Journal of Production, Enterprise, and Industrial Applications
Vol. 3, Issue 2, pp. 94–101 (2025)
doi: http://doi.org/10.25124/jpeia.v3i2.6467

RESEARCH ARTICLE

Implementasi dan Analisa *Security Hardening* pada Server Linux Menggunakan Ansible

Ramadani, Adityas Widjajarto* and Umar Yunan Kurnia Septo Hediyanto

Fakultas Rekayasa Industri, Universitas Telkom, Bandung, 40257, Jawa Barat, Indonesia
*Corresponding author: adtwjrt@telkomuniversity.ac.id

Abstrak

Otomatisasi proses *hardening* keamanan jaringan semakin penting dalam menghadapi ancaman siber yang kian kompleks dan dinamis. Penelitian ini mengeksplorasi efektivitas otomatisasi menggunakan Ansible dibandingkan dengan metode manual dalam penerapan *security hardening* pada sistem operasi Linux Ubuntu. Fokus utama penelitian ini adalah mengevaluasi efisiensi waktu, konsistensi, dan kemudahan penerapan kebijakan keamanan antara kedua metode tersebut. Metode penelitian melibatkan tahapan identifikasi masalah, formulasi hipotesis, serta pengujian perbandingan waktu antara metode manual dan otomatis, disertai analisis hasil. Proses *hardening* dilakukan pada beberapa *virtual machine* (VM) dengan menggunakan Ansible playbook dan metode manual. Konfigurasi manual melibatkan penerapan pengaturan keamanan secara langsung melalui antarmuka sistem, sementara konfigurasi otomatis menggunakan Ansible menerapkan kebijakan keamanan secara serentak melalui playbook. Hasil penelitian menunjukkan bahwa Ansible menyelesaikan tugas *hardening* secara signifikan lebih cepat, seperti konfigurasi *firewall* yang hanya memakan waktu 10 detik dan 92 milidetik, dibandingkan dengan metode manual yang memakan waktu 1 menit, 38 detik, dan 45 milidetik. Meski setup awal Ansible membutuhkan 16 menit dan 18,99 detik, hasil ini menegaskan bahwa otomatisasi dengan Ansible tidak hanya mempercepat proses *hardening* secara keseluruhan, tetapi juga mengurangi risiko kesalahan manusia dan memastikan konsistensi dalam penerapan kebijakan keamanan, terutama dalam skala besar.

Key words: *Hardening*, Otomasi Ansible, SSH, *Firewall*, Waktu.

Pendahuluan

Dalam era modern ini, kompleksitas dan dinamika keamanan siber menghadirkan tantangan serius. Serangan siber, termasuk *ransomware* yang merugikan secara ekonomi, pelanggaran data pribadi, dan ancaman siber yang semakin canggih, telah menjadi perhatian utama. Ancaman-ancaman ini berkembang seiring dengan kemajuan teknologi, sehingga muncul kerentanan baru yang dapat dieksploitasi oleh pihak-pihak yang berniat jahat. Sebagai tanggapan, langkah-langkah keamanan siber menjadi sangat penting untuk melindungi sistem komputer, termasuk perangkat keras dan perangkat lunak, dari ancaman, gangguan, dan serangan. Keamanan siber berfokus pada perlindungan informasi yang terdapat dalam sistem serta elemen-elemen lain di dalam ruang siber. Ia berfungsi sebagai pertahanan kritis terhadap pemantauan yang tidak sah dan kegiatan intrusif lainnya, dengan tujuan untuk memastikan ketersediaan, integritas, dan kerahasiaan informasi [1]. Penerapan *security hardening* secara manual, meskipun efektif, sering kali membutuhkan waktu dan tenaga yang signifikan serta rentan terhadap kesalahan manusia. Otomatisasi, di sisi lain, menawarkan solusi yang lebih efisien, memastikan bahwa langkah-langkah keamanan diterapkan secara merata dan cepat di seluruh sistem. Otomatisasi menggunakan Ansible tidak hanya memungkinkan penerapan *hardening* secara konsisten, tetapi juga memungkinkan penerapan kebijakan keamanan di lingkungan yang luas tanpa harus mengelola konfigurasi secara manual di setiap perangkat. Hal ini memberikan efisiensi operasional yang penting, terutama ketika mengelola infrastruktur yang besar dan kompleks.

Dalam konteks ini, penelitian ini bertujuan untuk mengeksplorasi dan membandingkan efektivitas serta efisiensi antara metode *hardening* otomatis menggunakan Ansible dengan metode manual. Eksperimen ini akan dilakukan dalam lingkungan yang terkendali, di mana Ansible digunakan sebagai alat otomatisasi utama untuk menerapkan langkah-langkah keamanan pada *virtual machines* (VM). Studi ini berfokus pada mengidentifikasi manfaat utama yang ditawarkan oleh otomatisasi, seperti penghematan waktu dan peningkatan keseluruhan dalam keamanan sistem, dibandingkan dengan pendekatan manual. Otomatisasi melalui Ansible diharapkan dapat memberikan hasil yang lebih konsisten dan efisien dibandingkan metode manual, sekaligus mengurangi potensi kesalahan manusia yang sering kali menjadi faktor

utama dalam kegagalan penerapan kebijakan keamanan. Tidak hanya di ranah penelitian, penerapan otomatisasi menggunakan Ansible juga telah diadopsi oleh berbagai perusahaan besar seperti IBM, Netflix, dan Adobe. Perusahaan-perusahaan ini menggunakan Ansible untuk mengotomatisasi *security hardening* dan manajemen keamanan di lingkungan yang kompleks dan luas, memastikan bahwa kebijakan keamanan diterapkan secara konsisten dan efisien di seluruh infrastruktur mereka. Implementasi Ansible dalam skala besar oleh perusahaanperusahaan tersebut semakin menegaskan relevansi dan urgensi penelitian ini. Melalui penelitian ini, diharapkan dapat dipahami lebih jauh bagaimana otomatisasi, khususnya dengan Ansible, dapat berkontribusi signifikan dalam peningkatan keamanan sistem dan efisiensi operasional, serta mengatasi tantangan yang dihadapi dalam metode manual. Tujuan dari eksperimen ini adalah untuk mengevaluasi dampak otomatisasi terhadap kecepatan dan akurasi proses *security hardening*, serta untuk menilai sejauh mana otomatisasi dapat meningkatkan keamanan sistem dibandingkan dengan metode manual. Hasil dari penelitian ini diharapkan dapat memberikan wawasan yang mendalam mengenai aplikasi praktis dari otomatisasi Ansible dalam upaya *hardening* keamanan dan potensi pengaruhnya dalam meningkatkan postur keamanan jaringan. Selain itu, penelitian ini dapat menjadi landasan bagi pengembangan lebih lanjut dalam penggunaan otomatisasi untuk keamanan jaringan, membuka jalan bagi penerapan teknologi yang lebih luas di berbagai sektor industri yang membutuhkan keamanan yang handal dan efisien.

Tinjauan Pustaka

Automation IT

Otomatisasi meningkatkan efisiensi dan kualitas dalam pengelolaan sistem dengan meminimalkan intervensi manusia. Sistem otomatisasi, seperti Ansible, memungkinkan konfigurasi dan manajemen sistem server secara konsisten dan efisien [2].

Ansible

Ansible adalah alat manajemen konfigurasi yang memungkinkan otomatisasi proses konfigurasi dan pengelolaan sistem pada server, memastikan penerapan kebijakan keamanan yang konsisten dan efisien [3].

Hardening

Security Hardening mencakup berbagai metode untuk memperkuat keamanan sistem, seperti pengamanan pada tingkat kode, proses perangkat lunak, desain, dan lingkungan operasional, guna mencegah eksploitasi dan memperbaiki kerentanan yang ada [4].

SSH

Secure Shell (SSH) adalah sebuah protokol untuk login jarak jauh yang aman serta layanan jaringan aman lainnya melalui jaringan yang tidak aman. Dokumen ini menjelaskan protokol lapisan transport SSH, yang biasanya berjalan di atas TCP/IP. Protokol ini dapat digunakan sebagai dasar untuk sejumlah layanan jaringan yang aman. SSH menyediakan enkripsi yang kuat, otentikasi server, dan perlindungan integritas. Selain itu, SSH juga dapat menyediakan kompresi. Metode pertukaran kunci, algoritma kunci publik, algoritma enkripsi simetris, algoritma otentikasi pesan, dan algoritma hash semuanya dinegosiasikan. Dokumen ini juga menjelaskan metode pertukaran kunci Diffie-Hellman dan set minimal algoritma yang diperlukan untuk mengimplementasikan protokol lapisan transport SSH [5].

Gambar 1. Model Konseptual

Firewall

Kebutuhan akan Keamanan Jaringan semakin meningkat seiring dengan peningkatan penggunaan Internet. Keamanan Jaringan mencegah akses tidak sah, peretasan, dan memastikan transportasi data yang autentik. Keamanan Jaringan mencakup ketentuan dan kebijakan yang diadopsi oleh seorang administrator jaringan untuk mencegah dan memantau akses yang tidak sah, perubahan, penyimpangan, penolakan terhadap jaringan komputer, dan sumber daya yang dapat diakses melalui jaringan. Keamanan Jaringan dicapai dengan *Firewall*. *Firewall* adalah perangkat keras atau perangkat lunak yang dirancang untuk mengizinkan atau menolak transmisi jaringan berdasarkan protokol tertentu. *Firewall* terletak pada titik akhir sistem yang menyaring semua lalu lintas dan pengguna yang tidak sah. Namun, *firewall* konvensional atau tradisional sangat bergantung pada topologi yang terbatas dan titik masuk yang terkendali untuk berfungsi, yang mengakibatkan kesulitan dalam memfilter protokol tertentu, masalah enkripsi *end-to-end*, dan sebagainya. Oleh karena itu, hal ini mendorong perkembangan *Firewall* Terdistribusi yang memperkuat kebijakan keamanan jaringan tanpa membatasi topologinya dari dalam atau luar. *Firewall* Terdistribusi adalah aplikasi perangkat lunak keamanan yang berlokasi di host yang melindungi server jaringan perusahaan dan mesin pengguna akhir dari intrusi yang tidak diinginkan. Makalah ini adalah makalah tinjauan literatur yang berfokus pada *firewall* tradisional, perkembangannya, masalah keamanan, berbagai kebijakan, dan konsep *firewall* terdistribusi [6].

Audit Daemon (auditd)

Audit Daemon (auditd) adalah komponen penting dalam *Linux Auditing System* yang berfungsi untuk mencatat aktivitas sistem secara rinci, terutama terkait keamanan. Auditd bekerja dengan mengumpulkan log peristiwa yang mencakup akses dan modifikasi file penting, perubahan hak akses, serta aktivitas pengguna dan proses. Log ini digunakan untuk memantau, menganalisis, dan memastikan kepatuhan terhadap kebijakan keamanan, serta mendukung investigasi forensik setelah insiden keamanan. Meskipun dapat menghasilkan volume data yang besar dan membutuhkan konfigurasi yang cermat, auditd memberikan wawasan kritis bagi administrator dalam memperkuat keamanan sistem Linux.

Metodologi Penelitian

Model Konseptual

Model konseptual dalam penelitian ini menggambarkan hubungan antara variabel yang digunakan untuk mengukur efektivitas metode *hardening*, baik secara manual maupun otomatis dengan Ansible, dalam meningkatkan keamanan dan mempercepat respons sistem yang ditunjukan pada gambar 1.

Sistematika Penelitian

Penelitian ini menggunakan metode *Security Hardening* yang diotomatisasi dengan Ansible dan dibandingkan dengan metode manual. Tahapan penelitian meliputi identifikasi masalah, perumusan hipotesis, pengujian, analisis data, dan penarikan kesimpulan. Sistematika ini disusun untuk memastikan bahwa setiap langkah penelitian diorganisir secara sistematis untuk mencapai tujuan penelitian.

Tahap Awal

Penelitian dimulai dengan identifikasi masalah, diikuti oleh studi literatur yang relevan. Variabel utama yang diteliti adalah waktu respons dan tingkat keamanan. Rencana implementasi dibuat untuk membandingkan metode manual dan otomatis.

Tahap Hipotesis

Hipotesis dirumuskan berdasarkan studi literatur dan identifikasi variabel. Hipotesis ini akan diuji melalui eksperimen yang dirancang untuk menilai efektivitas kedua metode dalam mempercepat waktu respons dan meningkatkan keamanan.

Tahap Pengujian

Eksperimen dirancang untuk membandingkan waktu respons dan keamanan antara *hardening* otomatis dengan Ansible dan *hardening* manual. Pengujian mencakup berbagai komponen seperti *firewall*, SSH, dan audit sistem. Data dari eksperimen dikumpulkan untuk dianalisis lebih lanjut.

Tahap Analisis

Data yang dikumpulkan dianalisis untuk mengidentifikasi perbandingan waktu respons dan tingkat keamanan antara metode otomatis dan manual. Analisis ini memberikan wawasan mengenai efektivitas kedua metode dalam konteks *hardening* keamanan.

Tahap Akhir

Peneliti menarik kesimpulan dari hasil analisis mengenai efektivitas Ansible dan *hardening* dalam mempercepat respons waktu dan meningkatkan keamanan. Rekomendasi praktis atau untuk penelitian lanjutan disampaikan berdasarkan temuan.

Alasan Pemilihan Metode

Metode pada penelitian ini adalah implementasi *hardening* pada otomasi dan manual, dikarenakan penelitian ini melakukan perbandingan Ansible adalah sebuah *tool open-source* yang digunakan untuk otomasi konfigurasi dan *deployment* sistem. Pemilihan metode *Security Hardening* dalam konteks penggunaan Ansible didasarkan pada beberapa pertimbangan utama yang memastikan keamanan dan efektivitas dalam pengelolaan sistem secara otomatis.

1. Penguatan Penggunaan *Password*
   Ansible memungkinkan penerapan kebijakan *password* yang kuat untuk melindungi koneksi ke sistem. Keamanan ini dapat diatur secara terpusat atau per host, sesuai dengan kebutuhan spesifik, menjamin bahwa setiap akses ke sistem dilindungi dengan mekanisme yang paling aman.
2. Integrasi dengan *Firewall*
   Ansible mendukung integrasi langsung dengan *firewall*, seperti iptables dan *firewall*, yang memungkinkan pengaturan port yang lebih aman. Hanya port yang benar-benar diperlukan yang dibuka, sementara port lainnya ditutup, mengurangi potensi serangan melalui jaringan.

Table 1. Spesifikasi *Hardware*

Komponen	Informasi	
Core Hardware Specification	Processor	11th Gen Intel(R) Core(TM) i7-1185G7 @ 3.00 GHz , 1805 Mhz , 4 Core(s), 8 Logical Processor(s)
	RAM	$16,0$ GB
	Storage	1TB
	System type	x64-based PC
	Operating System	Windows 11 Pro

3. Pemantauan Keamanan
   Metode *hardening* melalui Ansible memungkinkan pemantauan keamanan di seluruh infrastruktur. Ini memastikan bahwa setiap sistem atau perangkat diatur sesuai dengan kebijakan keamanan yang ditetapkan, sehingga meminimalkan celah keamanan.
4. Kemampuan Audit yang Lebih Baik
   Dengan menerapkan *hardening* keamanan, Ansible memberikan kemampuan audit yang lebih baik terhadap konfigurasi keamanan. Ini mencakup manajemen perubahan yang lebih terkontrol, identifikasi penyimpangan dari kebijakan keamanan, dan kemampuan untuk merespons lebih cepat terhadap potensi ancaman.
5. Pengelolaan Kerentanan
   Dengan penerapan metode *hardening*, pengelolaan kerentanan pada sistem dapat dilakukan secara lebih efektif. Termasuk di dalamnya adalah proses instalasi dan penghapusan kerentanan, yang dapat dijalankan dengan memastikan bahwa setiap perubahan atau penghapusan tersebut dilakukan dengan aman dan tidak menimbulkan risiko baru bagi sistem.

Metode ini dipilih karena mampu menyediakan solusi yang komprehensif dalam menjaga keamanan sistem secara proaktif dan efisien, yang sejalan dengan tujuan penelitian untuk membandingkan efektivitas otomasi versus metode manual dalam proses *hardening* keamanan.

Hasil dan Pembahasan

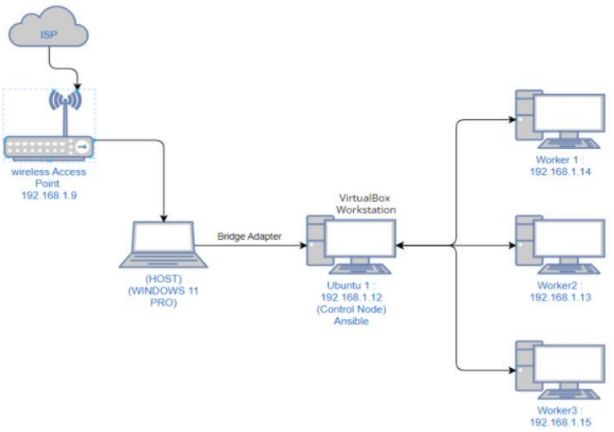
Identifikasi Sistem

Identifikasi sistem dilakukan untuk mencapai tujuan penelitian ini. Diperlukan arsitektur yang mencakup informasi *hardware* dan server sebagai langkah awal untuk melakukan analisis perbandingan pada server VM. *Hardware* dan server yang digunakan dalam penelitian ini dibutuhkan sebagai alat pendukung untuk melakukan perbandingan, pemindaian, dan perbaikan pada lingkungan sistem.

1. *Hardware*
   Penelitian ini menggunakan *hardware* dengan spesifikasi yang dirincikan pada tabel 1 berikut:
2. *Server*
   Spesifikasi *server* yang dipilih untuk penelitian dapat dilihat pada tabel 2 berikut:
3. Topologi Jaringan
   Topologi jaringan yang digunakan pada penelitian ini ditampilkan pada gambar 2 di bawah ini. Topologi ini memungkinkan pengujian *security hardening* dalam lingkungan virtual terstruktur.

Table 2. Spesifikasi *server*

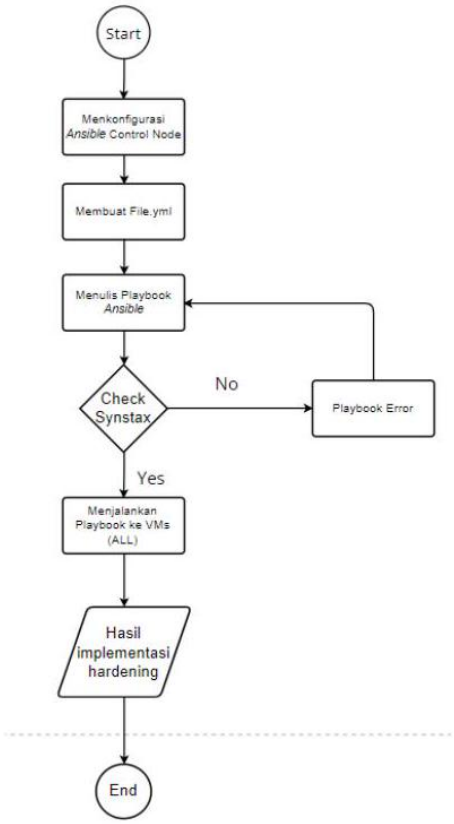
Komponen		Informasi
Server Hardware Specification	Virtual Machine	VirtualBox Workstation-version 7.0.20
	Storage	20 GB
	Operating System (Ubuntu1, Worker1, Worker2, Worker3)	Ubuntu (64bit) 22.04 LTS
		Ubuntu (64bit) 20.04 LTS
		Ubuntu (64bit) 18.04 LTS
		Ubuntu (64bit) 24.04 LTS
	CPU	1 Core
	RAM	2048 MB RAM
	System type	64 bit



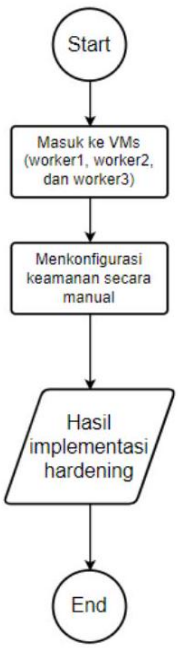
Gambar 2. Topologi Jaringan

Skenario Perbandingan *Hardening* Otomasi dan Manual

1. Uji Coba Skenario Otomasi
   Pengujian ini memvalidasi efektivitas Ansible dalam *security hardening* (Gambar 3):
   Proses otomatisasi *security hardening* menggunakan Ansible dimulai dengan mengkonfigurasi Ansible Control Node, misalnya ubuntu1, untuk mengelola automasi. Setelah itu, playbook Ansible yang mendefinisikan langkah-langkah keamanan yang harus diterapkan ditulis. Selanjutnya, playbook tersebut dijalankan pada semua VMs yang ditargetkan, seperti worker1,worker2, dan worker3, menggunakan Ansible. Setelah playbook dijalankan, verifikasi dan validasi dilakukan untuk memastikan bahwa perubahan keamanan telah diterapkan dengan benar dan sistem berfungsi sesuai yang diharapkan. Proses ini kemudian diakhiri dengan penyelesaian otomatisasi.
2. Uji Coba Skenario Manual
   Sebagaimana ditunjukan dalam gambar 4 dalam uji coba skenario manual untuk *security hardening*, proses dimulai dengan mengakses masing-masing VM (worker1,worker2,worker3). Setelah masuk ke VM, konfigurasi keamanan dilakukan secara manual, mengikuti langkah-langkah yang ditetapkan untuk penguatan keamanan. Setelah konfigurasi selesai, verifikasi dan validasi dilakukan untuk memastikan bahwa pengaturan keamanan diterapkan dengan benar dan sistem berfungsi sebagaimana mestinya. Proses ini kemudian diulang untuk VM lainnya jika ada. Uji coba ini berakhir setelah semua VM telah dikonfigurasi dan diverifikasi, memberikan wawasan mengenai efisiensi dan potensi tantangan dari metode manual dibandingkan dengan otomatisasi.



Gambar 3. Skenario Otomasi



Gambar 4. Skenario Manual

Table 3. *Hardening* ssh-hardening.yml

No	Nama	Hasil Waktu
1	Playbook-Ansible	0 : 13.03
2	Worker 1	0 : 07.77
3	Worker 2	1 : 23.07
4	Worker 3	0 : 05.33
	Total	1 : 36.17

Table 4. *Hardening* configure-firewall.yml

No	Nama	Hasil Waktu
1	Playbook-Ansible	0 : 10.92
2	Worker 1	0 : 08.165
3	Worker 2	1 : 21.68
4	Worker 3	0 : 08.605
	Total	1 : 38.45

Table 5. *Hardening* configure-auditd.yml

No	Nama	Hasil Waktu
1	Playbook-Ansible	0 : 12.03
2	Worker 1	0 : 4.08
3	Worker 2	1 : 34.65
4	Worker 3	0 : 5.34
	Total	1 : 44.79

Implementasi Pengujian *Hardening*

Pengujian dilakukan untuk mengevaluasi efektivitas kedua metode:

1. Manual: Konfigurasi dilakukan secara individual, memakan waktu lebih lama, dan lebih rentan terhadap kesalahan manusia.
2. Otomasi: Menggunakan Ansible untuk mengotomatiskan *hardening*, memastikan konsistensi, dan mengurangi waktu konfigurasi.

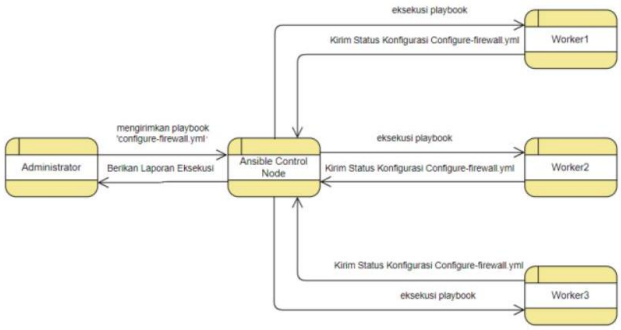
Data yang dihasilkan memberikan gambaran tentang kecepatan, keandalan, dan efisiensi dari kedua metode, membantu menentukan pendekatan terbaik dalam pengelolaan keamanan sistem.

Data Hasil Perbandingan Waktu

Analisis ini membandingkan efisiensi waktu antara *hardening* sistem manual dan otomatis menggunakan Ansible. *Hardening* manual memakan waktu lebih lama dan rentan kesalahan, sedangkan otomatisasi mempercepat dan mengurangi kesalahan.

1. Hasil Data Otomasi dan Manual
   a. *Hardening* ssh-hardening.yml
      Hasil data untuk *Hardening* ssh-hardening.yml ditunjukan pada tabel 3 berikut.
   b. *Hardening* configure-firewall.yml
      Hasil data untuk *Hardening* configure-firewall.yml ditunjukan pada tabel 4 berikut.
   c. *Hardening* configure-auditd.yml
      Hasil data untuk *Hardening* configure-auditd.yml ditunjukan pada tabel 5 berikut.

   Otomatisasi dengan Ansible terbukti lebih efisien dibandingkan metode manual.

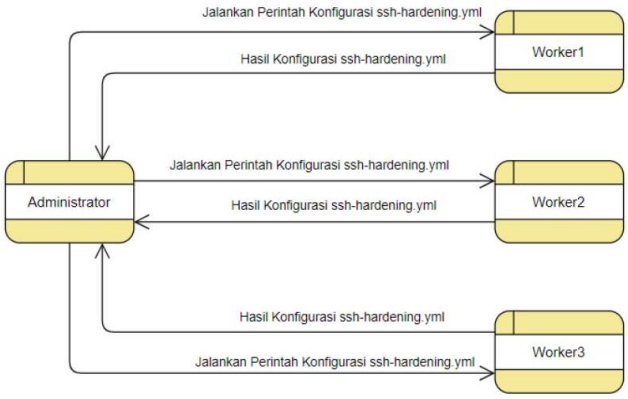


Gambar 5. Configure-firewall.yml

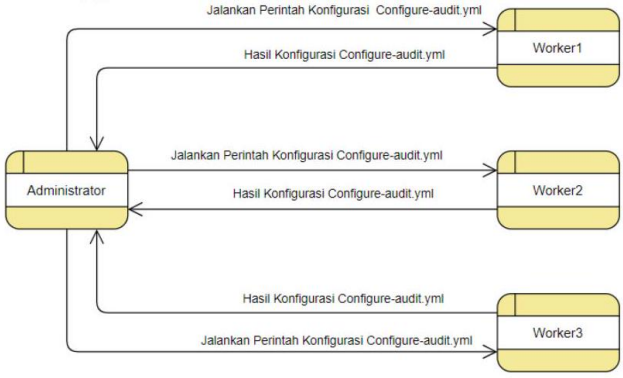
Analisis Hasil Manual *Hardening*

Proses otomatisasi *hardening* menggunakan Ansible dilakukan secara konsisten di semua target nodes. *Data Flow Diagram* (DFD) untuk setiap playbook menunjukkan alur kerja yang efisien dari Administrator ke Ansible Control Node hingga Target Nodes, dengan laporan status akhir dikirimkan kembali ke Administrator.

1. Configure-firewall.yml
   Pada gambar 5 di atas menjelaskan proses dari *Data Flow Diagram* (DFD) untuk manual *hardening* menggunakan 'configure-firewall.yml':
   a. Administrator mempersiapkan konfigurasi di setiap Target Node:
      Administrator secara manual mengakses setiap target node (worker1, worker2, worker3) melalui terminal atau SSH untuk memulai proses *hardening*. Di setiap node, Administrator mempersiapkan lingkungan dengan memastikan semua paket dan sistem diperbarui.
   b. Administrator memperbarui daftar paket dan menginstal UFW:
      Administrator menjalankan perintah sudo apt-get update && apt-get install-y ufw untuk memperbarui daftar paket dan menginstal U*ncomplicated Firewall* (UFW) di setiap target node. Ini adalah langkah awal yang memastikan bahwa firewall sudah terpasang dan siap dikonfigurasi.
   c. Administrator mengonfigurasi kebijakan *default firewall*:
      Administrator secara manual menetapkan kebijakan default untuk *firewall* dengan perintah sudo ufw default deny incoming, yang menginstruksikan *firewall* untuk menolak semua koneksi masuk secara default. Ini adalah langkah penting untuk meningkatkan keamanan dengan memblokir akses yang tidak diinginkan.
   d. Administrator mengizinkan koneksi pada port tertentu:
      Administrator menjalankan perintah sudo ufw allow 22/tcp, sudo ufw allow 80/tcp, dan sudo ufw allow 443/tcp di setiap target node. Perintah ini memungkinkan koneksi yang diperlukan untuk SSH, HTTP, dan HTTPS, yang diperlukan untuk administrasi jarak jauh dan akses web.
   e. Administrator mengaktifkan UFW dan memastikan UFW diatur untuk memulai otomatis:
      Setelah semua aturan diterapkan, Administrator mengaktifkan *firewall* dengan perintah sudo ufw –force enable. Ini memastikan bahwa UFW aktif dan akan memulai otomatis setiap kali sistem booting, menjaga keamanan sistem secara berkelanjutan.
   f. Administrator memverifikasi konfigurasi di setiap Target Node:
      Setelah semua langkah konfigurasi dijalankan, Administrator memverifikasi bahwa *firewall* telah diaktifkan dan berfungsi sebagaimana mestinya di setiap target node. Verifikasi ini dilakukan dengan memeriksa status UFW dan memastikan bahwa aturan-aturan yang telah diterapkan bekerja dengan benar.



Gambar 6. Ssh-hardening.yml



Gambar 7. Ssh-hardening.yml

2.Ssh-hardening.yml

Pada gambar 6 di atas menjelaskan proses dari *Data Flow Diagram* (DFD) untuk manual hardening menggunakan 'sshhardening.yml':

a.Administrator mempersiapkan konfigurasi di setiap Target Node:
Administrator secara manual mengakses setiap target node (worker1, worker2, worker3) melalui terminal atau SSH untuk memulai proses penguatan konfigurasi SSH. Administrator memastikan bahwa lingkungan siap untuk dilakukan konfigurasi keamanan.

b.Administrator menginstal OpenSSH Server:
Administrator menjalankan perintah sudo apt-get install y openssh-server di setiap target node untuk menginstal OpenSSH Server jika belum terpasang. Ini memastikan bahwa layanan SSH tersedia dan siap untuk dikonfigurasi.

c.Administrator memastikan layanan SSH berjalan dan diaktifkan:
Administrator menjalankan perintah sudo systemctl start ssh dan sudo systemctl enable ssh untuk memastikan bahwa layanan SSH dimulai dan diaktifkan secara otomatis setiap kali sistem booting. Ini memastikan bahwa akses SSH tetap tersedia setelah sistem restart.

d.Administrator mengonfigurasi pengaturan SSH:
Administrator mengedit file sshd_config di setiap target node untuk memperkuat pengaturan SSH. Perubahan penting termasuk menonaktifkan login root melalui SSH dengan mengubah parameter PermitRootLogin menjadi "no". Langkah ini mencegah akses root langsung, yang meningkatkan keamanan sistem.

e.Administrator me-restart layanan SSH:
Setelah melakukan perubahan pada konfigurasi SSH, Administrator menjalankan perintah sudo systemctl restart ssh untuk me-restart layanan SSH. Restart ini diperlukan agar perubahan konfigurasi yang baru diterapkan bisa mulai bekerja dan meningkatkan keamanan akses SSH.

f.Administrator memverifikasi konfigurasi di setiap Target Node:
Setelah semua langkah konfigurasi selesai, Administrator memverifikasi bahwa pengaturan SSH telah diterapkan dengan benar di setiap target node. Verifikasi ini dilakukan dengan memeriksa file sshd_config dan memastikan bahwa layanan SSH berjalan sesuai dengan konfigurasi yang diperkuat, seperti menonaktifkan login root.

3.Ssh-hardening.yml

Pada gambar 7 di atas menjelaskan proses dari *Data Flow Diagram* (DFD) untuk manual hardening menggunakan 'configure-audit.yml.':

a.Administrator mempersiapkan konfigurasi di setiap Target Node:
Administrator secara manual mengakses setiap target node (worker1, worker2, worker3) melalui terminal atau SSH untuk memulai proses konfigurasi audit sistem. Administrator memastikan bahwa sistem siap untuk di-*hardening* dengan melakukan audit konfigurasi.

b.Administrator menginstal paket auditd:
Administrator menjalankan perintah sudo apt-get install y auditd di setiap target node untuk menginstal paket auditd. Auditd adalah daemon yang digunakan untuk melacak dan mencatat aktivitas di sistem, sehingga Administrator dapat memantau berbagai tindakan yang dilakukan di server.

c.Administrator mengonfigurasi aturan audit:
Administrator mengedit file konfigurasi audit rules di setiap target node untuk menentukan aktivitas apa saja yang perlu diawasi dan dicatat. Ini dapat mencakup perubahan pada file penting, aktivitas pengguna yang mencurigakan, atau akses ke sumber daya sistem tertentu. Aturan-aturan ini memastikan bahwa aktivitas yang relevan dicatat untuk tujuan keamanan dan kepatuhan.

d.Administrator memastikan layanan auditd berjalan dan diaktifkan:
Administrator menjalankan perintah sudo systemctl start auditd dan sudo systemctl enable auditd untuk memastikan bahwa layanan auditd berjalan dan diaktifkan secara otomatis setiap kali sistem *booting*. Ini memastikan bahwa auditing selalu aktif, bahkan setelah sistem direstart.

e.Administrator me-restart layanan auditd (jika diperlukan):
Jika ada perubahan signifikan dalam konfigurasi audit, Administrator mungkin perlu me-restart layanan auditd dengan menjalankan perintah sudo systemctl restart auditd. Ini memastikan bahwa semua perubahan konfigurasi diterapkan dan audit berfungsi sesuai yang diharapkan.

f.Administrator memverifikasi konfigurasi di setiap Target Node:
Setelah semua konfigurasi selesai, Administrator memverifikasi bahwa layanan auditd berfungsi dengan baik dan aturan audit yang telah ditetapkan bekerja dengan benar di setiap target node. Verifikasi ini dilakukan dengan memeriksa log audit dan memastikan bahwa semua aktivitas yang ditentukan dalam aturan audit dicatat dengan benar, yang membantu dalam pemantauan dan keamanan sistem

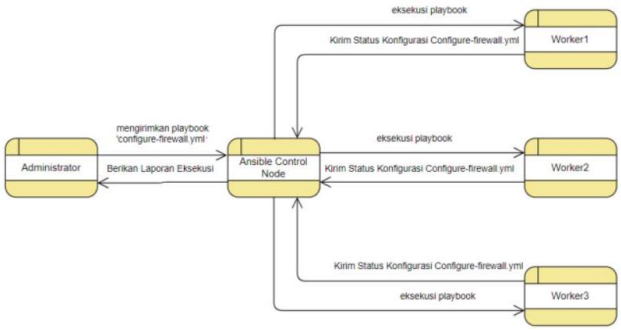
Analisis Hasil Otomasi *Hardening*

Manual *hardening* melibatkan konfigurasi individual pada setiap node. DFD menggambarkan langkah-langkah yang diambil Administrator dari persiapan hingga verifikasi konfigurasi di setiap target node.

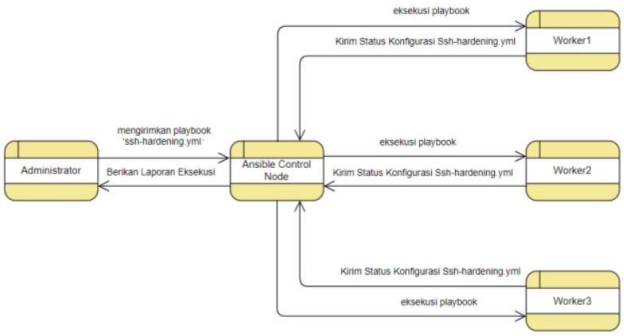
1.Configure-firewall.yml

Pada gambar 8 di atas menjelaskan proses dari *Data Flow Diagram* (DFD) untuk otomasi *hardening* menggunakan playbook Ansible 'configure-firewall.yml':

Administrator membuat dan mengunggah playbook configure-firewall.yml ke Ansible Control Node, yang berisi langkah-langkah



Gambar 8. Ssh-hardening.yml



Gambar 9. Ssh-hardening.yml

konfigurasi *firewall* untuk target nodes. Control Node kemudian menjalankan playbook tersebut dan mendistribusikan perintah ke target nodes (worker1, worker2, worker3), termasuk instalasi UFW, pengaturan kebijakan default, pembukaan port SSH, HTTP, HTTPS, serta pengaktifan *firewall*. Setiap target node menjalankan konfigurasi *firewall* sesuai instruksi, memastikan konfigurasi yang konsisten di seluruh nodes. Setelah konfigurasi selesai, setiap target node mengirimkan laporan status ke Control Node terkait instalasi UFW, penerapan kebijakan, dan aktivasi *firewall*. Control Node kemudian menyusun laporan akhir yang diberikan kepada Administrator, memberikan gambaran lengkap tentang keberhasilan atau kegagalan konfigurasi *firewall* di seluruh target VMs.

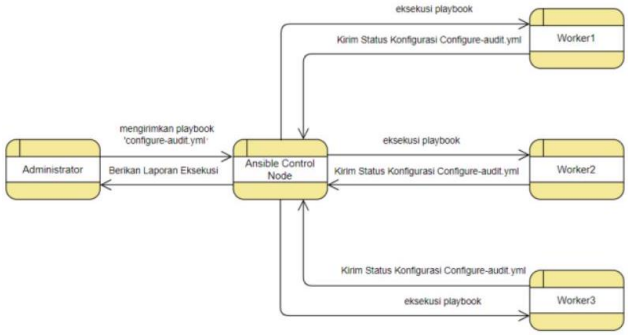
2. Ssh-hardening.yml

Pada gambar 9 di atas menjelaskan proses dari *Data Flow Diagram* (DFD) untuk otomasi *hardening* menggunakan playbook Ansible 'ssh-hardening.yml':

Administrator membuat dan mengunggah playbook ssh-hardening.yml ke Ansible Control Node, yang berisi perintah untuk mengamankan pengaturan SSH pada VMs. Control Node kemudian menjalankan playbook ini dan mendistribusikan perintah ke target nodes (worker1, worker2, worker3), yang mencakup instalasi OpenSSH Server, memastikan layanan SSH aktif saat booting, serta mengonfigurasi file sshd_config untuk menonaktifkan login root. Setiap target node menjalankan konfigurasi sesuai perintah, memastikan penguatan pengaturan SSH secara konsisten. Setelah itu, target nodes mengirimkan laporan status ke Control Node terkait keberhasilan instalasi dan penerapan konfigurasi. Control Node mengumpulkan laporan dari semua target nodes dan menyusunnya menjadi laporan akhir yang diberikan kepada Administrator, memberikan gambaran lengkap tentang hasil konfigurasi dan keamanan akses SSH di seluruh VMs.

3. Configure-auditd.yml

Pada gambar 10 di atas menjelaskan proses dari *Data Flow Diagram* (DFD) untuk otomasi *hardening* menggunakan playbook Ansible 'configure-auditd.yml':



Gambar 10. Ssh-hardening.yml

Administrator membuat dan mengunggah playbook configure-auditd.yml ke Ansible Control Node, yang berisi perintah untuk menginstal dan mengonfigurasi sistem auditd pada target VMs. Control Node kemudian menjalankan playbook tersebut dan mendistribusikan perintah ke target nodes (worker1, worker2, worker3), mencakup instalasi paket auditd, pengaturan kebijakan, dan konfigurasi aturan auditd untuk memantau aktivitas tertentu di sistem. Setiap target node menjalankan konfigurasi auditd sesuai instruksi, termasuk instalasi dan penyesuaian file konfigurasi serta penerapan aturan audit. Setelah selesai, target nodes mengirimkan laporan status mengenai keberhasilan instalasi dan penerapan kebijakan kembali ke Control Node. Control Node mengumpulkan semua laporan dan menyusunnya menjadi laporan akhir yang disampaikan kepada Administrator, memberikan gambaran lengkap tentang status konfigurasi auditd di seluruh target VMs dan memastikan sistem auditd terkonfigurasi dengan baik sesuai standar keamanan.

Kesimpulan

Penelitian ini mengungkapkan perbedaan antara penggunaan Ansible dan metode manual dalam penerapan *hardening* sistem. Data yang diperoleh menunjukkan bagaimana otomasi dengan Ansible mempercepat proses dan mengurangi risiko kesalahan dibandingkan dengan metode manual. Dengan memanfaatkan Ansible, setiap tahap *hardening* sistem dilakukan dengan konsistensi dan efisiensi yang tinggi. Selanjutnya, hasil perbandingan ini akan dirangkum untuk menilai secara lebih mendalam keberhasilan otomasi dalam konteks keamanan sistem.

1. Pengelolaan Keamanan Server Linux

Penelitian menunjukkan bahwa Ansible efektif dalam mengelola keamanan server Linux. Ansible mempermudah proses instalasi dan penghapusan perangkat lunak yang mendukung penguatan keamanan, seperti *firewall* dan konfigurasi SSH. Penerapan Ansible memungkinkan proses ini dilakukan dengan lebih cepat dan mengurangi risiko kesalahan manusia, memastikan pengelolaan yang lebih konsisten dan seragam dibandingkan dengan metode manual.

2. Kemudahan Pengelolaan dengan Otomatisasi

Otomasi menggunakan Ansible memperlihatkan kemudahan dalam pengelolaan server Linux. Dengan Ansible, proses konfigurasi pada banyak server dapat dilakukan secara bersamaan dengan satu perintah, menyederhanakan administrasi, dan memastikan standar keamanan yang konsisten. Proses penghapusan perangkat lunak yang rentan, seperti Telnet, juga dapat dilakukan dengan lebih mudah dan aman menggunakan Ansible, dibandingkan dengan

metode manual yang memerlukan lebih banyak waktu dan potensi risiko kesalahan.

3. Pengukuran Tingkat Kemudahan Otomatisasi

Hasil penelitian menunjukkan bahwa otomasi dengan Ansible memberikan beberapa keunggulan signifikan dibandingkan dengan metode manual dalam hal efisiensi waktu dan kemudahan implementasi. Meskipun waktu keseluruhan untuk proses otomatisasi, termasuk penyetingan awal, adalah 16 menit dan 18,99 detik, waktu yang dihabiskan untuk setiap tugas *hardening* individu jauh lebih singkat dibandingkan metode manual. Sebagai contoh, untuk konfigurasi SSH, Ansible menyelesaikan proses *hardening* dalam waktu total 13 detik dan 3 milidetik, sedangkan metode manual memerlukan waktu 1 menit, 36 detik, dan 17 milidetik. Pada konfigurasi *firewall*, Ansible menyelesaikan *hardening* dalam waktu total 10 detik dan 92 milidetik, sementara metode manual memakan waktu 1 menit, 38 detik, dan 45 milidetik.

Selain itu, Ansible memungkinkan pengelolaan audit dan penghapusan perangkat lunak rentan dengan lebih efisien, mengurangi waktu eksekusi dan meningkatkan keandalan hasil. Total waktu untuk konfigurasi audit menggunakan Ansible hanya 12 detik dan 3 milidetik, sedangkan metode manual membutuhkan waktu 1 menit, 44 detik, dan 79 milidetik. Keuntungan utama dari penggunaan Ansible mencakup pengurangan kemungkinan kesalahan manusia, peningkatan konsistensi, dan keandalan dalam penerapan kebijakan keamanan. Hal ini terutama penting ketika menghadapi lingkungan yang besar dan kompleks, di mana proses manual bisa memakan waktu lebih lama dan rentan terhadap kesalahan.

Daftar Pustaka

1. Fischer EA. Cybersecurity Issues and Challenges: In Brief; 2016.
2. Goldberg K. What Is Automation? IEEE Transactions on Automation Science and Engineering. 2012 Jan;9(1):1-2.
3. Moser R, Hochstein L. Ansible: Up and Running: Automating Configuration Management and Deployment the Easy Way; 2017.
4. Mourad A, Laverdiere MA, Debbabi M. Towards an Aspect Oriented Approach for the Security Hardening of Code. In: 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07). IEEE; 2007. p. 595-600.
5. Ylonen T. The Secure Shell (SSH) Transport Layer Protocol; 2006.
6. Chopra A. Security Issues of Firewall. International Journal of P2P Network Trends and Technology. 2016 Jan;22(1):4-9.