Implementasi Dan Analisis Openscap *Vulnerability Scanning* Dengan Sistem Manual Dan Otomatis Menggunakan Ansible

1st Muhammad Rizki Rafsyandjani
Fakultas Rekayasa Industri
Universitas Telkom
Bandung, Indonesia
rizkirafsyandjani@student.telkomuniversity.ac.id

2nd Adityas Widjajarto
Fakultas Rekayasa Industri
Universitas Telkom
Bandung, Indonesia
adtwjrt@telkomuniversity.ac.id

3rd Umar Yunan Kurnia Septo Hediyanto
Fakultas Rekayasa Industri
Universitas Telkom
Bandung, Indonesia
umaryunan@telkomuniversity.ac.id

Abstrak*— Keamanan sistem dan jaringan merupakan hal penting untuk menjaga integritas, kerahasiaan, dan ketersediaan data dalam suatu organisasi. Di era digital ini, vulnerability scanning merupakan teknologi kunci untuk mendeteksi kelemahan pada sistem dan jaringan komputer yang dapat memberikan peluang terjadinya serangan siber. Akan tetapi, metode vulnerability scanning secara manual seringkali kurang efisien, terutama pada lingkungan dengan banyak perangkat digital. Salah satu perusahaan yang kemungkinan besar membutuhkan dan juga memerlukan perubahan dari sistem manual ini adalah perusahaan penyedia layanan cloud, yang mana membutuhkan kemudahan perawatan terhadap sistem, perangkat, dan server yang digunakan dalam skala yang cukup besar atau masif. Penelitian ini difokuskan pada permasalahan efektivitas dan efisiensi dalam mendeteksi dan mengelola kerentanan keamanan pada sistem informasi. Untuk mengatasi permasalahan tersebut, maka dilakukan implementasi dan perbandingan pendekatan vulnerability scanning secara manual dengan pendekatan otomatis menggunakan OpenSCAP yang terintegrasi dengan Ansible. Percobaan dilakukan pada total 3 komputer target, dan analisis yang dilakukan adalah dengan membandingkan proses dan waktu yang dibutuhkan untuk mengimplementasikan kedua metode tersebut. Hasil penelitian menunjukkan bahwa penggunaan otomatisasi Ansible dapat mempengaruhi proses dan juga waktu yang dibutuhkan oleh sistem pemindaian kerentanan dimana pada sistem manual didapatkan total waktu sebesar 11.98s dan untuk sistem otomatisasi Ansible didapatkan total waktu sebesar 12.886s jika dilakukan pengujian pemindaian secara bersamaan pada ketiga perangkat komputer target. Berdasarkan literatur, waktu yang lebih lama ini dapat dipengaruhi oleh spesifikasi perangkat keras yang digunakan. Penelitian ini menyimpulkan bahwa pendekatan otomatisasi menggunakan Ansible dan OpenSCAP memiliki pengaruh yang relatif kecil apabila diaplikasikan pada penggunaan tiga perangkat. Terdapat peluang untuk dilakukan penelitian terkait dengan pengaruh spesifikasi perangkat keras yang digunakan terhadap lamanya percobaan.

Kata kunci*— *vulnerability scanning, ansible, time

I. PENDAHULUAN

Dalam keamanan sistem informasi, kemampuan untuk mendeteksi dan mengatasi sebuah vulnerability sangatlah dibutuhkan. Hal ini berguna untuk mengidentifikasi kerentanan pada sistem pertahanan, sehingga organisasi dapat memperbaiki kerentanan tersebut sebelum terjadinya serangan oleh pihak lain (Rizki, 2023). Metode untuk mendeteksi celah keamanan tersebut biasa dikenal sebagai *vulnerability scanning*.

Vulnerability Scannning merupakan sebuah metode atau proses untuk melakukan identifikasi kelemahan atau celah keamanan pada sistem komputer dan jaringan (Rizki, 2023). Hal ini merupakan metode penting yang dilakukan secara teratur agar dapat menjaga keamanan data dari serangan-serangan cyber. *Vulnerability scanning* tidak hanya dapat dilakukan dengan satu metode saja, melainkan ada beberapa metode yang dapat dilakukan terutama metode dengan pengecekan secara manual.

Namun dengan berkembangnya kebutuhan digital sekarang, melakukan *vulnerability scanning* dengan cara manual tidaklah efektif terutama jika suatu perusahaan menggunakan perangkat kerja digital yang terhitung banyak. Jika dilakukan *Vulnerability scanning* secara manual kepada perangkat kerja digital yang banyak, maka Perusahaan harus melakukan pengecekan secara satu per satu dari setiap perangkat yang mereka miliki.

Dilihat pada permasalahan diatas, OpenSCAP *(Security Content Automation Protocol)* dapat digunakan dan diaplikasikan untuk mengubah sistem *scanning* yang awalnya dilakukan secara manual, menjadi otomatis. OpenSCAP *(Security Content Automation Protocol)* sendiri merupakan seperangkat alat *open-source* yang digunakan untuk menerapkan dan mematuhi stpenggunar SCAP *(Security Content Automation Protocol)* bersertifikat NIST (National Institute of Standards and Technology). OpenSCAP *(Security Content Automation Protocol)* juga dapat diintegrasikan dengan Ansible, agar Perusahaan dapat melakukan pengecekan *vulnerability* kepada semua perangkat kerja digital tanpa harus melakukannya satu per satu. Btech (2023) menyatakan bahwa Ansible merupakan sebuah alat open-source untuk pengaturan perangkat lunak, pengelolaan konfigurasi, dan penerapan aplikasi. Alat ini dapat mengotomatisasi kan suatu proses konfigurasi dan pengelolaan *server*, serta penerapan dan pembaruan aplikasi.

Kerentanan pada sistem dapat dieksploitasi oleh pihak yang tidak bertanggung jawab untuk menimbulkan kerusakan atau mencuri data sensitif. Oleh karena itu, sangat penting untuk menerapkan metode yang efektif untuk mengidentifikasi dan mengelola kerentanan. Salah satu metode yang ada adalah melakukan otomasi pada proses *vulnerability scanning* menggunakan program seperti Ansible. Metode otomatisasi ini berpotensi untuk memudahkan proses *vulnerability scanning* pada perangkat dengan skala besar dibandingkan metode manual. Pada penelitian ini digunakan satu perangkat komputer *virtual* sebagai komputer kontroler dan tiga perangkat komputer *virtual* dengan OS dan spesifikasi yang identic sebagai *target*komputer Penelitian ini berfokus pada analisis perbandingan metode manual dan otomatis dalam konteks proses yang dilalui dan juga waktu yang diperlukan, yang bertujuan untuk mengetahui metode mana yang lebih efektif dalam meningkatkan keamanan sistem.

II. KAJIAN TEORI

A. *Vulnerability Scanning*

Vulnerability scanning, juga disebut "*Vulnerability Asessment*", adalah proses mengevaluasi jaringan atau aset TI untuk mengetahui adanya kerentanan keamanan - kekurangan atau kelemahan yang dapat dieksploitasi oleh pelaku ancaman eksternal atau internal. Pemindaian kerentanan adalah tahap pertama dari siklus manajemen kerentanan yang lebih luas. (Matt Kosinski, Amber Forrest. 2023)

B. Ansible

Btech (2023) menyatakan bahwa Ansible merupakan sebuah alat open-source untuk pengaturan perangkat lunak, pengelolaan konfigurasi, dan penerapan aplikasi. Alat ini dapat mengotomatisasi kan suatu proses konfigurasi dan pengelolaan server, serta penerapan dan pembaruan apliaksi. Ansible menggunakan bahasa sederhana yang mudah dibaca oleh manusia yang disebut YAML untuk menjelaskan tugas, dan dapat bekerja dengan beragam sistem dan teknologi.

C. *Security Content Automation Protocol* (SCAP)

SCAP merupakan solusi pemeriksaan kepatuhan stpenggunar untuk infrastruktur Linux tingkat perusahaan. SCAP adalah serangkaian spesifikasi yang dikelola oleh National Institute of Stpenggunards and Technology (NIST) untuk menjaga keamanan sistem untuk sistem perusahaan. (David Teimouri. 2019).

D. OpenSCAP

OpenSCAP adalah alat bantu audit yang menggunakan Extensible Configuration Checklist Description Format (XCCDF), yang merupakan format pengguna untuk mendefinisikan konten daftar periksa dan mendefinisikan daftar periksa keamanan. XCCDF juga dapat digunakan bersama dengan spesifikasi lain seperti CPE, CCE, dan OVAL untuk membuat daftar periksa yang diekspresikan SCAP yang dapat diproses oleh produk yang divalidasi SCAP

E. SSH Key

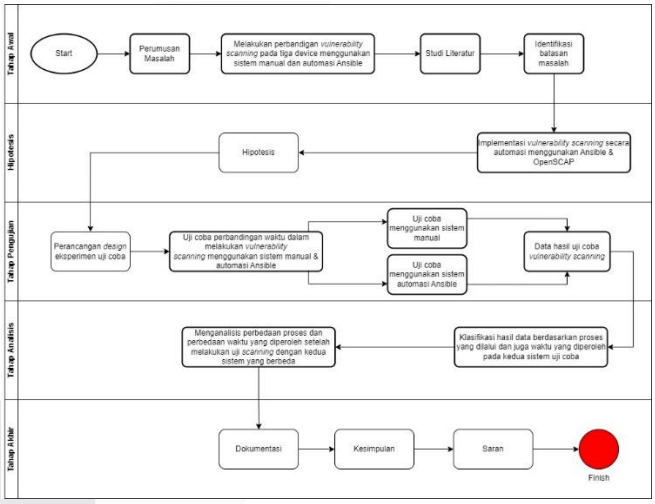
SSH (*Secure Shell*) Key adalah pasangan kunci kriptografi yang terdiri dari dua bagian yaitu *public key* dan *private key*. Mereka digunakan dalam suatu sesi SSH untuk mengenkripsi komunikasi antara klien dan *server* (Barret, 2005). SSH Key berperan sebagai alat untuk menghubungkan antara komputer utama dan juga ketiga komputer yang di*target*kan untuk melakukan *vulnerability scanning*.

F. YAML

YAML Ain't Markup Language (YAML) adalah bahasa serialisasi data yang sering kali berada di antara bahasa komputer yang paling populer. YAML biasanya digunakan sebagai format untuk file konfigurasi, tetapi kemampuan serialisasi objeknya menjadikannya alternatif potensial untuk bahasa seperti JSON atau Python (Erik Francis, 2023). YAML digunakan sebagai bahasa pemrograman untuk membuat perintah dalam Ansible-*playbook*.

III. METODE PENELITIAN

Sistematikan penyelesaian masalah adalah proses yang dilakukan peneliti untuk mencapai tujuan penelitian, dimana penelitian tersebut dibagi menjadi 5 tahapan yaitu Tahap Awal, Tahap Hipotesis, Tahap Pengujian, Tahap Analisi, dan Tahap Akhir. Berikut ilustrasi sistematikan penelitian yang dijelaskan dalam bentuk *flow chart*:



GAMBAR III. 1
Sistematika Penyelesaian Masalah

A. Tahap Awal

Tahap awal dimulai dengan melakukan identifikasi dari masalah terhadap latar belakang yang bertujuan untuk menggambarkan masalah yang akan diselesaikan. Setelah itu didapatkan permususan masalah dari penelitian dan akan mendapatkan juga batasan masalahnya. Batasan masalah sendiri bertujuan untuk membatasi permasalahan yang dibahas agar tidak menyimpang dari topik yang ada.

B. Tahap Hipotesis

Pada tahap ini melakukan pembuatan hipotesis yang merukapan praduga sementara. Terdapat hipotesis mengenai proses sistem manual dan otomasi Ansible dalam melakukan OpenSCAP *vulnerability scanning*.

C. Tahap Pengujian

Tahap ini diawali dengan melakukan perancangan terhadap *design* uji coba yang akan dilakukan, kemudian menjalankan uji coba *vulnerability scanning* menggunakan dua sistem yaitu secara manual dan juga secara otomasi menggunakan Ansible. Setelah melakukan simulasi uji coba, maka dilakukan pengumpulan data dari hasil simulasi dan pengukuran untuk keperluan analisis. Adapun data yang diambil adalah:

1. Proses dan waktu yang dibutuhkan pada sistem manual
2. Proses dan waktu yang dibutuhkan pada sistem otomasi ansible

D. Tahap Analisis

Pada tahap ini akan dilakukan analisis berdasarkan hasil-hasil yang sebelumnya telah didapat pada tahap pengujian. Tahap ini dilakukan untuk melihat perbandingan antara proses atau tahapan yang dilalui dan juga waktu yang dibutuhkan untuk proses berlangsung dari kedua sistem yang telah diuji.

E. Tahap Akhir

Tahapan ini merupakan tahapan terakhir dari penelitian yaitu membuat dokmentasi serta laporan akhir berdasarkan tahapan-tahapan yang telah dilalui. Tahap ini juga merupakan tahap pembuatan kesimpulan dari tahap awal hingga tahap analisis. Kesimpulan dan saran akan dibuat berdasarkan hasil dari uji coba dan simulasi yang sudah dilakukan

IV. PERANCANGAN DAN SKENARIO PENGUJIAN

A. Perancangan Sistem

Untuk mencapai tujuan penelitian yang telah direncakanan, diperlukan arsitektur yang terdiri dari *hardware* (perangkat keras) dan *software* (perangkat lunak) sebagai tahap awal untuk melakukan analisis pengujian. Dalam pembuatan perancangan system, dibutuhkan beberapa *hardware* dan *software* yang berupa alat, perangkat, dan aplikasi pendukung. Penelitian ini menggunakan *virtual machine* sebagai lingukan sistem untuk melakukan uji coba *vulnerability scanning* dengan Linux sebagai sistem operasi yang digunakan. Berikut merupakan *hardware* dan *software* yang digunakan:

1. Hardware

Pada proses uji coba dan perbandingan waktu dalam melakukan uji coba *vulnerability scanning* dengan sistem manual dan sistem otomasi Ansible. Dalam penelitian ini digunakan sejumlah empat perangkat *virtual*, yang dimana pada satu perangkat tersebut dijadikan komputer kontroler sebagai pemusatana otomasi menggunakan Ansible, kemudian tiga komputer dengan spesifikasi identic dijadikan sebagai target komputer untuk uji coba *vulnerability scanning*. Pada pengujian ini menggunakan OS (*Operating System*) Ubuntu 22.04 LTS karena dalam pengujian membutuhkan sebuah OS yang bersifat *open source*, stabil, dan juga fleksibel. Kapasitas penyimpanan dan memori yang digunakan pada penelitian ini merupakan ukuran spesifikasi minimal, dikarenakan pada penelitian ini tidak dibutuhkan terlalu banyak memori dan juga penyimpanan.

2. Software

Berikut adalah spesifikasi *software* yang digunakan dalam penelitian ini:

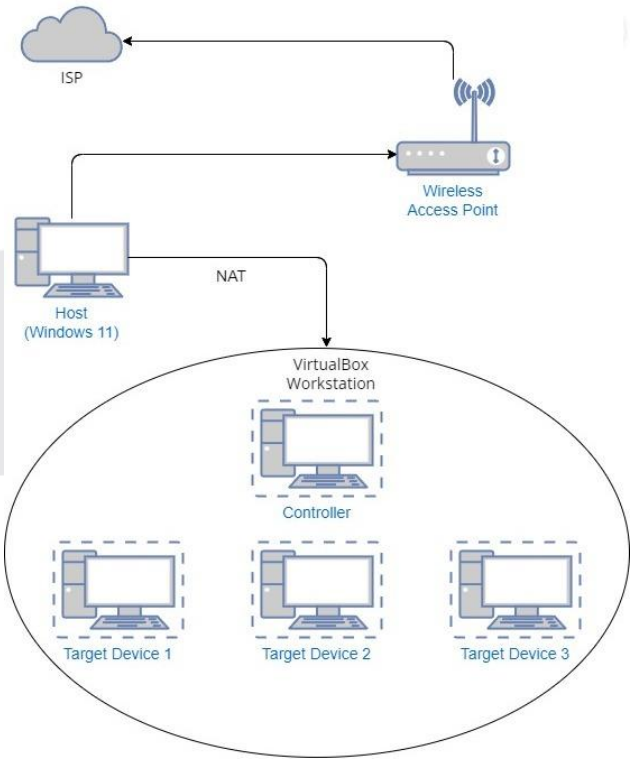
TABEL IV. 1
Software

Operating System	**Nama Aplikasi**	**Versi**
Windows 11 Home	Oracle VM VirtualBox	*Version* 7.0.14 r161095 (Qt5.15.2)
Ubuntu 22.04 LTS *Dekstop*	Ansible OpenSCAP	10.0.0 1.3.8
Ubuntu 22.04 LTS CLI	OpenSCAP	1.3.8
Ubuntu 22.04 LTS CLI	OpenSCAP	1.3.8
Ubuntu 22.04 LTS CLI	OpenSCAP	1.3.8

Tabel IV.2 Menjelaskan mengenai *software* yang digunakan dalam melakukan proses uji coba perbandingan waktu dalam melakukan otomisasi *vulnerability scanning*.

B. Perancangan Topologi

Penelitian yang dilakukan menggunakan sistem topologi *star* di mana terdapat satu OS/*device* yang menjadi *host* untuk keempat *virtual* OS/*device* melalui jaringan koneksi NAT secara *virtual*, pemilihan koneksi NAT diakarenakan agar tiap *virtual* device nya memiliki IP Address masing-masing dan ketika ada satu *server* yang bermasalah, *server* lain yang terhubung tidak akan ikut bermasalah.



GAMBAR IV. 1
Topologi

TABEL IV. 2
IP Address and Subnet Mask

Device	IP Address	Subnet Mask
Host (Windows 11)	192.168.18.10	255.255.255.0
Controller Computer	192.168.18.111	255.255.255.0
Target Device 1	192.168.18.114	255.255.255.0
Target Device 2	192.168.18.112	255.255.255.0
Target Device 3	192.168.18.113	255.255.255.0

Pada Gambar 4.1 menjelaskan bahwa topologi fisik yang digunakan pada penelitian ini terdiri dari 1 *Internet Service Provicer* (ISP), 1 *Device Controller*, dan 3 *Target Device*. Komputer *host* digunakan sebagai wadah/*device* dalam melakukan *virtual*isasi untuk uji coba, Controller digunakan sebagai *device* untuk menjalankan Ansible dan pusat dari management *device* lainnya dalam melakukan automatisasi dan uji coba. *Target Device* 1 hingga 3 berperan sebagai *device* yang akan melakukan uji coba OpenSCAP *vulnerability scanning* terhadap *server* kontroler Ansible. Komputer *host* yang sudah terpasang dengan *Virtual Machine* akan terhubung koneksinya dengan jaringan *internet*. *Internet* sendiri memiliki fungsi sebagai media pendukung agar implmentasi aplikasi yang digunakan dapat di install langsung kedalam *device* yang digunakan. NAT berfungsi sebagai tranlasi Alamat IP *public* ke IP *private* atau sebaliknya.

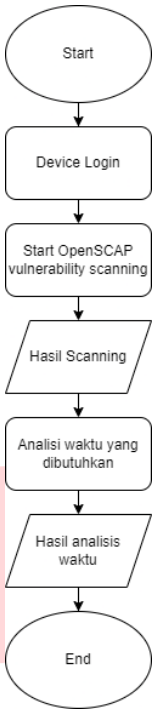
C. Simulasi Pengujian

Pada tahap penelitian ini, dilakukan simulasi uji coba dari sistem yang diimplementasikan pada lingukan LAN (*Local Area Network)* pada lingkungan *virtual* dengan penggunaan *virtual machine*. Simulasi ini dilakukan untuk mendemonstrasikan fungsionalitas dari sistem yang akan di uii. Peneliti menggunakan Oracle VM VirtualBox versi 7.0.14 untuk memvitualisasikan sistem yang dibangun sebagai simulasi prototipe. Adapun tujuan dari pembangunan prototipe simulasi yaitu untuk memenuhi sejumlah tujuan sebagai berikut:

a. Menjamin bahwa koneksi antar elemen atau komponen sistem berfungsi dengan baik.
b. Mengurangi kemungkinan kegagalan selama proses pembangunan dan implementasi sistem dalam dunia nyata.
c. Memastikan bahwa sistem yang digunakan telah menyelesaikan masalah dan memenuhi kriteria spesifikasi perancangan sistem.
d. Menjamin bahwa kesalahan yang terjadi selama proses perancangan, pembangunan, dan implementasi tidak mengganggu atau mempengaruhi lingkungan sistem.

1. Skenario Pengujian Manual Vulnerability scanning

Pada bagian ini membahas uji coba untuk memperoleh hasil waktu saat melakukan pengujian *vulnerability scanning* menggunakan *open-source software* OpenSCAP secara manual secara satu per satu yang kepada 3 komputer *target*.



GAMBAR IV. 2
Flow Chart Pengujian Manual

a. Instal OpenSCAP

Proses diawali dengan melakukan instalasi OpenSCAP yang akan digunakan untuk melakukan *vulnerability scanning*.

b. Melakukan set-up OpenSCAP

Pada proses ini melakukan setting dan juga persiapan pada software OpenSCAP yang akan digunakan, diantaranya yaitu memilih module file yang akan dihasilkan (Oval, XML, XCDDF, dll)

c. Menjalankan OpenSCAP *vulnerability scanning* pada komputer target

Pada proses ini dilakukan *vulnerability scanning* pada tiap komputer *target*dengan menggunakan open-source software OpenSCAP.

d. Hasil Scanning

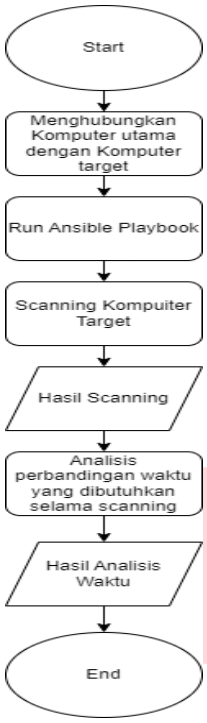
Hasil dari scan yang telah dilakukan berupa Common Vulnerability Scoring System (CVSS), Common Vulnerability and Exposures (CVE), penggolongan vulnerability yang terbagi kedalam kategori low, medium, dan high, dan waktu yang dilalui selama scan dilakukan. Namun, dalam penelitian ini akan berfokus pada perbandingan waktu yang dibutuhkan selama melakukan *vulnerability scanning* saja.

e. Analisis dan perbandingan waktu *vulnerability scanning* yang dilakukan

Melakukan pencatatan dan perbandingan waktu yang dibutuhkan selama melakukan OpenSCAP *vulnerability scanning* secara manual pada tiap komputer *target*yang diuji.

2. Skenario Pengujian Otomasi Vulnerability scanning

Pada bagian ini membahas uji coba untuk memperoleh waktu saat melakukan pengujian *vulnerability scanning* menggunakan *open-source software* OpenSCAP kepada tiga komputer *target* secara automisasi menggunakan Ansible pada komputer *controller*/utama



GAMBAR IV. 3
Flow Chart Pengujian Otomasi Ansible

1. Menghubungkan Komputer Utama dengan Komputer Target
Proses diawali dengan menghubungkan komputer utama/controller dengan 3 komputer *target*yang akan dilakukan uji coba, proses penghubungan menggunakan ssh-key yang akan di-generate pada komputer utama/controller kemudian dibagikan kepada 3 komputer target.
2. Instal OpenSCAP
Kemudian proses dilanjutkan dengan melakukan instalasi OpenSCAP yang akan digunakan untuk melakukan *vulnerability scanning*.
3. Instal Ansible
Melakukan instalasi Ansible pada komputer utama/controller agar dapat melakukan automisasi *vulnerability scanning*.
4. Membuat Ansible Playbook
Pada proses ini melakukan pembuatan perintah atau playbook yang berisikan perintah yang dapat dijalankan kepada ke 3 komputer target. Untuk pengujian, playbook berisikan perintah untuk melakukan OpenSCAP *vulnerability scanning* secara bersamaan.
5. Hasil scan
Hasil dari scan yang telah dilakukan berupa *Common Vulnerability Scoring System* (CVSS), *Common Vulnerability and Exposures* (CVE), penggolongan vulnerability yang terbagi kedalam kategori low, medium, dan high, dan waktu yang dilalui selama scan dilakukan. Pada pengujian yang dilakukan akan berfokus kepada waktu yang dibutuhkan untuk melakukan *vulnerability scanning* pada ke tiga komputer *target*secara bersamaan.
6. Analisis dan perbandingan waktu *vulnerability scanning* yang dilakukan
Proses terakhir adalah melakukan pencatatan dan perbandingan waktu yang dibutuhkan selama melakukan OpenSCAP *vulnerability scanning* secara automisasi melalui Ansible pada komputer kontroler terhadap ke tiga komputer *target* secara bersamaan.

D. Data Hasil Design
Pada bagian ini ditampilkan hasil dari simulasi dan hasil implementasi *vulnerability scanning* dari OpenSCAP secara manual dan juga secara automatis menggunakan Ansible.

1. Data Implementasi Manual OpenSCAP Vulnerability Scanning

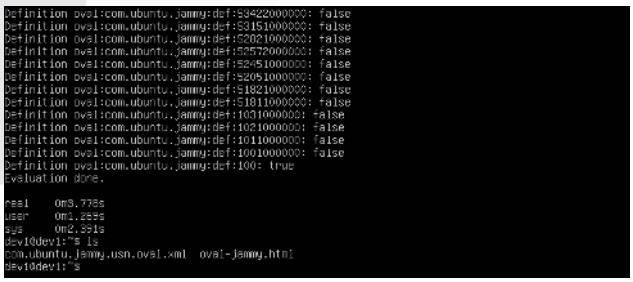
Bagian ini menjelaskan mengenai penggunaan ke 3 komputer target untuk melakukan OpenSCAP *vulnerability scanning* secara manual pada masing masing komputer target yang akan dijelaskan pada tahapan sebagai berikut:

a. Melakukan instalasi OpenSCAP sebagai software untuk melakukan *vulnerability scanning* pada tiap device yang digunakan dengan *command* “**~$ wget**”

Gambar IV. 4 Input dan Ouput Instalasi OpenSCAP

b. Setelah berhasil melakukan instalasi OpenSCAP pada tiap komputer *target* yang akan digunakan, memasukkan *command* “**~$ time oscap oval eval --report oval-jammy.html com.ubuntu.jammy.usn.oval.xml”** untuk menjalankan perintah *vulnerability scanning* beserta penghitungan waktu saat proses scanning berlangsung.

Gambar IV. 5 Input Command Vulnerability Scanning



Gambar IV. 6 Ouput Vulnerability Scanning

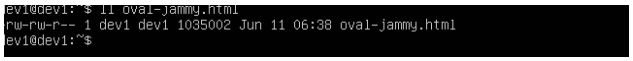
c. Setelah proses *vulnerability scanning* selesai berlangsung, maka didapatkan hasil dari waktu yang diperoleh setelah melakukan manual OpenSCAP *vulnerability scanning* sebagai berikut:

Tabel IV. 3 Hasil Waktu Manual Vulnerability Scanning

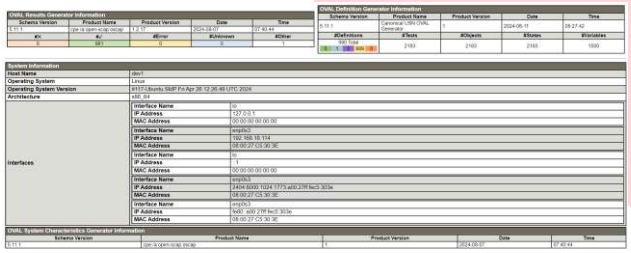
Tipe Scanning	Komputer Target	Waktu
	Dev1@192.168.18.114	3,788 detik

Manual Scanning	Dev2@192.168.18.112	4,124 detik
	Dev3@192.168.18.113	4,078 detik
	Total Jumlah	11,990 detik

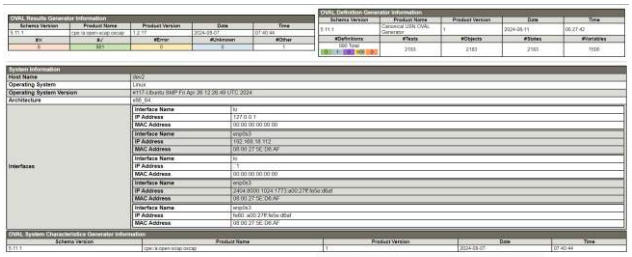
d. Gunakan *command*
"**$ ll [file name].html**"
untuk membuka file *report* yang dihasilkan setelah eksekusi *vulnerability scanning* dijalankan, file r*eport* yang dihasilkan bersifat "html" sehingga akan terbuka pada *web browser*.



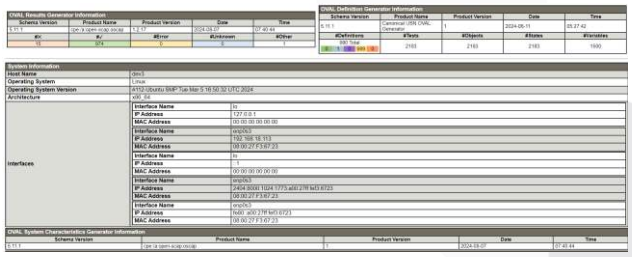
GAMBAR IV. 7
Input & Output Command Membuka File Pada Sistem Manual



GAMBAR IV. 8
File Report Scan Pada Device 1 Pada Sistem Manual



GAMBAR IV. 9
File Report Scan Pada Device 2 Pada Sistem Manual

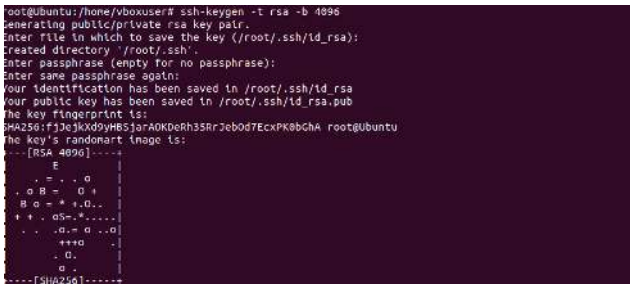


GAMBAR IV. 10
File Report Scan Pada Device 3 Pada Sistem Manual

2. Data Implementasi Otomasi OpenSCAP Vulnerability Scanning

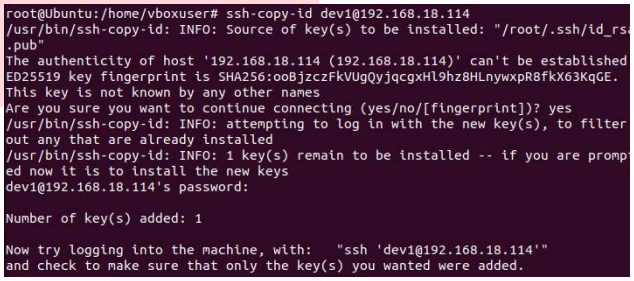
Pada bagian ini menjelaskan mengenai komputer kontroler yang mengoperasikan Ansible untuk melakukan otomasi *vulnerability scanning* pada ke 3 komputer target secara bersamaan, yang akan dijelaskan pada tahapan berikut:

a. Masuk kedalam terminal pada komputer kontroler sebagai root lalu menjalankan *command*
"**~$ ssh-keygen -t rsa -b 4096**"
untuk menghasilkan SSH Key yang akan digunakan untuk menghubungkan tiap masing-masing device.

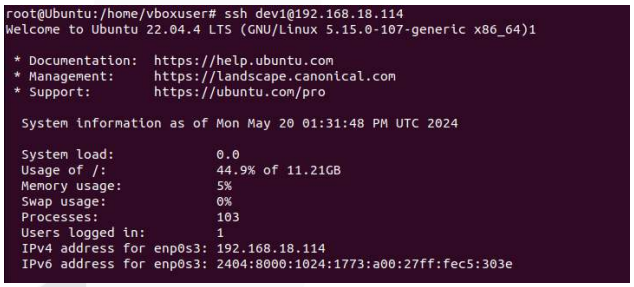


GAMBAR IV. 11
Input dan Output Command Generate SSH Key

b. Setelah ssh key berhasil dibuat, masukkan *command*
"**~$ ssh-copy-id user@ip**"
untuk menyebarkan ssh key pada tiap device komputer target yang ingin disambungkan. Masukkan *command*
"**~$ ssh 'user@ip'**"
untuk masuk kedalam device tersebut agar dapat memastikan bahwa device telah terhubung.

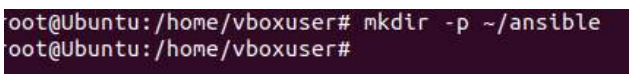


GAMBAR IV. 12
Input dan Output SSH Key Distribution



GAMBAR IV. 13
Input dan Output Untuk Masuk Kedalam Device

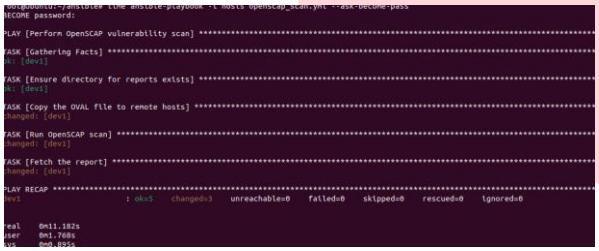
c. Buat dan masuk kedalam direktori '***root@ubuntu:~/***Ansible#**' sebagai direktori lalu membuat Ansible *playbook* dengan command :
"$ nano ~/ansible/ping.yml"
Playbook berisikan beberapa parameter dan juga perintah yang diantaranya untuk memastikan perintah terjalankan kepada 3 komputer *target* yang dituju, melakukan OpenSCAP *vulnerability scanning*, dan melakukan penyimpanan hasil scanning pada masing-masing komputer *target* kedalam komputer kontroler.



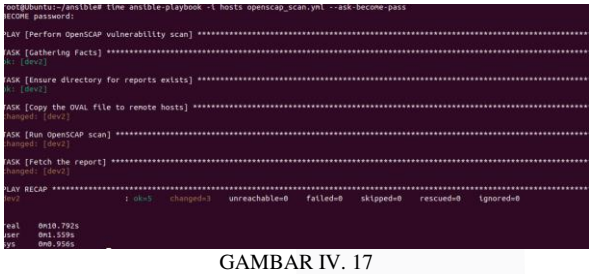
GAMBAR IV. 14
Pembuatan Ansible Direktori

GAMBAR IV. 15
Pembuatan Playbook

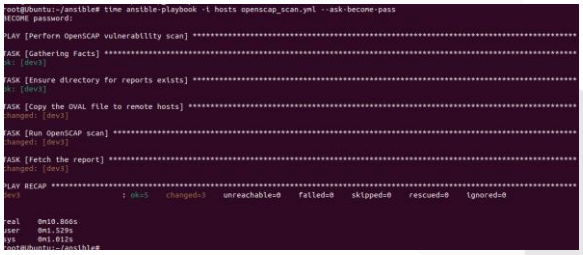
d. Setelah jalankan Ansible *Playbook* yang telah dibuat dan dikonfigurasi, maka akan muncul hasil setelah berhasil melakukan OpenSCAP *vulnerability scanning* menggunakan Ansible dan juga waktu yang dibutuhkan dalam melakukan scanning tersebut.



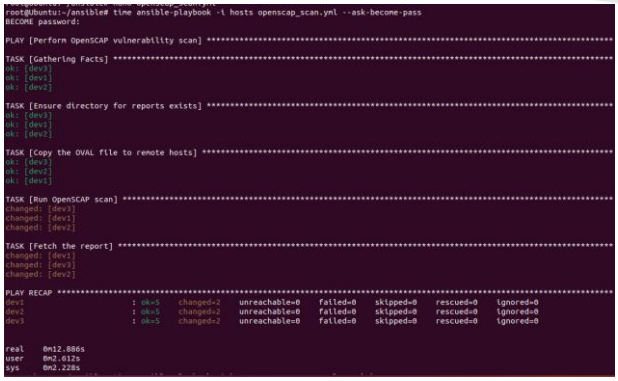
GAMBAR IV. 16
Hasil Scan Otomasi Pada Device 1



GAMBAR IV. 17
Hasil Scan Otomasi Pada Device 2

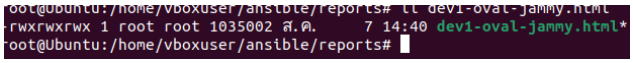


GAMBAR IV. 18
Hasil Scan Otomasi Pada Device 3

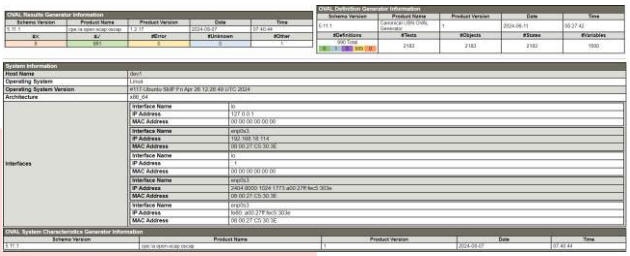


GAMBAR IV. 19
Hasil Scan Otomasi Pada Ketiga Device Secara Bersama

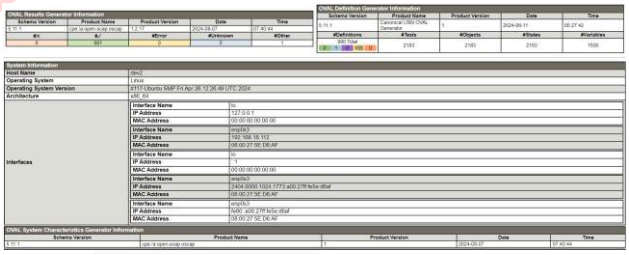
e. Gunakan *command* “**$ ll [file name].html**” untuk membuka file *report* yang dihasilkan setelah eksekusi *vulnerability scanning* dijalankan, file *report* yang dihasilkan bersifat “html” sehingga akan terbuka pada *web browser.*



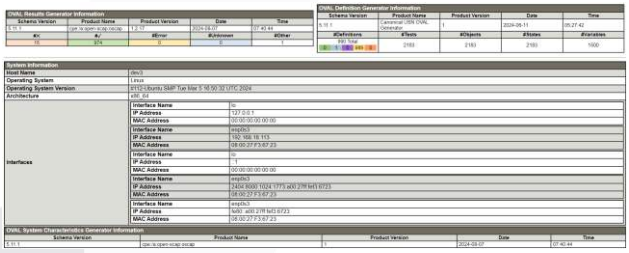
GAMBAR IV. 20
Input & Output Command Membuka File



GAMBAR IV. 21
File Report Scan Pada Device 1 Pada Sistem Otomasi Ansible



GAMBAR IV. 22
Gambar IV. 22 File Report Scan Pada Device 2 Pada Sistem Otomasi Ansible



GAMBAR IV. 23
Gambar IV. 23 File Report Scan Pada Device 3 Pada Sistem Otomasi Ansible

f. Berdasarkan hasil waktu yang diperoleh dari uji coba otomasi *vulnerability scanning*, dapat dilihat sebagai berikut:

TABEL IV. 4
Perolah Waktu Scanning Secara Otomasi

Tipe Scanning	Komputer target	Waktu
Automatic Scanning	Dev1@192.168.18.114	11,182 detik
	Dev2@192.168.18.112	10,792 detik
	Dev3@192.168.18.113	10,866 detik
	Semua device bersamaan	12,866 detik

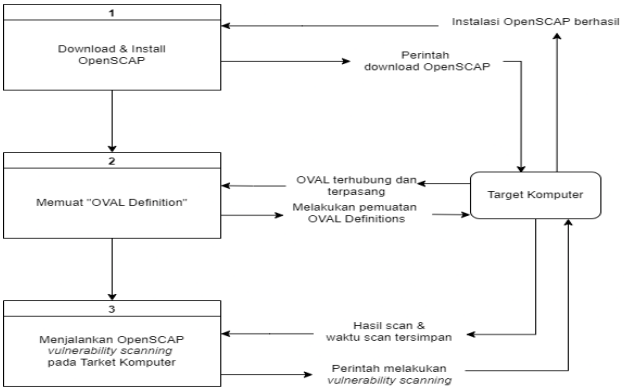
V. HASIL DAN ANALISIS

A. Analisis Data Flow Diagram (DFD)

Data flow diagram merupakan suatu *diagram* yang berisikan dan menampilkan data *input* serta data *output*. Pada uji coba ini DFD digunakan untuk memaparkan langkah-langkah, data masukkan dan juga data keluaran yang diperoleh ketika melakukan OpenSCAP *vulnerability scanning* baik secara manual maupun secara otomasi

1. DFD Manual OpenSCAP Vulnerability Scanning

Berikut merupakan DFD (Data Flow Diagram) yang menggambarkan proses uji coba *vulnerability scanning* secara manual yang telah dilakukan



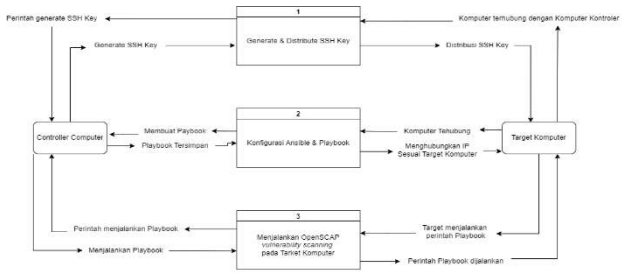
GAMBAR V. 1
Data Flow Diagram Manual Vulnerability scanning

Gambar V.1 mengenai uji coba untuk melakukan OpenSCAP Vulnerability Testing kepada 3 komputer target menggunakan cara manual yaitu mengoperasikannya pada setiap device komputer target.

1. Tahap pertama pada uji ini yaitu melakukan download dan instalasi OpenSCAP pada setiap komputer target menggunakan perintah
"~$ apt -y install libopenscap8 bzip2".
OpenSCAP berguna sebagai software berbasis open-source untuk melakukan vulnerability scanning.
2. Kemudian memuat OVAL (*Open Vulnerability and Assessment Language*) *Definitions* yang merupakan bahasa yang akan digunakan oleh OpenSCAP untuk nantinya menjalankan dan juga menyimpan data hasil dari vulnerability scanning dengan command
"~$ bzip2 -d com.ubuntu.jammy.usn.oval.xml.bz2".
3. Tahap terakhir yaitu menjalankan dan melakukan vulnerability scanning menggunakan software OpenSCAP dengan command
"~$ time oscap oval eval --report oval-jammy.html com.ubuntu.jammy.usn.oval.xml".
Setelah perintah dijalankan, maka sistem akan mengeluarkan dan menyimpan hasil dan juga waktu yang dibutuhkan saat melakukan vulnerability scanning.
4. Pada tahap terakhir dilakukan pengukuran waktu eksekusi vulnerability scanning dari awal hingga akhir untuk mendapatkan gambaran penggunaan waktu yang dibutuhkan untuk melakukan scanning, pengambilan waktu dilakukan dengan menggunakan command
"~$ time"

2. DFD Automatic OpenSCAP Vulnerability Scanning

Berikut merupakan DFD (Data Flow Diagram) yang menggambarkan proses uji coba *vulnerability scanning* secara otomasi menggunakan Ansible yang telah dilakukan:



GAMBAR V. 2
Data Flow Diagram Automatic Vulnerability scanning

Gambar V.2 mengenai uji coba untuk melakukan OpenSCAP Vulnerability Testing kepada 3 komputer *target* menggunakan Ansible pada komputer utama untuk melakukan otomisasi agar tidak perlu melakukan secara manual satu per satu pada 3 komputer *target*.

1. Tahap pertama yaitu menghubungkan semua komputer (Komputer utama dan 3 Komputer target), koneksi dilakukan dengan cara men-generate ssh key dengan command
"~$ ssh-keygen -t -rsa -b 4096".
Kemudian ssh key yang telah dibuat pada komputer utama akan dicopy dan didistribusi kepada 3 komputer target dengan perintah
"~$ ssh-copy-id".
2. Tahap kedua setelah memastikan bahwa semua komputer terhubung adalah melakukan konfigurasi Ansible dan juga Playbook pada komputer kontroler agar perintah yang dimasukkan dapat diterjalankan pada tiga komputer target lainnya. Konfigurasi dilakukkan dengan cara membuat Playbook yang berisikan beberapa perintah diantaranya yaitu; menjalankan memastikan semua komputer target memiliki OpenSCAP, menjalankan OpenSCAP untuk melakukan vulnerability scanning, lalu menyalin dan menyimpan data hasil scanning pada sebuah direktori di computer utama. Pada Playbook tersebut user juga harus memasukkan IP yang dimiliki komputer target agar perintah tersebut terhubung dengan komputer yang ditargetkan.
3. Tahap terakhir yaitu menjalankan playbook yang telah dibuat sebelumnya untuk melakukan OpenSCAP vulnerability scanning pada setiap komputer target dengan perintah
"~$ ansible-playbook -i hosts openscap_scan.yml".
Setelah uji coba berhasil tambahkan command "time" agar komputer mendeteksi dan menghitung jumlah waktu yang dibutuhkan untuk menjalankan playbook hingga vulnerability scanning selesai dilakukan, sehingga command yang dijalankan untuk uji coba terakhir yaitu
"~$ time ansible-playbook -i hosts openscap_scan.yml".
4. Pada tahap terakhir dilakukan pengukuran waktu eksekusi vulnerability scanning dari awal hingga akhir untuk mendapatkan gambaran penggunaan waktu yang dibutuhkan untuk melakukan scanning, pengambilan waktu dilakukan dengan menggunakan command
"~$ time"

B. Pengukuran Time Pada Uji Coba OpenSCAP Vulnerability Scanning

Dalam skenario pengujian yang didasarkan pada hasil perbandingan waktu, pengukuran *time* bertujuan untuk mengamati, mengukur, dan mencatat jumlah waktu yang dilalui untuk setiap proses *vulnerability scanning* yang dilalui. Tujuan dari pengukuran ini adalah untuk mengetahui seberapa efektif, efisien, dan cepat proses *scanning* dilakukan pada kedua tipe proses yang dilalui. Jenis pengukuran *time* pada pengujian dapat dikategorikan menjadi tiga mertrik yaitu *Real Time*, *User Time*, dan *System Time*

Pada uji coba ini hanya akan berfokus pada perbanding waktu *real time* yang diperoleh dari kedua uji coba yang dilakukan karena *real time* merupakan hasil waktu dari saat *vulnerability scanning* dijalankan hingga selesai dijalankan

1. Hasil Pengukuran Real Time Manual OpenSCAP Vulnerability scanning

Pada pengukuran *time* yang dilakukan berdasarkan hasil uji coba *vulnerability scanning* bertujuan untuk mengumpulkan informasi tentang jumlah waktu yang dihabiskan selama pengujian berlangusng. Waktu ini diambil berdasarkan uji coba sebanyak satu kali pada tiga komputer target yang berbeda. Berikut merupakan data hasil pengukuran *time* dari uji coba yang terdapat pada Tabel V.1:

TABEL V. 1
Hasil Pengukuran Time Manual OpenSCAP Vulnerability scanning

Tipe Scanning	Komputer target	Time Metrik (s)		
		Real	*User*	*System*
Manual	*Device* 1	3,778s	1,269s	2,391s
	Device 2	4,124s	1,313s	2,673s
	Device 3	4,078s	1,199s	2,753s

TABEL V. 2
Hasil Total Pengukuran Time Manual OpenSCAP Vulnerability scanning

Tipe Scanning	Komputer target	Time Metrik (s)
		Real
Manual	*Device* 1	3,778s
	Device 2	4,124s
	Device 3	4,078s
Real Time Total		11.98s

2. Hasil Pengukuran Real Time Automatic OpenSCAP Vulnerability scanning

Pada pengukuran *time* yang dilakukan berdasarkan hasil uji coba *vulnerability scanning* bertujuan untuk mengumpulkan informasi tentang jumlah waktu yang dihabiskan selama pengujian berlangusng. Waktu ini diambil berdasarkan uji coba sebanyak tiga kali secara terpisah yaitu dengan cara melakukan *scanning* secara automatis melalui komputer kontroler menggunakan Ansible pada ketiga komputer target. Berikut merupakan data hasil pengukuran *time* dari uji coba yang terdapat pada Tabel V.3:

TABEL V. 3
Hasil Pengukuran Real Time Ansible Automatic OpenSCAP Vulnerability Scanning

Tipe Scanning	Komputer target	Time Metrik (s)		
		Real	*User*	*System*
Automatic menggunakan Ansible	*Device* 1	11,182s	1,768s	0,895s
	Device 2	10,792s	1,559s	0,956s
	Device 3	10,866s	1,529s	1,012s

Setelah berhasil melakukan perhitungan waktu uji coba otomasi pada tiga *device* secara terpisah, selanjutnya akan dilakukan perhitungan *real time* jika otomasi dilakukan kepada tiga *device target* secara bersamaan melalui komputer kontroler menggunakan Ansible. Berikut merupakan data hasil pengukuran *real time* secara bersamaan dari uji coba yang terdapat pada table V.4:

TABEL V. 4
Hasil Pengukuran Real Time Ketiga Device Secara Bersamaan Ansible Automatic OpenSCAP Vulnerability Scanning

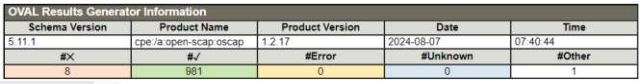
Tipe Scanning	Komputer target	*Real Time* (s)
Automatic menggunakan Ansible	Semua *device* target secara bersamaan menggunakan otomasi	12,886s

C. Scanning Result Dari Uji Coba OpenSCAP Vulnerability Scanning

Dalam pengujian OpenSCAP *vulnerability scanning* didapatkan laporan atau *report* yang dihasilkan setelah perintah *vulnerability scanning* dieksekusi, Laporan tersebut mencakup informasi tentang kerentanan, kepatuhan standar keamanan, dan rekomendasi perbaikan. Dalam laporan tersebut juga terdapat tingkat kerentanan yang terbagi menjadi tiga bagian yaitu *High Severity, Medium Severity, Low Severity*

1. *Scanning Result* Uji Coba Dengan Sistem Manual

Laporan OpenSCAP *vulnerability scanning* yang dihasilkan memberikan informasi terperinci tentang kerentanan yang ditemukan, statusnya, dan postur keamanan sistem secara keseluruhan. Data hasil *scan* ini merupakan hal penting untuk mengidentifikasi potensi risiko keamanan. Pada uji coba menggunakan sistem manual, OS dan juga *device* yang digunakan bersifat identik antara ketiga komputer target, sehingga isi dari file *scanning result* tidak memiliki banyak perbedaan. Pada sistem uji coba manual, file *scanning result* harus dibuka pada masing-masing *device* yang telah dilakukan *vulnerability scanning*.


OVAL Results Generator Information

Schema Version	Product Name	Product Version	Date	Time
5.11.1	cpe:/a:open-scap:oscap	1.2.17	2024-08-07	07:40:44
#✕	#✓	#Error	#Unknown	#Other
8	981	0	0	1

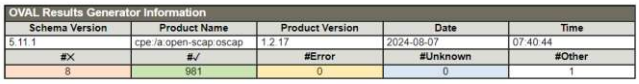
Gambar V. 3 Scanning Result General Information Pada Sistem Manual

Pada gambar V.3 memiliki beberapa bagian, diantaranya yaitu:
1. Schema Version
2. Product Name
3. Date and Time
4. Result Summary

2. *Scanning Result* Uji Coba Dengan Sistem Otomasi Ansible

Laporan OpenSCAP *vulnerability scanning* yang dihasilkan memberikan informasi terperinci tentang kerentanan yang ditemukan, statusnya, dan postur keamanan sistem secara keseluruhan. Data hasil *scan* ini merupakan hal

penting untuk mengidentifikasi potensi risiko keamanan. Pada uji coba menggunakan sistem ini, file *scanning result* yang dimiliki komputer target sudah tersimpan pada komputer kontroler sehingga hanya perlu dibuka pada melalui komputer kontroler saja.


OVAL Results Generator Information

Schema Version	Product Name	Product Version	Date	Time
5.11.1	cpe:/a:open-scap:oscap	1.2.17	2024-08-07	07:40:44
#X	#✓	#Error	#Unknown	#Other
8	981	0	0	1

GAMBAR V. 4
Scanning Result General Information Pada Sistem Otomasi Ansible

Pada gambar V.4 memiliki beberapa bagian, diantaranya yaitu:

1. Schema Version
2. Product Name
3. Date and Time
4. Result Summary

D. Analisis Perbanding Proses, Waktu, dan Scanning Result Dari Manual dan Automatic Vulnerablity Scanning

Data flow diagram dan metrik *real time* digunakan untuk menganalisis *vulnerability scanning* dengan tujuan untuk mengetahui serta mengevaluasi penggunaan dua tipe *scanning* yang berbeda untuk mendapat tipe langkah yang efektif dan efisien. DFD mengacu pada proses dan tahapan yang dilalui dan juga input dan output yang dihasilkan selama proses uji coba berlangsung. Metrik *time* mengacu pada waktu dari proses *vulnerability scanning* berlangsung dari awal hingga akhir dalam satuan detik. Sedangkan *scanning result* merupakan data yang diperoleh dari hasil eksekusi *vulnerability scanning* yang telah dijalankan.

1. Analisi Perbandingan Proses

Analisis dari proses yang berada pada *data flow diagram* diawali dari proses *set-up device*, penghubungan tiap *device,* download dan install aplikasi yang digunakan, hingga tahap keluaran hasil dari uji coba *vulnerability scanning*. Berikut merupakan tabel yang menunjukkan analisis perbandingan yang dinilai berdasarkan beberapa aspek diantaranya yaitu:

TABEL V. 5
Ringkasan Analisi Perbandingan Data Flow Diagram

Aspek	Ansible *Automation*	Manual
Otomasi	Automatis secara penuh melalui Ansible dan Ansible Playbook	Memerlukan eksekusi manual untuk setiap langkah.
Kompleksitas Pengaturan	Pengaturan awal yang lebih kompleks (distribusi *SSH key*, pembuatan Ansible *Playbook*).	Pengaturan awal yang lebih sederhana pada satu sistem, namun lebih rumit jika menggunakan banyak sistem.
Kemudahan Konfigurasi Pada Tiap Sistem	Memastikan konfigurasi yang sama pada semua sistem target.	Konfigurasi tergantung pada eksekusi manual dan dapat bervariasi jika tidak disamakan pada tiap *device* nya.
Upaya Pengguna	Upaya pengguna lebih minimal setelah pengaturan awal terselesaikan.	Upaya pengguna yang tinggi diperlukan untuk mengatur pada tiap sistem target yang digunakan.
Risiko Kesalahan Manusia	Risiko kesalahan manusia yang lebih rendah karena otomatisasi.	Risiko kesalahan manusia yang lebih tinggi selama eksekusi manual.
Pusat Kontrol	Kontrol terpusat dari komputer kontroler.	Tidak terpusat; setiap komputer target dikelola secara individual.

2. Analisis Perbandingan Waktu (Real Time)

Pengukuran *time* yang dilakukan berdasarkan berapa lama proses *vulnerability scanning* dilakukan, waktu proses yang diukur adalah ketika menjalankan perintah untuk *scan* hingga hasil *scan* keluar yang kemudian dinilai dalam nilai detik (s).

Berikut merupakan tabel yang menunjukkan perbandingan *real time* yang dihasilkan dari proses melakukan *vulnerability scanning*:

TABEL V. 6
Perbandingan Real Time Manual

Tipe *Scanning*	**Device 1 (s)**	**Device 2 (s)**	**Device 3 (s)**	**Total Time for All Devices (s)**
Manual *Scanning*	3.778	4.124	4.078	11.98
Waktu dari *scan* secara bersamaan				Tidak dapat melakukan scan secara bersamaan

TABEL V. 7
Perbandaingan Real Time Otomasi Ansible

Tipe *Scanning*	**Device 1 (s)**	**Device 2 (s)**	**Device 3 (s)**	**Total Time for All Devices (s)**
Ansible *Automation*	11.182	10.792	10.866	32.84
Waktu dari *scan* secara bersamaan				12.886 s

Berdasarkan Tabel V.4 dan Tabel V.5 perbandingan *real time* diatas, maka diperoleh analisis sebagai berikut:

a. Pemindaian manual lebih cepat per perangkat, dengan *real time* diantara 3,778s hingga 4,124s.
b. Ansible membutuhkan lebih banyak waktu per *device* (sekitar 10,7s hingga 11,2s), tetapi memungkinkan *scan* pada *device* dalam jumlah yang banyak secara sekaligus.

3. Analisis Perbandingan Scanning Result Dari Vulnerability Scanning

Berdasarkan bagian *scanning result* pada uji coba OpenSCAP *vulnerability scanning* menggunakan sistem manual dan sistem otomasi Ansible, didapat analisis yaitu:

a. Untuk membuka file pada sistem manual, perlu dilakukan pada tiap device yang diuji. Hal ini dikarenakan file akan tersimpan otomatis pada device yang menjalankan perintah *vulnerability scanning*. Sedangkan pada sistem otomasi Ansible, eksekusi perintah dijalankan pada komputer kontroler sehingga file *scanning result* pada ketiga *device* komputer target akan tersimpan pada komputer kontroler. Sehingga jika ingin membuka file tersebut, dapat dilakukan pada komputer kontroler.
b. Tidak terdapat perbedaan pada *scanning result* yang dilakukan melalui sistem manual maupun sistem otomasi Ansible, hal ini dikarenakan Ansible hanya digunakan untuk menjalankan dan menyalurkan perintah eksekusi OpenSCAP *vulnerability scanning*. Sehingga otomasi menggunakan Ansible tidak mempengaruhi hasil dari uji *scan*

VI. KESIMPULAN

A. Kesimpulan

Proses *vulnerability scanning* dipengaruhi oleh penerapan sistem otomasi menggunakan Ansible. Ansible memungkinkan pemindaian berbagai perangkat secara bersamaan, membuat *vulnerability scanning* lebih efisien dalam situasi di mana banyak *device* yang perlu dipindai secara bersamaan. Lalu, Implementasi *vulnerability scanning* menggunakan Ansible dan OpenSCAP terbukti berpengaruh dalam mengidentifikasi dan mengelola kerentanan keamanan. Dengan menggunakan konfigurasi Ansible yang sama, didapatkan hasil yang sama pada tiap sistem *device* yang dikonfigurasi, sehingga system ini dapat diandalkan karena mengurangi risiko *human error* yang sering terjadi dalam proses konfigurasi satu per satu pada tiap devicenya.Secara keseluruhan, pada jumlah tiga *device* waktu respon yang diperoleh didapatkan waktu yang lebih besar, sehingga diperkirakan pada jumlah *device* yang lebih besar akan didapatkan waktu respon yang lebih cepat pada penggunaan sistem otomasi Ansible dan OpenSCAP. Hal ini dipengaruhi aspek jumlah dan juga spesifikasi dari *device* yang diuji serta *software* OpenSCAP. Berdasarkan literatur, spesifikasi dari sebuah *device* dan sifat *software* yang digunakan dapat mempengaruhi baik itu mempercepat maupun memperlambat waktu dari eksekusi *scan*. Pada sejumlah tiga *device* komputer target yang menggunakan OS dan juga spesfikasi identik. *Scanning result* yang dihasilkan oleh sistem manual maupun sistem otomasi Ansible tidak memiliki perbedaan, hal ini dikarenakan pada sistem otomasi, Ansible digunakan sebagai *software* atau *tools* untuk mengotomatiskan penyebaran perintah untuk melakukan eksekusi uji coba OpenSCAP *vulnerability scanning* pada ketiga *device* komputer target. Sehingga pada pemilihan pengguna sistem yang akan digunakan, akan lebih condong kepada perbandingan proses yang dilalui kedua sistem, kemudahan penggunaan pada tiap sistem, dan juga hasil perolehan waktu yang didapatkan pada kedua sistem.

B. Saran

Sebagai saran yang dapat digunakan untuk peluang sebagai kelanjutan dari penelitian ini adalah:

1. Pengoptimalan pada *playbook* Ansible dan konfigurasi OpenSCAP untuk meningkatkan efektivitas dan mengurangi waktu *scan* pada tiap *device* tanpa mengorbankan kualitas dan jangka *scan*.
2. Menganalisis dan menguji coba aplikasi atau *software* otomasi dan *software vulnerability scanning* lainnya, untuk menemukan kelebihan dan kelemahan *tools* yang telah digunakan
3. Peluang penelitian terkait dampak penggunaan Ansible dan software *vulnerability scanning* OpenSCAP terhadap beban yang diberikan pada *hardware* yang digunakan
4. Peluang penelitian terkait dampak spesifikasi *hardware* yang digunakan kepada durasi waktu percobaan

REFERENSI

[1] GeeksForGeeks. (2024). *Difference Between User-CPU-Time and System-CPU-Time in UNIX*. https://www.geeksforgeeks.org/difference-between-user-cpu-time-and-system-cpu-time-in-unix/

[2] Lakshmanan, R. (2016). *REAL TIME IS GREATER THAN USER AND SYS TIME*. https://blog.gceasy.io/real-time-greater-than-user-and-sys-time/

[3] Lucidchart. (2024). *What is a Data Flow Diagram*. https://www.lucidchart.com/pages/data-flow-diagram

[4] Rizki. (2023). *Vulnerability scanning: Pengertian, Manfaat, Hingga Cara Kerjanya*. https://r17.co.id/insight/article/vulnerability-scanning-pengertian-manfaat-hingga-cara-kerjanya

[5] Ramadhan, Harry Wahyu (2021) *Implementasi dan Analisis Security Auditing Menggunakan Open Source Vulnerability Scanner Software Pada Server Kontroler Ansible*

[6] Maulan, Dimas Bayu (2021) *Perancangan dan Realisasi Sistem Otomasi Manajemen Konfigurasi Jaringan Menggunakan Ansible dan Elasticsearch (Studi Kasus: Bagian Pengembangan Jaringan Di Direktorat Sistem Informasi Telkom University Di Gedung Tokong Nanas)*

[7] Alwi, H., & Umar (2020) *Analisis Keamanan Website Menggunakan Teknik Footprinting dan Vulnerability scanning*

[8] Btech (2023) *Configuration Management Skills Building with Ansible* https://www.btech.id/en/news/configuration-management-skills-building-with-ansible/

[9] RedHat (n.d) Chapter 16. *Scanning the system for security compliance and vulnerabilities* https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/8/html/system_design_guide/scanning_the_system_for_security_compliance_and_vulnerabilities

[10] RedHat (n.d) Chapter 8. *Compliance and Vulnerability Scanning with OpenSCAP* https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/6/html/security_guide/chap-compliance_and_vulnerability_scanning#sect-Security_Compliance_in_RHEL

[11] Barret, D.J. (2005). *SSH, the Secure Shell: The Definitive Guide, Second Edition*

[12] Teimouri, D. (2018). *What is OpenSCAP?. Virtualization and Data Center* https://www.teimouri.net/2018/12/

[13] Kosinski, M., & Forrest, A. (2023). *What is Vulnerability Scanning? https://www.ibm.com/id-id*

[14] Irawan, Alfian Rifki (2023) *Implementasi dan Analisis Attack Tree Pada Aplikasi DVWA Berdasarkan Metrik Time dan Probability*