kemudahan Pengoperasian Jaringan Cloud Computing (Studi Kasus: Smart City di Provinsi Bali) Infrastructure as Code (IaC) Using OpenStack for Ease of Operation of Cloud Computing Network (Case Study . *Jurnal Ilmu Pengetahuan dan Teknologi Komunikasi*, *23*(1), 93–105.

Akin, T. (2002). *Hardening Cisco Routers* (J. Sumser (ed.)). O'Reilly Media.

Bahnasse, A., Bensalah, F., Louhab, F. E., Khiat, A., Khiat, Y., & Talea, M. (2019). Automation of network simulation: concepts related to IPv4 and IPv6 convergence. *Procedia Computer Science*, *155*(2018), 456–461. https://doi.org/10.1016/j.procs.2019.08.063

Ceron, J. M., Scholten, C., Pras, A., & Santanna, J. (2020). MikroTik Devices Landscape, Realistic Honeypots, and Automated Attack Classification. *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 1–9. https://doi.org/10.1109/NOMS47738.2020.9110336

Christanto, F. W., & Suprayogi, M. S. (2017). Pemantauan Sumber Daya Virtual Server pada Cloud Computing Universitas Semarang Menggunakan Network Monitoring System. *Simetris : Jurnal Teknik Mesin, Elektro dan Ilmu Komputer*, *8*(2), 629. https://doi.org/10.24176/simet.v8i2.1555

CISA. (2020). *Security Tip (ST18-001) Securing Network Infrastructure Devices*. CISA. https://www.cisa.gov/uscert/ncas/tips/ST18-001

Dalla Palma, S., Di Nucci, D., Palomba, F., & Tamburri, D. A. (2020). Toward a catalog of software quality metrics for infrastructure code. *Journal of Systems and Software*, *170*, 110726. https://doi.org/10.1016/j.jss.2020.110726

Dalla Palma, S., Di Nucci, D., & Tamburri, D. A. (2020). AnsibleMetrics: A Python library for measuring Infrastructure-as-Code blueprints in Ansible. *SoftwareX*, *12*, 100633.

https://doi.org/10.1016/j.softx.2020.100633

Haeruddin, H. (2021). Analisa dan Implementasi Sistem Keamanan *Router* Mikrotik dari Serangan Winbox Exploitation, Brute-Force, DoS. *JURNAL MEDIA INFORMATIKA BUDIDARMA*, *5*(3), 848. https://doi.org/10.30865/mib.v5i3.2979

Haris, A. I., Riyanto, B., Surachman, F., & Ramadhan, A. A. (2022). Analisis Pengamanan Jaringan Menggunakan *Router* Mikrotik dari Serangan DoS dan Pengaruhnya Terhadap Performansi. *Komputika : Jurnal Sistem Komputer*, *11*(1), 67–76. https://doi.org/10.34010/komputika.v11i1.5227

Hariyadi, I. P., & Marzuki, K. (2020). Implementation Of Configuration Management Virtual Private Server Using Ansible. *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, *19*(2), 347–357. https://doi.org/10.30812/matrik.v19i2.724

Islami, M. F., Musa, P., & Lamsani, M. (2020). Implementation of Network Automation using Ansible to Configure Routing Protocol in Cisco and Mikrotik *Router* with Raspberry PI. *Jurnal Ilmiah Komputasi*, *19*(2), 127–134. https://doi.org/10.32409/jikstik.19.2.80

Jeni Rahman, Azhari, M. L., Tamba, S. R., Ramadhan, A. N., Fakhriyah, I., Hilmi, M. A., Hartadi, E. E., & Kristallia, R. (2022). *Laporan Tahunan Hasil Monitoring Keamanan Siber Tahun 2021*.

Khumaidi, A. (2021). Implementation of DevOps Method for Automation of Server Management Using Ansible. *Jurnal Transformatika*, *18*(2), 199. https://doi.org/10.26623/transformatika.v18i2.2447

Kokuryo, S., Kondo, M., & Mizuno, O. (2020). An Empirical Study of Utilization of Imperative Modules in Ansible. *Proceedings - 2020 IEEE 20th International Conference on Software Quality, Reliability, and Security, QRS 2020*, 442–449. https://doi.org/10.1109/QRS51102.2020.00063

MikroTik. (2019). *Manual: Securing Your Router*. Wiki MikroTik. https://wiki.mikrotik.com/wiki/Manual:Securing_Your_*Router*

Mohd Fuzi, M. F., Abdullah, K., Abd Halim, I. H., & Ruslan, R. (2021). Network Automation using Ansible for EIGRP Network. *Journal of Computing Research and Innovation*, *6*(4), 59–69. https://doi.org/10.24191/jcrinn.v6i4.237

Pambudi, R., & Muslim, M. A. (2017). Implementasi Policy Base Routing dan Failover Menggunakan *Router* Mikrotik untuk Membagi Jalur Akses Internet di FMIPA Unnes. *Jurnal Teknologi dan Sistem Komputer*, *5*(2), 57. https://doi.org/10.14710/jtsiskom.5.2.2017.57-61

Perera, H. M. D. G. V., Samarasekara, K. M., Hewamanna, I. U. K., Kasthuriarachchi, D. N. W., Abeywardena, K. Y., & Yapa, K. (2021). NetBot - An Automated *Router* Hardening Solution for Small to Medium Enterprises. *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 0015–0021. https://doi.org/10.1109/IEMCON53756.2021.9623186

Pratama, M. A. A., & Hariyadi, I. P. (2021). Otomasi Manajemen dan Pengawasan Linux Container (LCX) Pada Proxmox VE Menggunakan Ansible. *Jurnal Bumigora Information Technology (BITe)*, *3*(1), 82–95. https://doi.org/10.30812/bite.v3i1.807

Rifki Afandi, M., Hatta, P., Efendi, A., Kunci-Otomatisasi Jaringan, K., Komputer, L., & Jaringan, P. (2020). Otomatisasi Perangkat Jaringan Komputer Menggunakan Ansible Pada Laboratorium Komputer. *SMARTICS Journal*, *6*(2), 48–53.

Spichkova, M., Li, B., Porter, L., Mason, L., Lyu, Y., & Weng, Y. (2020). VM2: Automated security configuration and testing of virtual machine images. *Procedia Computer Science*, *176*, 3610–3617. https://doi.org/10.1016/j.procs.2020.09.025

Swastika, I. M. B., & Atitama, I. G. O. G. (2017). Otomatisasi Konfigurasi Mikrotik *Router* Menggunakan Software Ansible. *Internet of Think (IoT) & Big Data : Teknologi, Tantangan dan Peluang*, 495–502.

Tantoni, A., Ashari, M., & Zaen, M. T. A. (2020). Analisis Dan Implementasi Jaringan Komputer Brembuk.Net Sebagai Rt/Rw.Net Untuk Mendukung E-Commerce Pada Desa Masbagik Utara. *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, *19*(2), 312–320. https://doi.org/10.30812/matrik.v19i2.591

Wilkins, S. (2011). *Cisco's PPDIOO Network Cycle*. Cisco Press. https://www.ciscopress.com/articles/article.asp?p=1697888

Halaman ini sengaja dikosongkan