HARDENING SERVER MENGGUNAKAN METODE *PORT KNOCKING* PADA SISTEM PROGRAM STUDI TEKNIK INFORMATIKA UNIVERSITAS MUHAMMADIYAH PONOROGO

[1]Muhammad Reza, [2]Adi Fajaryanto Cobantoro, [3]Ismail Abdurrozzaq Zulkarnain*
[1,2,3]Program Studi Teknik Informatika, Fakultas Teknik Universitas Muhammadiyah Ponorogo
Jl. Budi Utomo No.10, Ronowijayan, Kabupaten Ponorogo, Jawa Timur 63471
[1]rezam9987@gmail.com,[2]adifajaryanto@umpo.ac.id,[3]ismail@umpo.ac.id
**) Penulis Korespondensi*

Abstrak

Pada era digitalisasi, keamanan data menjadi sangat rentan terhadap ancaman kebocoran, khususnya pada server yang menyimpan informasi sensitif. Penelitian ini bertujuan meningkatkan keamanan server Program Studi Teknik Informatika Universitas Muhammadiyah Ponorogo melalui implementasi hardening server. Langkah-langkah yang diterapkan meliputi konfigurasi port knocking untuk otentikasi akses, pengaturan firewall iptables, aktivasi portsentry, pemanfaatan Snort sebagai Intrusion Detection System (IDS), dan pemblokiran permintaan ICMP guna menangkal serangan berbasis ping. Pengujian keamanan menggunakan alat audit Lynis menunjukkan peningkatan signifikan, dengan skor keamanan awal 65, yang menunjukkan kerentanan tinggi, meningkat menjadi 96 setelah implementasi hardening. Penelitian ini menghadirkan pendekatan baru dengan mengintegrasikan berbagai mekanisme keamanan secara simultan, termasuk kombinasi port knocking dengan IDS Snort. Pendekatan ini memberikan perlindungan lebih baik terhadap risiko akses tidak sah, yang jarang diterapkan secara bersamaan dalam penelitian serupa. Langkah-langkah utama mencakup pembaruan sistem berkala, perlindungan port SSH (port 22) melalui pengaturan firewall, serta uji urutan otentikasi port knocking yang terintegrasi dengan IDS. Evaluasi dilakukan secara berulang menggunakan Lynis untuk mengukur efektivitas setiap langkah. Hasil penelitian membuktikan bahwa metode ini mampu meningkatkan ketahanan sistem secara substansial, menjaga kerahasiaan, integritas, dan ketersediaan data. Dengan demikian, server Program Studi Teknik Informatika menjadi lebih kuat dalam menghadapi ancaman siber.

***Kata Kunci**:IDS Snort, Metode Port knocking, Pengerasan server, Portsentry*

Abstract

In the digitalization era, data security has become highly vulnerable to leakage threats, particularly on servers storing sensitive information. This study aims to enhance the security of the server in the Informatics Engineering Program at Universitas Muhammadiyah Ponorogo through the implementation of server hardening. The applied measures include configuring port knocking for access authentication, setting up iptables firewall, activating portsentry, utilizing Snort as an Intrusion Detection System (IDS), and blocking ICMP requests to mitigate ping-based attacks. Security testing using the Lynis audit tool demonstrated significant improvement, with the initial security score of 65, indicating high vulnerabilities, increasing to 96 after implementing hardening measures. This study introduces a novel approach by integrating various security mechanisms simultaneously, including the combination of port knocking with Snort IDS. This approach provides better protection against unauthorized access risks, which are rarely implemented together in similar studies. Key steps include regular system updates, securing the SSH port (port 22) through firewall configuration, and testing port knocking authentication sequences integrated with IDS. Repeated evaluations using Lynis measured the effectiveness of

each step. The results prove that this method substantially enhances system resilience, maintaining data confidentiality, integrity, and availability. Thus, the Informatics Engineering Program server becomes more robust against cyber threats.

***Keywords**: Hardening Server, IDS Snort, Port knocking Method, Portsentry*

PENDAHULUAN

Pada era digitalisasi yang pesat teknologi informasi telah menjadi komponen integral dalam berbagai sektor, termasuk bisnis, pemerintahan, dan pendidikan[1]. Data dan informasi menjadi aset yang sangat berharga dan harus dijaga keamanannya[2]. Seiring dengan meningkatnya ketergantungan pada teknologi, risiko kebocoran dan penyalahgunaan data juga meningkat secara signifikan[3]. Meningkatnya insiden kebocoran data di Indonesia menjadi ancaman serius yang sering kali disebabkan oleh kesalahan konfigurasi perangkat lunak atau kelemahan pada sistem keamanan *server*. Menurut Badan Siber dan Sandi Negara (BSSN), sepanjang tahun 2023 terdapat 207 dugaan insiden kebocoran data di Indonesia, dengan sektor administrasi pemerintahan menjadi yang paling terdampak, mencapai 55% dari total insiden[4]. Hal ini disebabkan oleh tingginya jumlah sistem elektronik yang digunakan dalam kementerian dan lembaga pemerintahan, yang mencapai ratusan hingga ribuan sistem. Temuan ini menyoroti pentingnya penerapan langkah-langkah keamanan yang lebih ketat untuk melindungi data sensitif dalam administrasi pemerintahan guna mencegah insiden kebocoran data di masa depan[4].

Insiden semacam ini tidak hanya merugikan perusahaan yang mengalami kebocoran, tetapi juga berdampak negatif pada pihak-pihak terkait[5]. Kebocoran data dapat terjadi karena berbagai faktor, salah satunya adalah ketidakamanan *server* yang digunakan untuk menyimpan dan mengelola data sensitif[6]. Fakta ini menggarisbawahi perlunya langkah preventif yang lebih kuat dalam menjaga integritas, kerahasiaan, dan ketersediaan data di era digital.

Salah satu penyebab utama kebocoran data adalah ketidakamanan *server* yang digunakan untuk menyimpan dan mengelola data sensitif[7]. Ketidakamanan ini dapat disebabkan oleh berbagai faktor, seperti kurangnya pembaruan sistem secara rutin, konfigurasi keamanan yang tidak memadai, kelemahan dalam pengelolaan akses pengguna, serta tidak adanya penerapan teknologi keamanan seperti *firewall*, enkripsi, dan sistem deteksi intrusi[1]. *Server* yang tidak dilindungi dengan baik menjadi sasaran empuk bagi pihak yang tidak bertanggung jawab untuk melakukan akses tidak sah, pemalsuan identitas, atau pencurian data[8]. Ketidakamanan ini dapat berdampak serius pada operasional sehari-hari, terutama bagi institusi pendidikan yang sangat bergantung pada data untuk mendukung aktivitas akademik maupun administratif[9].

Penerapan proses pengamanan *server* secara menyeluruh melalui teknik hardening menjadi langkah penting dalam meningkatkan keamanan *server*. Hardening *server* merupakan serangkaian tindakan yang dirancang untuk mengurangi kerentanan terhadap serangan dan meningkatkan keamanan secara keseluruhan[10]. Tindakan yang termasuk dalam *hardening server* mencakup konfigurasi yang lebih ketat pada sistem, pengelolaan akses yang lebih baik untuk memastikan hanya pengguna yang berwenang yang dapat mengakses sistem, serta pemantauan aktivitas secara intensif untuk mendeteksi potensi ancaman secara dini[11]. Selain itu, penerapan berbagai tindakan keamanan lainnya, seperti *firewall* yang diperbarui, penggunaan *Intrusion Detection System* (IDS), serta pembatasan akses berbasis *IP* atau *port*, turut berperan penting dalam meningkatkan ketahanan *server* terhadap serangan[12][13]. Melalui langkah-langkah ini, *hardening server* bertujuan untuk meminimalkan celah yang dapat dimanfaatkan oleh penyerang, sekaligus mengoptimalkan kinerja dan keandalan sistem *server* secara menyeluruh[14].

Penelitian ini bertujuan untuk mengimplementasikan *hardening server* pada sistem Program Studi Teknik Informatika Universitas Muhammadiyah Ponorogo dengan pendekatan holistik yang mengintegrasikan beberapa metode sekaligus, seperti konfigurasi *port knocking*, pengaturan *firewall* menggunakan *iptables*, penerapan *Snort* sebagai *Intrusion Detection System* (IDS), dan pemblokiran ICMP untuk mencegah serangan berbasis *ping*. Berbeda dengan penelitian sebelumnya yang dilakukan oleh Fakhry, Reza Rivaldo (2023) yang hanya berfokus pada penerapan keamanan *server* melalui teknik *hardening* untuk meminimalkan kerawanan terhadap *attacker* pada *Ubuntu Server*, penelitian tersebut menghasilkan *output* berupa file images atau ISO yang dapat digunakan oleh perusahaan sebagai dasar keamanan *server*[15]. Selain itu, pengujian yang dilakukan menunjukkan kemampuan *server* dalam mendeteksi alamat penyerang, membatasi akses *client* asing, membatasi koneksi, menolak *ping*, dan memantau aktivitas penyerang[16]. Penelitian ini berbeda karena mengambil pendekatan yang lebih luas dan terintegrasi, seperti seperti *firewall* yang diperbarui, penggunaan *Intrusion Detection System* (IDS), serta pembatasan akses berbasis *IP* atau *port*, turut berperan penting dalam meningkatkan ketahanan *server* terhadap serangan. Sehingga diharapkan implementasi metode yang digunakan tidak hanya meningkatkan keamanan *server* di institusi pendidikan, tetapi juga memberikan panduan praktis yang lebih komprehensif bagi institusi lain yang menghadapi tantangan serupa.

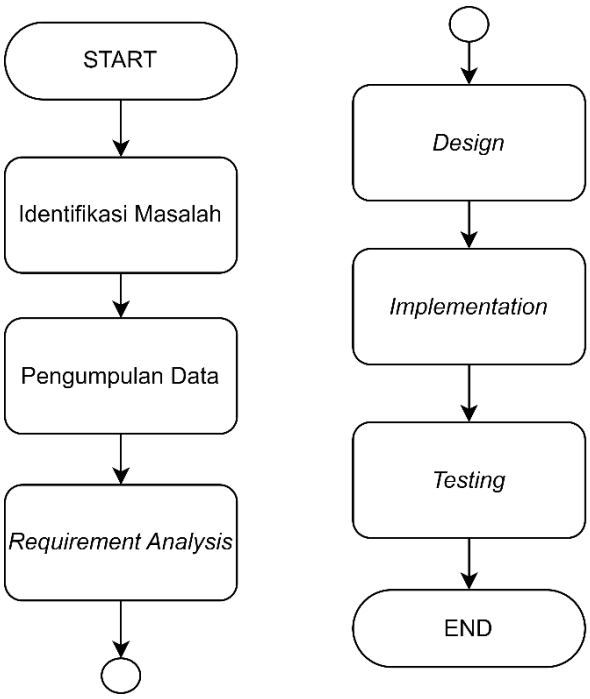
METODE PENELITIAN

Metode penelitian ini meliputi beberapa tahapan, dimulai dari identifikasi masalah untuk memahami isu utama, dilanjutkan

dengan pengumpulan data yang relevan sebagai dasar analisis. Selanjutnya, dilakukan *requirement analysis* untuk menentukan kebutuhan sistem, diikuti dengan tahap desain solusi. Setelah itu, dilakukan implementasi solusi ke dalam bentuk sistem atau aplikasi, yang kemudian diuji pada tahap pengujian untuk memastikan fungsionalitasnya sesuai dengan kebutuhan. Tahapan ini memastikan penelitian berjalan secara terstruktur dan sistematis.

1. Identifikasi Masalah

Proses identifikasi masalah keamanan *server* di Sistem Program Studi Teknik Informatika Universitas Muhammadiyah Ponorogo mengungkapkan bahwa *server* rentan terhadap serangan eksternal, khususnya melalui *port* SSH yang menjadi target utama akses ilegal. *Server* ini menyimpan data vital terkait navigasi udara, seperti lalu lintas udara, telekomunikasi penerbangan, dan informasi aeronautika, yang menjadikannya sasaran menarik bagi *attacker*. Risiko ini menuntut solusi yang efektif untuk mencegah akses tidak sah dan melindungi integritas serta kerahasiaan data. Salah satu solusi yang dapat diterapkan adalah metode *Port knocking* untuk meningkatkan keamanan akses. Namun, *server* masih kekurangan langkah-langkah keamanan memadai, seperti konfigurasi *firewall* yang ketat dan *hardening server*. Penting untuk segera melakukan pembatasan akses, mengaktifkan pengaturan keamanan bawaan, dan menyesuaikan konfigurasi *firewall* guna melindungi data dari serangan eksternal dan meningkatkan keamanan keseluruhan.



Gambar 1. Tahapan Penelitian

2. Pengumpulan data

Pada tahap pengumpulan data dilakukan dengan Studi literatur melibatkan pencarian dan analisis terhadap berbagai sumber informasi yang relevan, seperti jurnal ilmiah, buku teks, dan sumber-sumber online yang terkait dengan keamanan *server* dan metode *Port knocking*. Proses ini bertujuan untuk memahami konsep-konsep dasar, prinsip kerja, manfaat, dan tantangan terkait dengan penggunaan metode *Port knocking* dalam meningkatkan keamanan *server*. Selain itu, studi literatur juga membantu dalam mengevaluasi praktik terbaik dan solusi yang telah diterapkan dalam lingkup keamanan sistem informasi, yang dapat diadopsi dan disesuaikan dalam konteks penelitian.

3. *Requriment Analysis*

Pada tahap *Requirement Analysis*, langkah awal yang dilakukan penelitian yaitu melakukan analisis kebutuhan untuk menetapkan prioritas dalam pengamanan *server*. Dengan menyadari pentingnya data terkait navigasi udara seperti lalu lintas udara, telekomunikasi penerbangan, dan informasi meteorologi penerbangan, peneliti memfokuskan upaya pada pengamanan sisi SSH *server*. Selain itu, batasan akses ke *server* juga ditegakkan dengan memperbolehkan hanya *host* yang terdaftar untuk mengaksesnya, sementara *host* yang tidak dikenali akan ditolak secara otomatis. Berikut beberapa perangkat lunak dan keras yang diperlukan :

a) Laptop (Lenovo *Ideapad* 320, *Processor Intel Core* i3 6006u, *Nvidia GeForce* 920MX, RAM 12GB, 128GB SSD + 1TB *Harddisk*, *Windows* 11 Pro)

b) Oracle VM *VirtualBox* (*Linux Client* dan *Server*)

c) *Kali Linux* 2024.3

d) *Ubuntu server* 20.04 LTS

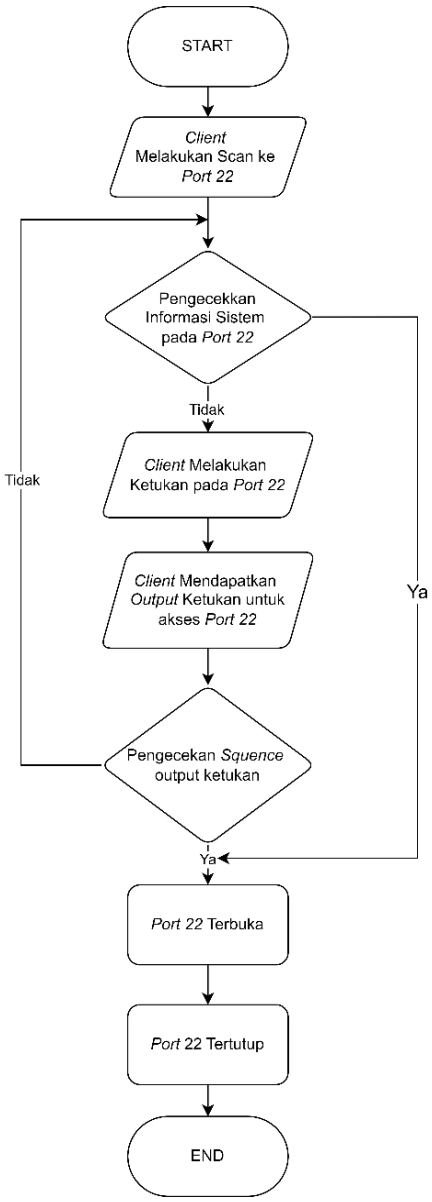
e) *Ip tables, Port knocking, Portsentry, IDS Snort, ICMP*

Analisis kebutuhan keamanan berikutnya melibatkan evaluasi terhadap jenis data yang disimpan di *server*, yang termasuk informasi sensitif seperti data mahasiswa dan administratif. Selain itu, potensi ancaman dari serangan luar seperti serangan *brute force* dan upaya peretasan melalui *port* SSH juga dipertimbangkan dengan seksama. Berdasarkan pemahaman yang mendalam tentang kebutuhan keamanan dan potensi ancaman, langkah-langkah *hardening server* yang efektif dirancang dan diimplementasikan. Ini mencakup pembaruan perangkat lunak secara teratur, dan penerapan kebijakan keamanan yang ketat. Selain itu, penggunaan metode *Port knocking* dan integrasi *Snort* sebagai IDS menjadi bagian penting dari strategi keamanan yang bertujuan untuk melindungi *server* dengan optimal dari berbagai ancaman yang mungkin terjadi.

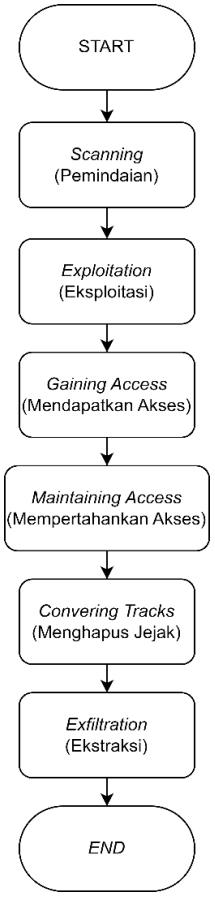
4. *Design*

Setelah tahap analisis kebutuhan selesai, langkah berikutnya adalah merancang sistem keamanan *server* secara terperinci untuk

menggambarkan implementasi langkah-langkah *hardening server*. Pada tahap ini, desain visual seperti *flowchart Port knocking* dibuat untuk menunjukkan alur akses yang aman ke *server*, di mana hanya pengguna yang mengikuti urutan tertentu dapat membuka *port* SSH. Selain itu, *flowchart* "Alur *Attacker Server*" dirancang untuk mengilustrasikan bagaimana ancaman dari penyerang akan diidentifikasi dan diblokir melalui mekanisme keamanan, sehingga dapat mencegah akses tidak sah. Skema desain sistem keamanan juga diperlukan untuk menunjukkan bagaimana komponen-komponen keamanan seperti *firewall*, IDS (*Intrusion Detection System*), dan aturan-aturan *port knocking* akan bekerja bersama dalam melindungi *server*. Desain yang matang ini akan menjadi acuan utama dalam implementasi langkah-langkah hardening guna memperkuat keamanan *server*.



Gambar 2. *Flowchart Port knocking*



Gambar 3. *Flowchart* Alur *Attacker Server*

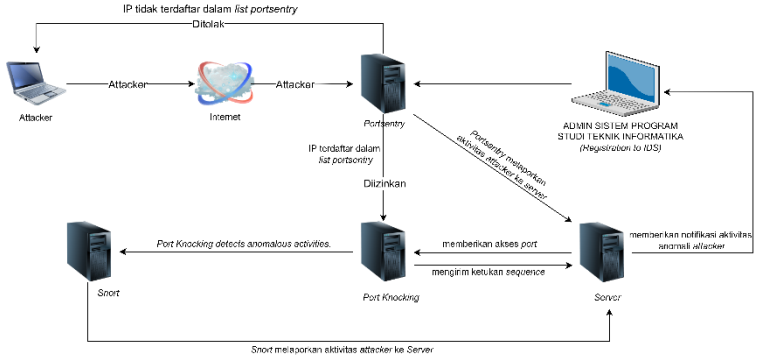
Pada gambar 2, ditampilkan *flowchart* yang menggambarkan algoritma *port knocking*, yang dimulai dengan langkah awal yaitu Start. Kemudian, client melakukan scan ke *port* 22 untuk mencoba terhubung dengan *server*. Jika *port* 22 terbuka atau client memiliki akses, maka client dapat langsung terhubung ke *server*. Namun, jika *port* 22 tertutup, *server* akan melanjutkan ke langkah berikutnya, yaitu pengecekan informasi sistem pada *port* 22. Selanjutnya, client mengirimkan ketukan pada *port* 22 dengan urutan yang telah ditentukan sebelumnya. Setelah itu, *server* memverifikasi urutan ketukan dan memastikan apakah ketukan yang diterima sesuai dengan aturan yang sudah ditetapkan. Jika urutan ketukan sesuai, *server* akan membuka *port* 22 untuk koneksi dari client. Setelah koneksi selesai, *server* akan menutup *port* 22 untuk mencegah akses yang tidak diinginkan, dan akhirnya proses berakhir pada langkah END. *Flowchart* ini menggambarkan secara rinci alur kerja *port knocking* untuk membuka akses yang aman ke *server*.

Gambar 3, ditampilkan *flowchart* yang menggambarkan alur serangan seorang *attacker* terhadap *server* sistem Program Studi Teknik Informatika Universitas Muhammadiyah Ponorogo. Proses dimulai dengan tahap pemindaian, di mana *attacker* menggunakan alat pemindaian seperti Nmap untuk mencari kerentanannya pada *server* atau

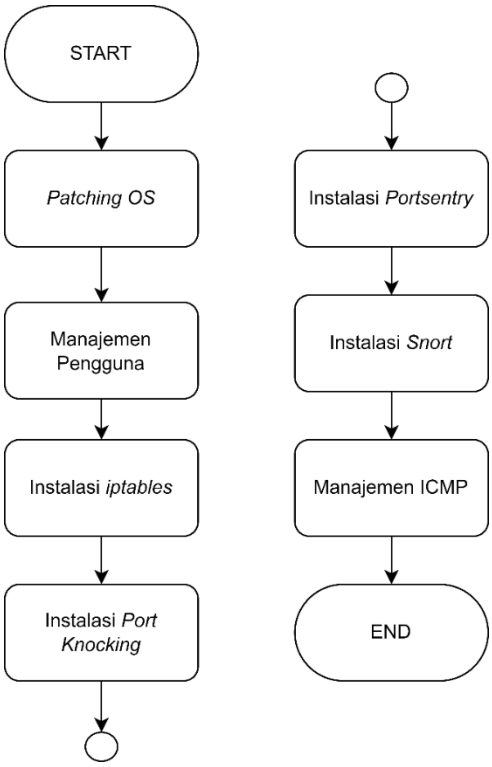
infrastruktur yang dapat dieksploitasi. Setelah menemukan kerentanan, *attacker* melanjutkan dengan eksploitasi, mencoba untuk mengeksploitasi celah tersebut, misalnya dengan menggunakan kata sandi default atau serangan *brute force*. Jika berhasil, *attacker* mendapatkan akses ke dalam sistem dan seluruh data yang ada di dalamnya. Tahap berikutnya, *attacker* mungkin berusaha untuk mempertahankan akses dengan memasang *backdoor* atau menciptakan akun tambahan yang tidak terdeteksi oleh administrator. Setelah itu, *attacker* akan menghapus jejak aktivitasnya dengan menghapus *log* akses atau *log* keamanan agar tidak terdeteksi. Terakhir, tahap ekstraksi, *attacker* mengekstraksi data atau informasi sensitif dari *server*, menggunakan teknik seperti menyalin file atau mentransfer data keluar dari *server*. *Flowchart* ini menggambarkan langkah-langkah yang diambil oleh *attacker* untuk mengeksploitasi dan mengakses sistem *server* secara ilegal.

Gambar 4 menggambarkan skema desain sistem keamanan *server* dengan tiga lapisan pelindung, yaitu *IDS Portsentry, Port knocking*, dan *Snort*, yang bekerja secara berlapis untuk melindungi layanan file *server* dari serangan pada *port* SSH. *IDS Portsentry* bertugas mendeteksi aktivitas *port scanning* dan memblokir IP yang tidak terdaftar dalam konfigurasi file seperti pada *portsentry.conf* dan *portsentry.ignore.static*. Jika ada IP tidak terdaftar yang mencoba mengakses *Portsentry* akan otomatis menolak koneksi tersebut. Jika akses berhasil melalui *Portsentry*, *Port knocking* akan memastikan bahwa hanya IP yang mengikuti urutan (*sequence*) tertentu dapat membuka *port* SSH, sehingga lapisan ini bertindak sebagai kontrol akses tambahan yang hanya dapat diakses oleh admin sistem[13].

Sebagai lapisan terakhir *Snort* berfungsi sebagai *Intrusion Detection System* (IDS) yang mengawasi lalu lintas jaringan di *server*. *Snort* mencocokkan pola lalu lintas dengan aturan yang telah ditentukan untuk mendeteksi ancaman, seperti *port scanning* atau akses mencurigakan, serta mengambil tindakan respons sesuai aturan tersebut. Ketiga lapisan ini bekerja terintegrasi, dan aktivitas yang terdeteksi akan dikirim ke *server* utama yang mengirimkan notifikasi langsung, seperti pesan atau email, kepada admin sistem. Dengan adanya laporan aktivitas mencurigakan secara *real-time*, admin dapat merespons secara cepat untuk menjaga keamanan dan integritas *server*. Gambar 5 menjelaskan langkah-langkah konfigurasi teknik *hardening server* di Kali Linux yang mencakup tujuh tahap utama. Pertama, patching diterapkan untuk memperbarui kernel Linux dan perangkat lunak, menutup celah kerentanan agar penyerang tidak dapat mengeksploitasi sistem.



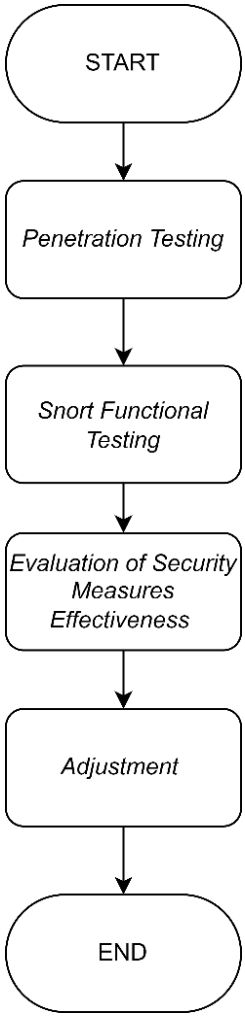
Gambar 4. Skema Desain Perancangan Sistem Keamanan *Server*



Gambar 5. Tahapan Implementasi

Kedua, manajemen pengguna dilakukan dengan menambah atau menghapus pengguna yang tidak perlu untuk mengurangi risiko akses tidak sah. Ketiga, *iptables* diinstal untuk menerapkan aturan keamanan, termasuk pembatasan akses ke *port* SSH menggunakan iptables-persistent. Keempat, *Port knocking* diterapkan pada *port* SSH menggunakan knockd untuk memastikan akses hanya diberikan pada pengguna yang mengikuti sequence tertentu. Kelima, *port*sentry diinstal untuk mendeteksi aktivitas mencurigakan seperti pemindaian *port*, dan keenam, Snort diinstal sebagai IDS untuk memantau lalu lintas jaringan dan mendeteksi ancaman seperti serangan hacker atau malware. Terakhir, permintaan ICMP dikelola melalui sysctl.conf untuk mencegah serangan DDoS.

Setiap langkah ini bertujuan untuk meningkatkan keamanan *server* dari berbagai potensi ancaman. Gambar 6 menggambarkan proses pengujian keamanan *server* untuk memastikan efektivitas langkah-langkah yang telah diterapkan. Pengujian dimulai dengan penetration testing guna mengidentifikasi kerentanan terhadap serangan seperti DoS, brute force, dan eksploitasi perangkat lunak. Setelah itu, Snort diuji secara fungsional untuk memastikan kemampuannya mendeteksi serangan jaringan seperti *port* scanning dan malware. Hasil pengujian ini dievaluasi untuk menilai efektivitas langkah-langkah keamanan yang diterapkan, termasuk respons *server* terhadap serangan dan deteksi serangan oleh Snort. Jika diperlukan, konfigurasi diperbarui atau kebijakan keamanan diperketat untuk mengurangi risiko serangan di masa depan[13].



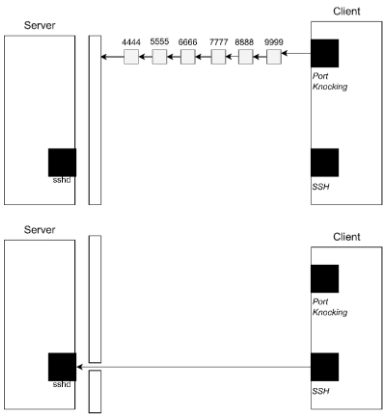
Gambar 6. Tahapan *Testing*

HASIL DAN PEMBAHASAN

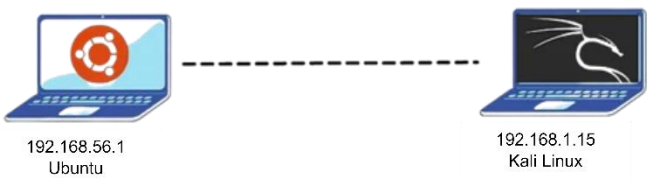
Implementasi *port knocking* melibatkan topologi antara klien dan *server*, di mana klien harus mengirimkan urutan ketukan pada port tertentu agar *server* membuka akses ke layanan yang dilindungi, seperti SSH. Dalam topologi *attacking server*, dilakukan simulasi serangan untuk menguji efektivitas *port knocking* dalam melindungi *port* dari akses tidak sah. Konfigurasi pada *server* mencakup pengaturan daemon *port knocking*, seperti urutan *port* yang harus diketuk, batas waktu validasi, dan integrasi dengan *firewall* (misalnya *iptables*) untuk membuka atau menutup akses sesuai pola ketukan.

Gambar 7 menunjukkan topologi *Port knocking* di mana *client* mengirimkan urutan ketukan melalui *port* 4444, 5555, 6666, 7777, 8888, dan 9999. Setelah urutan ketukan ini diterima dengan benar oleh *server*, *port* SSH akan terbuka, memungkinkan *client* untuk melakukan koneksi SSH ke *server* tersebut. Hal ini mencegah akses langsung ke layanan SSH, sehingga meningkatkan keamanan jaringan.

Gambar 8 menunjukkan topologi *attacking server*, di mana dua perangkat terhubung dalam jaringan yang sama. Di sebelah kiri, ada perangkat dengan sistem operasi *Ubuntu* dan alamat IP 192.168.56.1. Di sebelah kanan, ada perangkat dengan sistem operasi *Kali Linux* dan alamat IP 192.168.1.15. *Kali Linux*, yang dikenal sebagai *platform* untuk pengujian penetrasi, digunakan untuk melakukan serangan terhadap *server Ubuntu*. Koneksi ini menggambarkan skenario di mana *Kali Linux* bertindak sebagai *attacker*, sementara *Ubuntu* berperan sebagai target atau *server* yang diserang.



Gambar 7. Topologi *Port knocking*



Gambar 8. Topologi *Attacking Server*

Tabel 1. Konfigurasi *Apache*

Parameter	*Value*
DocumentRoot	/var/ww/html/
FileETag	None
Timeout	10
LimitRequestBody	102400
LimitRequestFieldsize	1024
LimitRequestline	512
LogLevel	notice core:info
Protocols	h2 http/1.1
*Server*Tokens	Prod
TraceEnable	off

Tabel 2. Konfigurasi PHP

Parameter	*Value*
allow_url_fopen	0
allow_url_include	0
display_errors	0
display_start_errors	0
expose_php	0
html_errors	0
log_errors	1
open_basedir	/var/www/html
post_max_size	100K
session.use_strict_mode	1
session.cookie_httponly	1
session.cookie_secure	1

Tabel 1 menunjukkan konfigurasi *Apache* yang digunakan pada *server*. Parameter *DocumentRoot* menunjukkan direktori utama di mana file web disimpan yaitu /var/www/html/. Pengaturan *FileETag* diatur ke *None*, yang berarti pengurangan *overhead* dalam menangani file *ETag*. Waktu batas koneksi Timeout diatur ke 10 detik. Batas ukuran tubuh permintaan diatur melalui *LimitRequestBody* sebesar 102400 *byte*, sementara *LimitRequestFieldsize* dan *LimitRequestline* masing-masing dibatasi hingga 1024 *byte* dan 512 karakter untuk menghindari serangan *buffer overflow*. *Level log* yang digunakan adalah *notice core:info*, yang memastikan informasi inti *server* dicatat. Protokol yang digunakan adalah h2 (HTTP/2) dan http/1.1, dengan *ServerTokens* diatur ke Prod untuk menyembunyikan informasi versi *server*. Terakhir, *TraceEnable* dinonaktifkan dengan nilai *off* untuk mencegah penyalahgunaan fitur *trace*.

Tabel 2 menampilkan konfigurasi PHP yang digunakan untuk meningkatkan keamanan aplikasi web. Pengaturan seperti allow_url_fopen, allow_url_include, display_errors, display_start_errors, dan expose_php semuanya disetel ke 0 untuk mencegah celah keamanan dari file yang diambil melalui URL dan menghindari pengungkapan informasi sistem kepada pengguna. html_errors juga disetel ke 0, sehingga pesan kesalahan tidak di-output dalam format HTML. Sebaliknya, *log_errors* diaktifkan dengan nilai 1, yang memungkinkan kesalahan dicatat ke file log

untuk referensi administrator. Direktori kerja untuk PHP dibatasi menggunakan open_basedir yang disetel ke /var/www/html, dan ukuran maksimum untuk post data ditetapkan sebesar 100K melalui post_max_size. Pengaturan tambahan untuk keamanan sesi termasuk session.use_strict_mode,session.cookie_httpo nly, dan session.cookie_secure yang semuanya disetel ke 1, untuk memastikan keamanan sesi yang lebih ketat dan melindungi *cookie* dari serangan *cross-site scripting* (XSS). Tabel 3 menunjukkan konfigurasi parameter autentikasi pada *OpenSSH*. \Konfigurasi ini terdiri dari beberapa metode autentikasi yang dapat diaktifkan atau dinonaktifkan. Parameter seperti*ChallengeResponseAuthentication,GSS APIAuthentication,HostbasedAuthentication, PasswordAuthentication,RhostsRSAAuthentic ation,* dan *RSAAuthentication* semuanya disetel ke no, yang berarti metode autentikasi tersebut dinonaktifkan. Namun, *PubkeyAuthentication* diatur ke yes, menunjukkan bahwa autentikasi dengan kunci publik diizinkan.

Konfigurasi ini menunjukkan pendekatan yang lebih aman dengan menonaktifkan metode autentikasi yang kurang kuat dan hanya mengaktifkan autentikasi berbasis kunci publik.

Tabel 3. Konfigurasi *Openssh Authentication*

Parameter	*Value*
ChallengeResponeAuthetication	no
GSSAPIAuthetication	no
HostbaseAuthetication	no
PasswordAuthetication	no
PubkeyAuthetication	yes
RhostsRSAAuthetication	no
RSAAuthetication	no

Tabel 4. Konfigurasi *OpenSSH*

Parameter	*Value*
Banner	/etc/issue.net
AllowAgentForwarding	no
AllowGroups	ssh
AllowTcpForwarding	no
ClientAliveInterval	300
Compression	no
DebianBanner	no
IgnoreRhosts	yes
LoginGraceTime	60
LogLevel	VERBOSE
MaxAuthTries	2
Massessions	2
PermitRootLogin	no
Port	2222
Protocol	2
TCPKeepAlive	no
UseDNS	no
UsePrivilegeSeparation	SANDBOX
X11Forwarding	no

Tabel 4 menunjukkan konfigurasi parameter pada *OpenSSH* yang meliputi berbagai aspek keamanan dan fungsionalitas. Parameter Banner disetel ke /etc/issue.net yang menunjukkan file banner yang akan ditampilkan pada saat login. *AllowAgentForwarding*,*AllowTcpForwarding* , dan *Compression* disetel ke no, menandakan fitur-fitur tersebut dinonaktifkan untuk meningkatkan keamanan. *ClientAliveInterval* disetel ke 300, yang berarti *server* akan mengirimkan sinyal ke klien setiap 300 detik. *LoginGraceTime* disetel ke 60, membatasi waktu login menjadi 60 detik. *LogLevel* disetel ke VERBOSE untuk mencatat detail log yang lebih rinci. Selain itu, *port* yang digunakan adalah 2222, protokol SSH versi 2, dan beberapa pengaturan seperti *PermitRootLogin*, *UsePrivilegeSeparation*,serta *X11Forwarding* juga disesuaikan untuk meningkatkan keamanan, di mana root login dinonaktifkan dan penggunaan hak istimewa dipisahkan.

Tabel 5. Hasil Pengujian *Lynis Audit Security*

Solusi	***Score***
Implementasi Solusi	96
Total konfigurasi	256
Default konfigurasi	65

Tabel 6. Hasil Evaluasi Pengujian *Hardening Server*

Kelas Uji	**Skenario Uji**	**Input**	***Response Server***	**Hasil**	**Keterangan**
Patching	Melakukan Pengecekan Pembaruan Sistem	lsb-release -a	Menampilkan versi sistem	Valid	Sistem telah berhasil terupdate dengan versi terbaru
Iptables	Melakukan inisalisasi untuk 3 pilar keamanan dengan tes scan ke *localhost*	nmap 192.168.56.1	Memfilter *port* 22	Valid	Sistem telah berhasil mengubah status *port* 22 menjadi *filtered*
Port knocking	Mencoba *knocking*	knock –v 192.168.56.1 4444 5555 6666 7777 8888 9999	Membuka *knock*	Valid	Sistem telah berhasil membuka *port knock*
Port knocking	Mencoba *knocking*	knock –v 192.168.56.1 4444 5555 6666 7777 8888 9999	Menutup *knock*	Valid	Sistem telah berhasil menutup *port knock*
IDS Snort	Melakukan scanning terhadap *server*	nmap 192.168.56.1	Memberikan *alert attack* & memblokir IP *attacker*	Valid	Sistem telah berhasil menjebak penyerang masuk ke dalam *server* tipuan
ICMP	Melakukan tes ping	ping 192.168.56.1	Memblokir permintaan icmp	Valid	Sistem telah berhasil menolak permintaan icmp

Tabel 5 menunjukkan hasil audit keamanan yang dilakukan menggunakan Lynis, sebuah alat untuk menilai keamanan sistem. Tabel ini mencantumkan skor dari berbagai aspek konfigurasi dan implementasi keamanan. Skor untuk Implementasi Solusi adalah 96, yang menunjukkan bahwa tindakan keamanan yang diimplementasikan sudah cukup baik. Total konfigurasi mencatat skor 256, yang mencerminkan keseluruhan konfigurasi sistem yang telah diatur dan diperiksa selama audit. Sementara itu, *Default* konfigurasi memiliki skor 65, menunjukkan bahwa konfigurasi default sistem yang belum dimodifikasi memiliki tingkat keamanan yang lebih rendah dibandingkan dengan konfigurasi yang telah disesuaikan atau dioptimalkan.

Tabel 6 menunjukkan hasil dari konfigurasi *sysctl.conf,* yang mencatat sebanyak 338 paket permintaan mengalami paket loss. Dalam percobaan ICMP, jika tidak dihentikan secara manual melalui ctrl+c, akan terjadi *loop* paket *loss* yang terus berulang tanpa henti. Penggunaan *iptables* di sini berfungsi sebagai penerapan aturan yang akan diaplikasikan pada aplikasi-aplikasi terkait. Peneliti memilih menggunakan *iptables* persistent agar jika *server* mengalami hang atau perlu di-*reboot*, aturan-aturan yang telah ditetapkan tidak hilang. Setelah penerapan aturan tersebut, iptables menyebabkan *port* 22 berstatus *filtered*, yang berarti tidak semua koneksi akan diterima.

Konfigurasi *iptables* ini dilakukan untuk mengatur *port knocking* yang beroperasi pada *port* 22. Setelah melakukan konfigurasi, *knocking* menghasilkan beberapa temuan:

a) Proses *knocking open* dari client 192.168.56.1 yang diizinkan oleh *server* IDS berjalan dengan baik.

b) Proses *knocking close* dari *client* 192.168.56.1 yang juga diizinkan oleh *server* IDS berfungsi dengan baik.

c) Proses *knocking open* dari *client* 192.168.56.1 yang tidak diizinkan oleh *server* IDS mengalami penolakan koneksi. Setiap percobaan koneksi dari *IP address* 192.168.56.1 dicatat oleh *IDS portsentry*.

Port knocking, client 192.168.56.1 mengalami penolakan akses SSH yang disebabkan oleh *portsentry*. *Portsentry* mencatat kejadian tersebut dan langsung memblokir IP menggunakan *TCPwrapper* dan *iptables*. Daftar alamat IP yang diizinkan untuk mengakses telah dikonfigurasi dalam file *portsentry.ignore.static*.

KESIMPULAN DAN SARAN

Berdasarkan hasil dan pembahasan, penelitian ini berhasil mengimplementasikan *hardening server* pada sistem Program Studi Teknik Informatika Universitas Muhammadiyah Ponorogo dengan pendekatan holistik yang mengintegrasikan beberapa metode sekaligus. Konfigurasi keamanan dilakukan dengan menggunakan *port knocking*, pengaturan *firewall* melalui *iptables*, penerapan *Snort* sebagai *Intrusion*

Detection System (IDS), serta pemblokiran ICMP untuk mencegah serangan berbasis *ping*. Setiap langkah konfigurasi dilakukan secara manual dan selektif sesuai dengan kebutuhan spesifik *server*, kemudian diaudit menggunakan *Lynis audit system* dengan mengacu pada pedoman *CIS Benchmark*. Hasilnya, penerapan *hardening* ini berhasil mencapai skor audit 96/100, yang menunjukkan tingkat keamanan *server* yang sangat ketat. Sebanyak 256 konfigurasi keamanan telah diterapkan, didukung dengan alat *monitoring* dan deteksi seperti *Grafana, Kibana, OSSEC*, dan *Nagios* yang memungkinkan pemantauan dan deteksi ancaman secara *real-time*.

Pengujian *port knocking*, konfigurasi berhasil membuka dan menutup *port* dengan urutan ketukan autentik, memastikan hanya pengguna yang mengetahui urutan yang dapat mengakses *port*, menambah lapisan perlindungan pada *server*. Uji *Snort* IDS menunjukkan kemampuan sistem untuk mendeteksi dan memberi peringatan terhadap upaya pemindaian menggunakan Nmap, serta memblokir IP penyerang. Pengujian *Portsentry* berhasil memantau dan merespons upaya akses tidak sah, sementara pengujian ICMP menunjukkan bahwa *server* dapat menolak permintaan ping, memperkuat pertahanan terhadap serangan *SYN Flood Attack* dan *DNS Spoofing*. Terakhir, uji konfigurasi Iptables menunjukkan sistem berhasil memfilter *port* 22 dengan status "*filtered*" saat dilakukan pemindaian menggunakan Nmap, yang meningkatkan keamanan akses jaringan dan mencegah akses tidak sah. Dengan demikian, implementasi *hardening server* pada sistem ini berhasil meningkatkan keamanan *server* Program Studi Teknik Informatika Universitas Muhammadiyah Ponorogo secara signifikan, dengan pendekatan yang komprehensif dan terintegrasi.

DAFTAR PUSTAKA

[1] D. Rahman, H. Amnur, and I. Rahmayuni, "Monitoring *Server* dengan Prometheus dan Grafana serta Notifikasi Telegram," *JITSI J. Ilm. Teknol. Sist. Inf.*, vol. 1, no. 4, pp. 133–138, 2020, doi: 10.30630/jitsi.1.4.19.

[2] M. A. Prabowo, U. Darusalam, and S. Ningsih, "Perancangan Keamanan *Server* Linux Dengan Metode Hardening Pada Layer 1 dan Layer 7," *J. Media Inform. Budidarma*, vol. 4, no. 3, pp. 591–603, 2020, doi: 10.30865/mib.v4i3.2157.

[3] R. Ernawati, I. Ruslianto, and S. Bahri, "Implementasi Metode *Port knocking* Pada Sistem Keamanan *Server* Ubuntu Virtual Berbasis Web Monitoring," *J. Komput. dan Apl.*, vol. 10, no. 01, pp. 158–169, 2022, [Online]. Available: https://jurnal.untan.ac.id/index.php/jcskommipa/article/download/54226/75676593086

[4] L. Septiani, "BSSN Catat Ada 207

Pencurian Data, Pemerintah Paling Banyak," *Dkatadata.com*, 2023. https://katadata.co.id/digital/teknologi/6537ae3a29314/bssn-catat-ada-207-pencurian-data-pemerintah-paling-banyak

[5] D. Febriawan and H. Marisa, "Understanding Indonesia's Cyber Security Policies: Op*port*unities and Challenges In The Digitalization Transformation Era," *JOELS J. Elect. Leadersh.*, 2024, [Online]. Available: https://journal.unilak.ac.id/index.php/joels/article/download/15908/5974

[6] Apjii, "Kasus Data Pribadi yang Selalu Bocor," *APJII BULETIN*, no. 94, pp. 1–10, 2021. [Online]. Available: www.TEKNO.KOMPAS.COM

[7] L. Kestina, Yuhandri, and G. W. Nurcahyo, "Penanganan Celah Keamanan Website dengan Ethical Hacking dan Issaf Menggunakan Acunetix Vulnerability (Studi Kasus di Bkpsdmd Kabupaten Kerinci)," *Innov. J. Soc. Sci. Res.*, vol. 3, no. 4, pp. 9192–9203, 2023.

[8] R. J. Tampubolon, Poningsih, Solikhun, I. Gunawan, and Z. M. Nasution, "Implementasi Filtering Firewall Dan Hardening Web *Server* Untuk Mencegah Serangan Http Dos Pada Dinas Lingkungan Hidup Pematangsiantar," *Device*, vol. 11, no. 2, pp. 21–29, 2021, doi: 10.32699/device.v11i2.2091.

[9] M. A. Ikhsan and B. Handaga, "Penerapan Keamanan *Server* Menggunakan Security Information Event And Management Pada Sistem Operasi Ubuntu *Server*," vol. 20, pp. 1–23, 2021.

[10] R. Z. Naufal, U. Yunan, and M. Fathinuddin, "Hardening Cloudfri Dengan Metode Security Hardening Pada Aplikasi Berbasis Website Tap2Go.Cloudfri.Id," *e-Proceeding Eng.*, vol. 8, no. 5, pp. 9190–9201, 2021, [Online].Available:https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/view/15852

[11] Nazwita and S. Ramadhani, "Analisis Sistem Keamanan Web *Server* Dan Database *Server* Menggunakan Suricata," *Semin. Nas. Teknol. Informasi, Komun. dan Ind. 9*, pp. 2579–5406, 2019.

[12] M. A. J. Hidayat and H. M. Putra, "Sistem Keamanan *Server* Linux CentOS Dengan Metode *Port* Knock dan RST Cookies," *J. Inform. dan Teknol.*, vol. 6, no. 2, pp. 411–420, 2023.

[13] M. Iqbal, A.- Arini, and H. B. Suseno, "Analisa Dan Simulasi Keamanan Jaringan Ubuntu *Server* Dengan *Port knocking*, Honeypot, Iptables, Icmp," *Cyber Secur. dan Forensik Digit.*, vol. 3, no. 1, pp. 27–32, 2020, doi: 10.14421/csecurity.2020.3.1.1933.

[14] A. Handayani *et al.*, "Implementasi

Sistem Keamanan Komputer Host Menggunakan Sistem Operasi Fedora Linux," *Innov. J. Soc. Sci. Res.*, vol. 3, no. 2, pp. 721–736, 2023.

[15] R. R. Fakhry, "Penerapan Keamanan *Server* dengan Teknik Hardening pada Sistem Operasi Ubuntu *Server*," 2020. [Online].Available:http://eprints.ums.ac.id/id/eprint/90707

[16] A. H. D. Putri, A. Muhdiyyah, and R. R. Anarki, "Analisis Metode-Metode Peningkatan Keamanan Web *Server*," *J. Ilm. Sains dan Teknol.*, vol. 2, no. 8, pp. 1–5, 2024.