เครื่องมือสำหรับตรวจสอบการตั้งค่าตามข้อกำหนดความปลอดภัย

Security Configuration Compliance Audit Tool

นายณัฐวิทย์ ลีลาวรรณศิลป 5804800059

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

มหาวิทยาลัยสยาม

ปีการศึกษา 2561

หัวข้อปริญญานิพนธ์	เครื่องมือสำหรับตรวจสอบการตั้งค่าตามข้อกำหนดความปลอดภัย Security Configuration Compliance Audit Tool
หน่วยกิตของปริญญานิพนธ์	3 หน่วยกิต
รายชื่อผู้จัดทำ	นายณัฐวิทย์ ลีลาวรรณศิลป 5804800059
อาจารย์ที่ปรึกษา	อาจารย์จรรยา แหยมเจริญ
ระดับการศึกษา	วิทยาศาสตรบัณฑิต
ภาควิชา	วิทยาการคอมพิวเตอร์
ปีการศึกษา	2561

อนุมัติให้ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์

คณะกรรมการสอบปริญญานิพนธ์

.....ประธานกรรมการ
(พลอากาศตรี ผศ.ดร.พาห์รณ สงวนโภคัย)

.....กรรมการ
(อาจารย์เอก บำรุงศรี)

.....อาจารย์ที่ปรึกษา
(อาจารย์จรรยา แหยมเจริญ)

หัวข้อปริญญานิพนธ์ เครื่องมือสำหรับตรวจสอบการตั้งค่าตามข้อกำหนดความปลอดภัย

หน่วยกิตของปริญญานิพนธ์ 3 หน่วยกิต

รายชื่อผู้จัดทำ นายณัฐวิทย์ ลีลาวรรณศิลป 5804800059

อาจารย์ที่ปรึกษา อาจารย์จรรยา แหยมเจริญ

ระดับการศึกษา วิทยาศาสตรบัณฑิต

ภาควิชา วิทยาการคอมพิวเตอร์

ปีการศึกษา 2561

บทคัดย่อ

เนื่องด้วยตามมาตรฐาน ธ.ป.ท (ธนาคารแห่งประเทศไทย) เกี่ยวกับความปลอดภัยบนระบบคอมพิวเตอร์ล่าสุดที่พึ่งออกมาซึ่งว่าด้วยทุกสถาบันการเงินที่อยู่ภายใต้การควบคุมดูแลของธนาคารแห่งประเทศไทยจะต้องมีการตรวจสอบระบบคอมพิวเตอร์ทั้งหมดให้เป็นไปตามข้อกำหนดความปลอดภัยของสถาบันการเงินนั้น ๆ โดยจะต้องมีการตรวจสอบทุก ๆไตรมาสและตรวจสอบเครื่องคอมพิวเตอร์ทั้งหมดไม่ว่าจะเป็นของเดิมหรือของใหม่ ซึ่งทำให้สถาบันการเงินหลายแห่งได้รับผลกระทบโดยเฉพาะสถาบันการเงินที่มีระบบคอมพิวเตอร์ขนาดใหญ่และมีจำนวนเครื่องที่ต้องตรวจสอบมาก ทำให้การตรวจสอบเครื่องคอมพิวเตอร์ทั้งหมดนั้นเป็นงานที่ใช้เวลานานและต้องใช้ทรัพยากรณ์หรือบุคลากรเป็นจำนวนมาก เพื่อแก้ปัญหาดังกล่าวผู้จัดทำจึงได้เลือกใช้เครื่องมือและเทคโนโลยีในปัจจุบันเข้ามาช่วยให้การตรวจสอบตามข้อกำหนดความปลอดภัยเป็นไปอย่างอัตโนมัติและมีความถูกต้องและเพื่อช่วยลดระยะเวลาและบุคลากรในการตรวจสอบระบบคอมพิวเตอร์ลงได้

คำสำคัญ: ความปลอดภัยบนระบบคอมพิวเตอร์, ข้อกำหนดความปลอดภัย, การตรวจสอบระบบคอมพิวเตอร์

Project Title	Security Configuration Compliance Audit Tool
Project Credits	3 Units
Candidate	Mr. Nutthawit Lilawannasin 5804800059
Advisor	Miss Janya Yamcharoen
Program	Bachelor of Science
Field of Study	Computer Science
B.E.	2561

Abstract

Information communication technology and computers are an important tool for driving organizations to operate effectively, especially with security systems. Financial institutions, under the supervision of the Bank of Thailand, must check the security configurations according to the standards and report the audit results to the BOT every quarter. Each financial institution has various computer equipment that has to be an audited. At present, the auditing process by the administrator that takes a long time and mistakes occur easily. Therefore, the developer has developed a security configuration compliance audit tool with an application of container technology, a virtual machine and git help automate operations, reduce time, and reduce human error. In system development, the use of Ansible, which is an open source software, to be an intermediary between the user and the backend system and scripting was done with YAML. This tool supports Linux operating systems and can check the standards specified by 50 items. The developer had tested this tool on a virtual simulation system receiving the results as specified.

Keywords: computer system security, security compliance, system audits

Approved by .....

Approved by .....

กิตติกรรมประกาศ

(Acknowledgment)

การจัดทำปริญญานิพนธ์ฉบับนี้สำเร็จได้นั้น ผู้จัดทำได้รับความกรุณาจาก อาจารย์ผู้สอนทุกท่านที่ให้ข้อมูลต่างๆ ส่งผลให้ผู้จัดทำได้รับความรู้และประสบการณ์ต่างๆ ที่มีค่ามากมาย สำหรับปริญญานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยดีจากความร่วมมือและสนับสนุนจาก อาจารย์จรรยา แหยมเจริญ อาจารย์ที่ปรึกษาและบริษัท ออพซ์ตา (ประเทศไทย) จำกัด

ผู้จัดทำใคร่ ขอขอบพระคุณคณะกรรมการสอบปริญญานิพนธ์ ที่ได้ให้คำแนะนำสำคัญในการสอบปริญญานิพนธ์ฉบับนี้ และผู้มีส่วนร่วมทุกท่านที่ไม่ได้กล่าวนาม ที่มีส่วนร่วมในการให้ข้อมูลให้ความช่วยเหลือ และเป็นที่ปรึกษาให้คำแนะนำต่าง ๆ จนทำให้งานทุกอย่างประสบความสำเร็จไปด้วยดี ซึ่งผู้จัดทำขอขอบพระคุณเป็นอย่างสูงไว้ ณ ที่นี้ด้วย

ผู้จัดทำ

นายณัฐวิทย์ ลีลาวรรณศิลป

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

สารบัญรูปภาพ (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

เนื่องด้วยระบบสารสนเทศและคอมพิวเตอร์ (ICT) เป็นเครื่องมือช่วยในการขับเคลื่อนองค์กรให้มีการดำเนินงานที่มีประสิทธิภาพและน่าเชื่อถือ โดยอย่างยิ่งระบบรักษาความปลอดภัย (Security System) ซึ่งสถาบันการเงินทุกสถาบันจะต้องมีการตรวจสอบระบบและอุปกรณ์คอมพิวเตอร์ของตนให้เป็นไปตามข้อความกำหนดความปลอดภัยที่ธนาคารแห่งประเทศไทยได้กำหนดไว้ ซึ่งมาตรฐานในการตรวจสอบทางสถาบันการเงินสามารถกำหนดมาตรฐานในการตรวจสอบได้เอง ในสถาบันการเงินแต่ละแห่งมีจำนวนอุปกรณ์คอมพิวเตอร์เป็นจำนวนมาก และมีการใช้แพลทฟอร์มที่หลากหลาย ที่จะต้องได้รับการตรวจสอบอย่างสม่ำเสมอทุก ๆ ไตรมาศ โดยเดิมเป็นการตรวจสอบ (Audit) โดยผู้ดูแลระบบ (Administrator) ที่สามารถกระทำได้ทีละเครื่องหรือทีละอุปกรณ์ จึงทำให้ต้องใช้เวลาในการดำเนินการนาน และอาจจะเกิดข้อผิดพลาดได้ง่าย

ดังนั้นผู้จัดทำปริญญานิพนธ์จึงได้พัฒนาเครื่องมือเพื่อช่วยผู้ดูแลระบบของสถาบันการเงินในการตรวจสอบความปลอดภัยของระบบคอมพิวเตอร์ที่มีอยู่เป็นจำนวนมาก ให้มีความถูกต้องรวดเร็ว และมีประสิทธิภาพมากยิ่งขึ้น โดยเลือกใช้เครื่องมือที่เป็นโอเพ่นซอร์ส (Open Source Tool) ได้แก่ Ansible และประยุกต์ใช้เทคโนโลยีเวอชวลแมชชีน (Virtual Machine) คอนเทนเนอร์ (Container) และเทคโนโลยีกิท (Git) เข้ามาช่วย เพื่อให้การทำงานเป็นโดยอัตโนมัติและประมวลผลได้พร้อมกันครั้งละหลายๆ เครื่อง โดยเครื่องมือที่ผู้จัดทำพัฒนาขึ้นมานี้รองรับแพลทฟอร์มของลีนุกซ์ Red Hat Enterprise Linux (RHEL) และรองรับมาตรฐานทั้งหมด 50 ข้อ เมื่อนำไปทดสอบกับระบบจำลอง (Virtual System) พบว่าสามารถตรวจสอบได้รวดเร็ว และให้ผลลัพธ์ที่ถูกต้อง สามารถออกรายงานสรุปในรูปแบบของ CSV ได้

1.2 วัตถุประสงค์

เพื่อพัฒนาเครื่องมือสำหรับตรวจสอบการตั้งค่าตามข้อกำหนดความปลอดภัย

1.3 ขอบเขตของปริญญานิพนธ์

1.3.1 เป็นระบบทดสอบเพื่อแสดงประสิทธิภาพการทำงานของเครื่องมือ

1.3.2 พัฒนาสคริปต์ในการตรวจสอบข้อกำหนดความปลอดภัยด้วยภาษา YAML

1.3.3 พัฒนาโดยใช้เทคโนโลยี Container และ Git

1.3.4 พัฒนาส่วนติดต่อกับผู้ใช้ (User Interface) ด้วย Ansible AWX

1.3.5 จำนวนข้อที่จะตรวจสอบทั้งหมด 50 ข้อ ตามที่สถาบันการเงินกำหนด

1.3.6 แพลทฟอร์มที่ตรวจสอบ คือ RHEL 7 (Red Hat Enterprise Linux 7)

1.3.7 สามารถออกรายงานแสดงผลการทดสอบในรูปแบบ CSV ไฟล์ได้

1.4 ขั้นตอนและวิธีการดำเนินงาน

1.4.1 ศึกษาและรวบรวมข้อมูล (Detail Study)

รวบรวมความต้องการโดยศึกษาจากผู้ใช้งานที่มีประสบการณ์จากสถาบันการเงินแห่งหนึ่ง ประกอบไปด้วยข้อกำหนดความปลอดภัย รูปแบบของเอกสารแสดงผลการทดสอบรวมถึงระบบปฏิบัติการที่ต้องการจะทดสอบ

1.4.2 วิเคราะห์ระบบและออกแบบระบบ (System Analysis and Design)

เริ่มวิเคราะห์และออกแบบสถาปัตยกรรมของระบบโดยที่ระบบจะต้องมีขนาดไม่ใหญ่เกินไปสามารถติดตั้งได้บนเครื่องเครื่องเดียวและใช้ซอฟต์แวร์ที่เป็นโอเพ่นซอร์สเพื่อลดต้นทุนของระบบ

1.4.3 พัฒนาระบบ (System Development)

ในขั้นตอนการพัฒนาระบบจะเริ่มจากการติดตั้งซอฟร์แวร์ตามที่ได้ออกแบบไว้และจำลองระบบที่มีระบบปฏิบัติการเป็น RHEL เวอร์ชั่น 7 สำหรับการทดสอบเมื่อเสร็จแล้วจึงเริ่มเขียน Ansible roles สำหรับตรวจสอบตามข้อกำหนดความปลอดภัย เพิ่มในส่วนออกรายงานด้วย python ซึ่งทำให้ผู้ใช้งานนำไปใช้ได้ง่าย

1.4.4 ทดสอบระบบ (System Testing)

ในขั้นตอนการทดสอบระบบจะเริ่มจากการทดสอบบนระบบที่ผู้พัฒนาได้เตรียมไว้ก่อนโดยทำการทดสอบเทียบกับข้อกำหนดความปลอดภัยแบบข้อต่อข้อว่าได้ผลตรงกันหรือไม่และทำการตรวจสอบรูปแบบการออกรายงานการแสดงผลทั้งหมดว่าถูกต้องตามรูปแบบหรือไม่

1.4.5 จัดทำเอกสารประกอบปริญญานิพนธ์ (Documentation)

จัดทำเอกสารประกอบปริญญานิพนธ์ แนวทางในการจัดทำภาคนิพธ์วิธีการและขั้นตอนการดำเนินปริญญานิพนธ์เพื่อแสดงรายละเอียดการพัฒนาเครื่องมือและเป็นคู่มือในการใช้งานเครื่องมืออีกทั้งยังเป็นเอกสารสำหรับการนำเครื่องมือไปพัฒนำต่อในอนาคต

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ช่วยเพิ่มผลผลิตในการตรวจสอบระบบ

1.5.2 ช่วยเพิ่มประสิทธิภาพในการตรวจสอบระบบ

1.5.3 ช่วยเพิ่มความพร้อมใช้งานในการตรวจสอบระบบ

1.5.4 ช่วยเพิ่มความน่าเชื่อถือในการตรวจสอบระบบ

1.5.5 ช่วยลดต้นทุนในการตรวจสอบระบบ

1.6 แผนและระยะเวลาดำเนินงานปริญญานิพนธ์

ตางราง 1.1 แผนและระยะเวลาดำเนินงานปริญญานิพนธ์

ขั้นตอนการดำเนินงาน	2561								2562															
	พฤศจิกายน				ธันวาคม				มกราคม				กุมภาพันธ์				มีนาคม				เมษายน			
	ส1	ส2	ส3	ส4	ส1	ส2	ส3	ส4	ส1	ส2	ส3	ส4	ส1	ส2	ส3	ส4	ส1	ส2	ส3	ส4	ส1	ส2	ส3	ส4
ศึกษาและรวบรวมข้อมูล																								
วิเคราะห์ระบบและออกแบบระบบ																								
พัฒนาระบบ																								
ทดสอบระบบ																								
จัดทำเอกสารประกอบปริญญานิพนธ์																								

1.7 อุปกรณ์และเครื่องมือที่ใช้ในการพัฒนา

1.7.1 ฮาร์แวร์

1.7.1.2 เครื่องแม่ข่ายจำลองสำหรับติดตั้ง Ansible AWX : CPU 4 Cores / RAM 8 GB / Disk 20 GB

1.7.1.2 เครื่องแม่ข่ายจำลองสำหรับทดสอบ : CPU 4 Cores / RAM 8 GB / Disk 20 GB

1.7.2 ซอฟต์แวร์

1.7.2.1 ระบบปฏิบัติการ Ubuntu 16.04 (Ubuntu Linux version 16.04 LTS)

1.7.2.2 ระบบปฏิบัติการ RHEL 7 (Red Hat Enterprise Linux version 7)

1.7.2.3 โปรแกรม docker

1.7.2.4 โปรแกรม docker-compose

1.7.2.5 ไฟล์ awx_web docker images

1.7.2.6 ไฟล์ awx_task docker images

1.7.2.7 ไฟล์ awx_rabbitmq docker images

1.7.2.8 ไฟล์ memcache docker images

1.7.2.9 โปรแกรม postgresql 10

1.7.2.10 โปรแกรม python 2.7

บทที่ 2
การทบทวนวรรณกรรมที่เกี่ยวข้อง

ผู้จัดทำได้ทำการเก็บรวบรวมข้อมูลโดยการศึกษาจากเทคโนโลยีในปัจจุบันเกี่ยวกับเทคโนโลยีในเรื่องระบบปฏิบัติการ (Operating System) เทคโนโลยีคอนเทนเนอร์ (Container) และเครื่องมือการจัดการการคอนฟิกกูเรชั่นแบบศูนย์กลาง (Configuration Management Tools) โดยนำมาวิเคราะห์ข้อดีข้อเสียของซอฟต์แวร์ต่าง ๆ จากนั้นจึงนำมาออกแบบระบบและพัฒนาระบบ เพื่อใช้งานรวมถึงนำข้อมูลที่ได้มาใช้ในการจัดทำปริญญานิพนธ์

2.1 Red Hat Enterprise Linux (RHEL)[1]

Red Hat เปิดตัวระบบปฏิบัติการลีนุกซ์ (Linux) รุ่นแรกเมื่อเดือนพฤษภาคมปี คศ. 1995 ซึ่งเป็นลีนุกซ์ระบบแรกที่ใช้ระบบการจัดการแพคเกจ (Package Manager) ที่ชื่อว่า RPM (Redhat Packages Manager) ซึ่งต่อมาเป็นต้นกำเนิดของลีนุกซ์ที่ได้รับความนิยมรุ่นต่างๆ ตามมา เช่น Mandriva Linux

ในปี 2003 Red Hat หยุดการพัฒนา Red Hat Linux และเปลี่ยนมาเป็น Red Hat Enterprise Linux (RHEL) โดยมีจุดมุ่งหมายเพื่อรองรับกลุ่มลูกค้าองกรณ์ขนาดใหญ่ และเริ่มโครงการฟีโดรา (Fedora Linux) สำหรับเป็นทางเลือกแก่ผู้ใช้งานทั่วไปซึ่งไม่มีการเก็บค่าใช้จ่ายในการใช้งานแต่

จากที่กล่าวมาผู้จัดทำได้เล็งเห็นว่าระบบปฏิบัติการ RHEL มีประวัติมาอย่างยาวนานและสามารถรองรับกับองกรณ์ขนาดใหญ่ได้เป็นอย่างดี จึงได้นำมาเป็นระบบปฏิบัติการตัวอย่างที่จะใช้ตรวจสอบในระบบนี้

รูปที่ 2.1 ระบบปฏิบัติการ Linux Red Hat

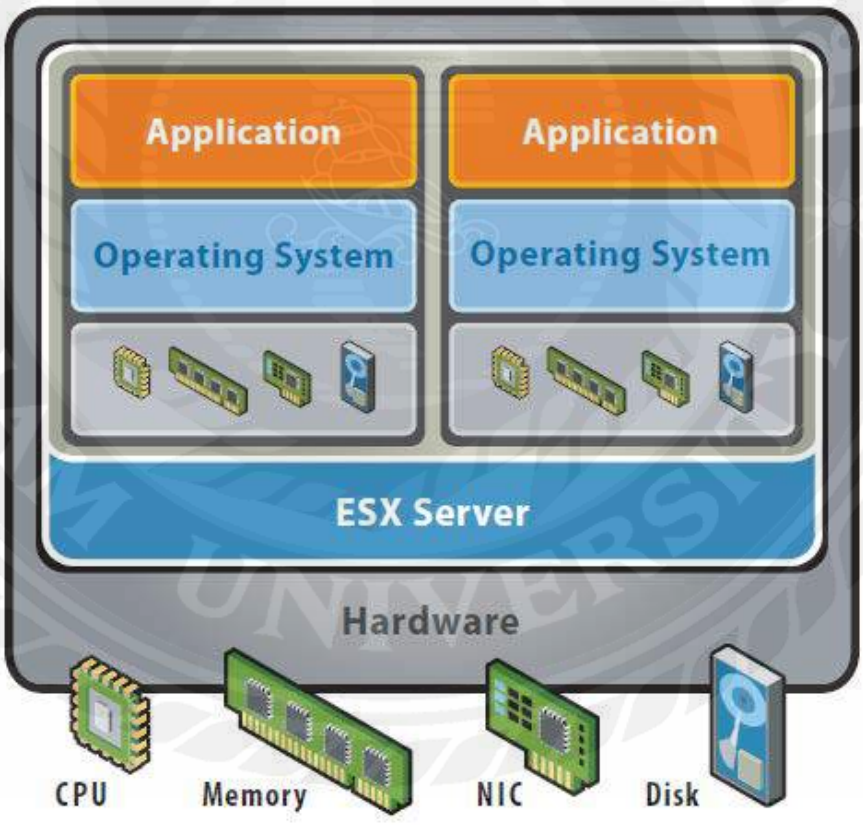
[1]อ้างอิง https://en.wikipedia.org/wiki/Red_Hat

2.2 Virtual Machine (VM)[2]

ต้นกำเนิดของเทคโนโลยีเวอร์ชวล แมชชีน (Virtual Machine: VM) นั้นมาจากการใช้งานเครื่องแม่ข่าย (Server) ในปัจจุบันที่มีขนาดใหญ่ได้ไม่คุ้มค่ามากพอ จึงได้มีการนำเครื่องแม่ข่ายนั้นมาแบ่งเป็นเครื่องย่อยๆลงมาให้มีขนาดพอเหมาะกับแอปพลิเคชันที่ทำงานอยู่จริง โดยการใช้เทคโนโลยี Hypervisor เข้ามาช่วยในการทำงาน

Hypervisor เป็นซอฟต์แวร์ที่ทำงานอยู่ใน เคอร์เนล (Kernel) ของระบบปฏิบัติการ โดย Hypervisor จะทำงานอยู่บนเครื่องแม่ข่ายจริงๆ ซึ่งเครื่องแม่ข่ายที่มี Hypervisor ทำงานอยู่จะถูกเรียกว่า Host machine ซึ่ง Guest machine จะทำงานอยู่บน Host machine โดยมี Hypervisor เป็นตัวจำลองทรัพยากรณ์ที่จำเป็นได้แก่ ซีพียู (CPU), หน่วยความจำ (RAM), พื้นที่ (Disk) และ เครือข่าย (Network) ซึ่งแบ่งทรัพยากรมาจาก Host machine

เพื่อรองรับกับเทคโนโลยีในปัจจุบันที่องกรณ์ต่างๆ นำเทคโนโลยี Virtual Machine เข้ามาใช้เป็นพื้นฐานในองกรณ์อยู่แล้ว ผู้จัดทำจึงได้ใช้ Guest machine เป็นเครื่องแม่ขายแทนเครื่องแม่ข่ายแทนเครื่องแม่ข่ายจริงในการติดตั้งโปรแกรมต่าง ๆ ที่จำเป็นต้องใช้ในโปรเจคนี้



รูปที่ 2.2 สถาปัตยกรรมเทคโนโลยี Virtual Machine (VM)

[2]อ้างอิง https://en.wikipedia.org/wiki/Virtual_machine

2.3 Docker Container[3]

เทคโนโลยีคอนเทนเนอร์ (Container) ในปัจจุบันคือ Linux container (LXC) ซึ่งถูกพัฒนาขึ้นในปี 2008 โดยใช้เทคโนโลยี cgroups และ Linux namespaces ใน Linux kernel

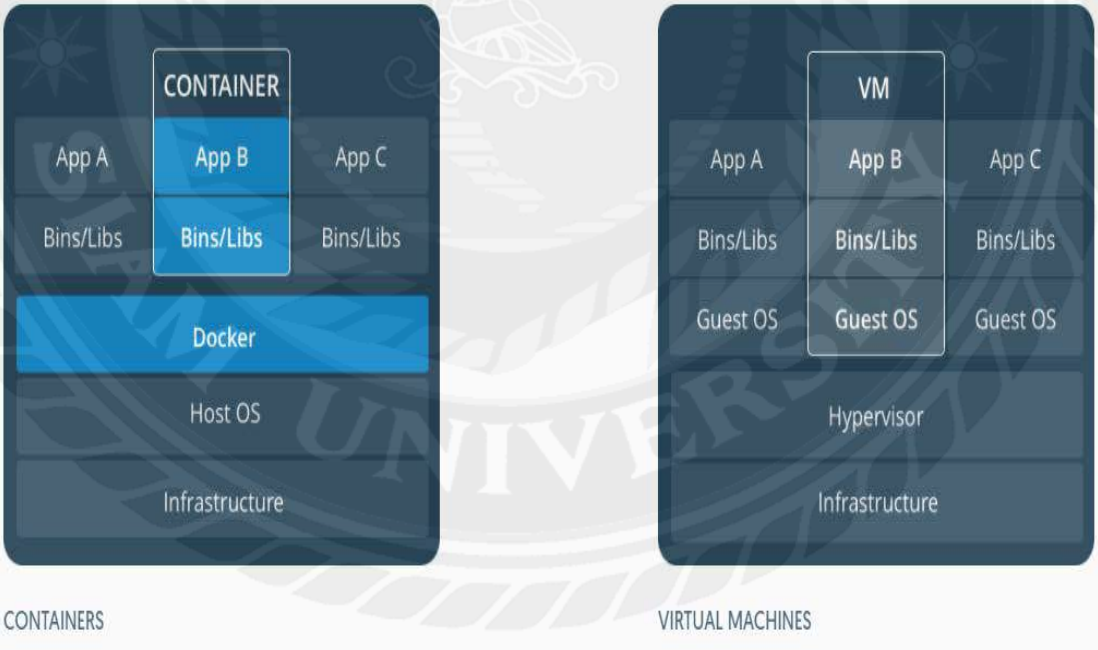
แนวคิดของเทคโนโลยี Container คือการแบ่งทรัพยากรมาจากระบบปฏิบัติการโดยตรงที่เรียกว่า user space โดยมีจุดประสงค์เช่นเดียวกับ Virtual Machine คือการแบ่งพื้นที่และทรัพยากรแบบเป็นส่วนตัวเพื่อมาใช้ในการประมวลผลงาน ข้อแตกต่างที่สำคัญระหว่าง Virtual Machine กับ Container คือ

- Container แบ่ง Host-system kernel กับ Container อื่นๆ

- Virtual Machine จำลอง Hardware จาก Host-system และมี kernel สำหรับ Guest machine ของแต่ละเครื่อง

Docker เป็นโอเพ่นซอร์สโปรเจ็ค (Open-source project) ที่ได้รับความนิยมและมีการใช้งานอย่างแพร่หลาย ซึ่งพัฒนาต่อยอดมากจากเทคโนโลยี LXC ให้ใช้งานได้ง่ายขึ้นพร้อมกับมีเครื่องมือในการอำนวยความสะดวกต่างๆ แก่ผู้ใช้งาน

เนื่องจากในโปรเจ็คมีแอปพลิเคชันที่จำเป็นอยู่หลายส่วนที่ต้องนำมาประกอบกันเพื่อให้ง่ายต่อการจัดการและดูแล ทางผู้จัดทำจึงเลือกใช้เทคโนโลยีคอนเทนเนอร์ในการแบ่งทรัพยากรของเครื่องสำหรับแต่ละแอปพลิเคชัน



รูปที่ 2.3 เปรียบเทียบระหว่างเทคโนโลยี Container กับ Virtual Machine

[3]อ้างอิง https://en.wikipedia.org/wiki/Docker_(software)

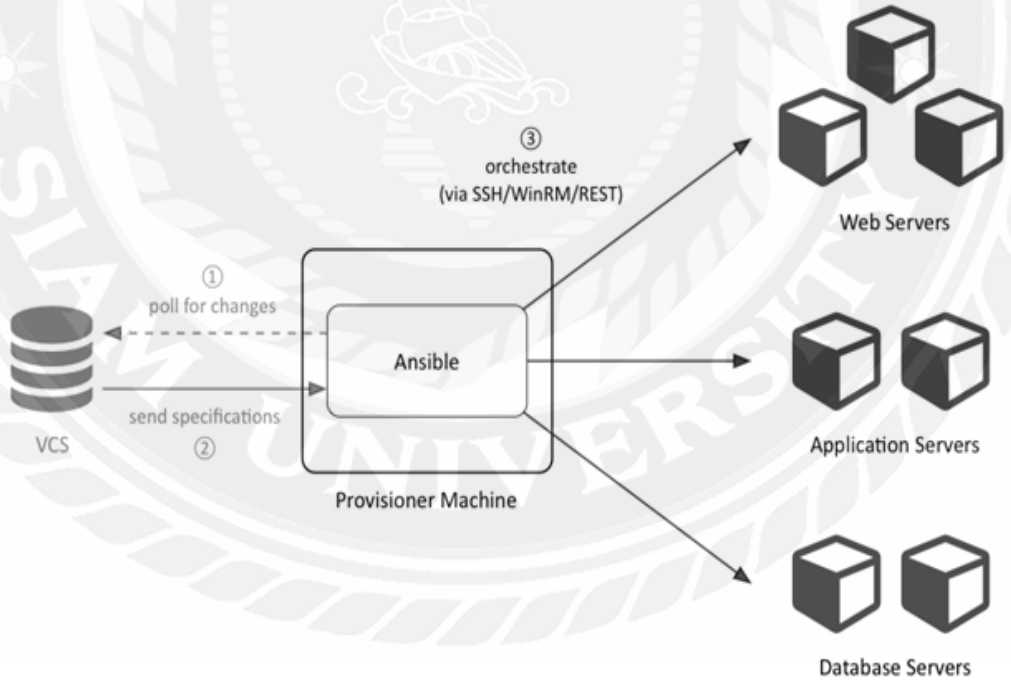
2.4 Ansible AWX[4]

ในปัจจุบันที่ระบบสารสนเทศมีขนาดใหญ่ขึ้นทำให้ผู้ดูแลระบบต้องดูแลเครื่องแม่ข่าย ในระบบที่มีมากขึ้น การที่จะติดตั้งหรืออัพเกรดซอฟต์แวร์ในระบบจึงเป็นงานที่ต้องใช้เวลามาก เพื่อรองรับการทำงานในรูปแบบนี้จึงจำเป็นต้องมีเครื่องมือเพื่อเข้ามาช่วยเหลือในการทำงาน หนึ่งในนั้นคือ Ansible

Ansible เป็นเครื่องมือสำหรับการจัดการกำหนดค่าต่างๆ (Configuration) จากศูนย์กลาง ซึ่งหมายถึง เครื่องมือสำหรับการติดตั้ง อัพเกรดและกำหนดค่าภายในระบบแบบอัตโนมัติพร้อมๆ กัน โดยผู้ใช้งานจะต้องเขียนรายการงานที่จะให้ Ansible ทำ ซึ่งจะอยู่ในรูปแบบภาษา YAML เพื่อให้ Ansible ทำงานตามที่ต้องการ โดยปกติ Ansible จะทำงานในรูปแบบของ Agent less และไม่มีส่วนติดต่อผู้ใช้งานแบบกราฟฟิก (Graphical User Interface) ให้ใช้งาน

Ansible AWX เกิดขึ้นเพื่อเพิ่มความสะดวกในการใช้งาน Ansible โดยการพัฒนาในรูปแบบของเว็บ (Web Base GUI) และเพิ่มความสามารถในการเก็บข้อมูลการทำงานลงฐานข้อมูล ความสามารถในการยืนยันตัวตนของผู้ใช้งานและอื่น ๆ

จากที่ได้กล่าวมาผู้จัดทำจึงเล็งเห็นถึงข้อดีข้อเสียต่างๆ และเพื่อพัฒนาระบบให้สะดวกแก่การใช้งานจึงได้นำ Ansible AWX ร่วมกับ Ansible มาใช้ในการพัฒนาระบบนี้



รูปที่ 2.4 สถาปัตยกรรมการการทำงานของ Ansible

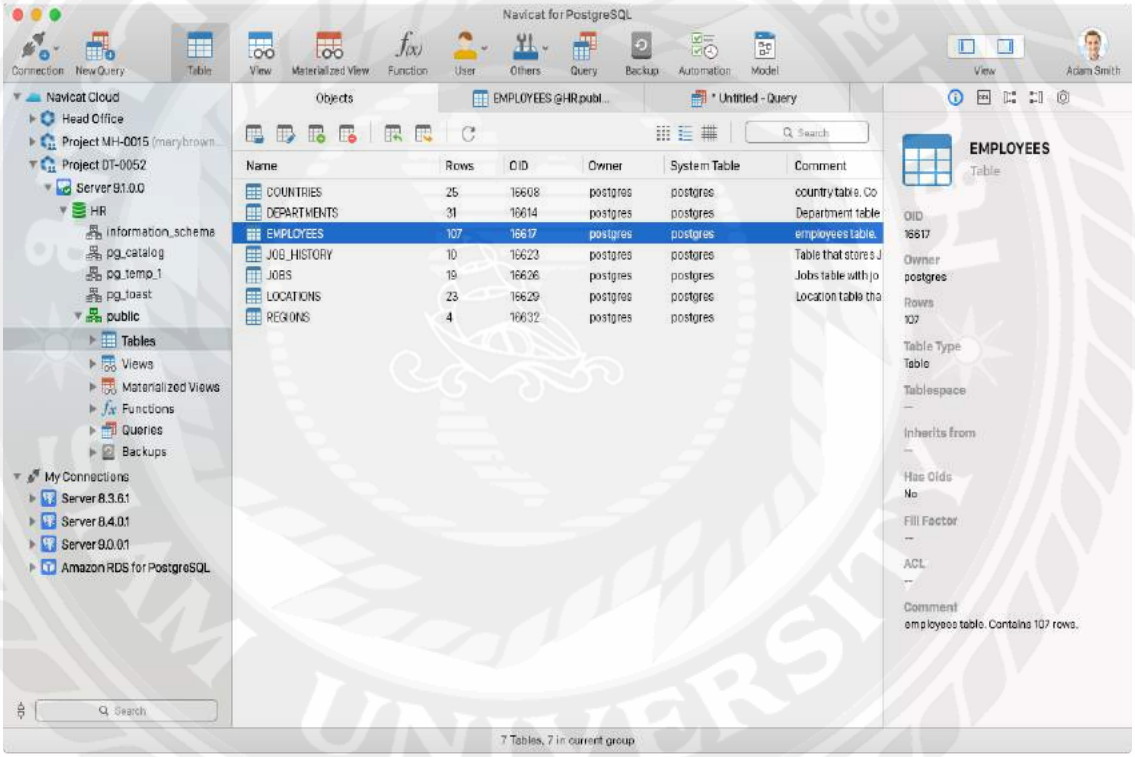
[4]อ้างอิง https://en.wikipedia.org/wiki/Ansible_(software)

2.5 PostgreSQL[5]

PostgreSQL เป็นหนึ่งในฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management System (RDBMS)) ที่ได้รับความนิยมตัวหนึ่ง โดยมีความสามารถในการจัดการจำนวนข้อมูลตั้งแต่ระบบขนาดเล็กไปจนระบบขนาดใหญ่ เช่น Data Warehouse

PostgreSQL ถูกพัฒนาโดยมหาวิทยาลัย California, Berkeley ในปี 1982 และได้รับรางวัล Turing Award ในปี 2014 รวมทั้งยังเป็นฐานข้อมูลที่ถูกติดตั้งมาโดยเริ่มต้นในระบบปฏิบัติการ Linux, FreeBSD รวมถึงยังรองรับการติดตั้งบนระบบปฏิบัติการวินโดว์ด้วย

เป็นข้อบังคับและข้อจำกัดอย่างหนึ่งในการติดตั้งซอฟต์แวร์ Ansible AWX ที่จะต้องมีการติดตั้งซอฟต์แวร์ฐานข้อมูลสำหรับเก็บข้อมูลผู้ใช้รวมถึงข้อมูลการทำงาน ด้วยฐานข้อมูล PostgreSQL



รูปที่ 2.5 ฐานข้อมูล PostgreSQL

[5]อ้างอิง https://en.wikipedia.org/wiki/PostgreSQL

2.6 Security Audit[6]

Security Audit หมายถึง การประเมินผลความปลอดภัยของระบบเทคโนโลยีสารสนเทศภายในบริษัท โดยจะตรวจสอบตามเกณฑ์หรือมาตรฐานที่กำหนดเอาไว้ ซึ่งการตรวจสอบโดยทั่วไปจะครอบคลุมในเรื่องของความปลอดภัยทางกายภาพและสภาพแวดล้อม ความปลอดภัยทางซอฟต์แวร์และการจำกัดการเข้าถึงระบบจากผู้ใช้งาน เป็นต้น โดยส่วนมากการทำ Security Audit นั้นจะยึดจากมาตรฐานดังนี้ HIPPA, Sarbanes-Oxley Act, California Security Breach Information Act หรืออาจเป็นมาตรฐานที่องค์กรณ์นั้นๆจัดทำขึ้นมาโดยเฉพาะ

รูปที่ 2.6 Security Audit

2.7 เปรียบเทียบเครื่องมือที่ใช้ในการพัฒนา

ปัจจุบันซอฟต์แวร์ที่ใช้สำหรับการทำระบบอัตโนมัติ (Automate System) มีอยู่หลายระบบด้วยกัน โดยแบ่งเป็นซอฟต์แวร์ในกลุ่มที่ต้องติดตั้งโปรแกรมบนเครื่องลูกข่าย (Agent-base) และกลุ่มที่ไม่ต้องติดตั้งโปรแกรมบนเครื่องลูกข่าย (Agent-less) โดยในกลุ่มแรกคือ กลุ่มที่ต้องติดตั้งโปรแกรมบนเครื่องลูกข่าย ซอฟต์แวร์ที่ได้รับความนิยมได้แก่ Chelf และ Puppet ซึ่งเป็นซอฟต์แวร์ที่มีมานานและถูกนำไปในในองค์กรขนาดใหญ่หลายองค์กร โดยในรูปแบบการทำงานของซอฟตแวร์แบบนี้คือ ผู้ดูแลระบบจะต้องติดตั้งซอฟต์แวร์บนเครื่องแม่ข่ายและติดตั้งซอฟต์แวร์บนเครื่องลูกข่าย เพื่อให้เครื่องแม่ข่ายสามารถเข้าไปสั่งงานได้ ซึ่งในการทำงานรูปแบบนี้มีข้อเสียหลักๆ คือ ผู้ดูแลระบบจะต้องติดตั้งซอฟต์แวร์บนเครื่องลูกข่ายทุกเครื่องและเมื่อมีการอัพเกรดซอฟต์แวร์บนเครื่องลูกข่าย ผู้ดูแลระบบก็จะต้องตามอัพเกรดซอฟต์แวร์บนเครื่องลูกข่ายทุกเครื่องเช่นกัน ในกลุ่มต่อมาคือ กลุ่มที่ไม่ต้องติดตั้งซอฟต์แวร์บนเครื่องลูกข่ายซอฟต์แวร์ที่ได้รับความนิยมที่สุดได้แก่ Ansible ซึ่งในปัจจุบันกำลังเป็นที่นิยมและถูกนำมาใช้ในองค์กรขนาดใหญ่หลายองค์กร โดยข้อดีหลักๆ ของซอฟต์แวร์แบบนี้คือ มาทดแทนข้อเสียในส่วนของซอฟต์แวร์แบบที่ต้องติดตั้งซอฟต์แวร์บนเครื่องลูกข่าย ซึ่งช่วยลดภาระงานของผู้ดูแลระบบลงได้เป็นอย่างดี

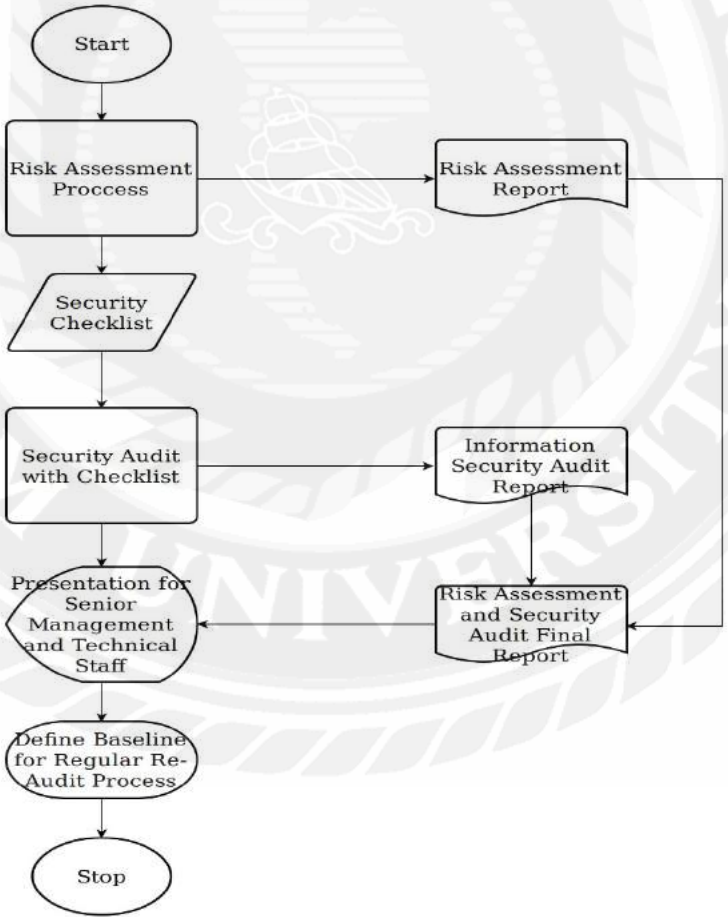
[6]อ้างอิง https://searchcio.techtarget.com/definition/security-audit

บทที่ 3
การวิเคราะห์และออกแบบระบบ

3.1 วิเคราะห์ระบบงานเดิม (As - Is System Analysis)

ในขั้นตอนการตรวจสอบตามมาตรฐานความปลอดภัยนั้นเริ่มจากการประเมินความเสี่ยงที่จะเกิดขึ้นต่อระบบหลังจากนั้นทีมวิเคราะห์ความเสี่ยงจะออกเอกสารความเสี่ยงที่ได้หลังจากการวิเคราะห์และเอกสารรายการการตรวจสอบระบบเพื่อใช้เป็นมาตรฐานในการตรวจสอบจากนั้นฝ่ายตรวจสอบระบบ (Security Audit) จะนำเอกสารมาตรการตรวจสอบเพื่อไปตรวจสอบกับเครื่องแม่ข่ายต่างๆ ภายในระบบโดยจะใช้ เวลามากหรือน้อยขึ้นอยู่กับจำนวนเครื่องแม่ข่ายที่ต้องตรวจสอบ หลังจากนั้นจึงออกรายงานการตรวจสอบว่ามีเครื่องแม่ข่ายเครื่องใดผ่านและไม่ผ่านโดยระบุรายละเอียดว่าผ่านข้อใดและไม่ผ่านข้อใดบ้างจากนั้นจึงนำเสนอต่อผู้รับผิดชอบและทำการกำหนดแผนการตรวจสอบครั้งต่อไป

3.1.1 ขั้นตอนการตรวจสอบข้อกำหนดความปลอดภัยของระบบงานเดิม



รูปที่ 3.1 ขั้นตอนการตรวจสอบข้อกำหนดความปลอดภัยของระบบงานเดิม

3.1.2 ปัญหาของระบบงานเดิม

3.1.2.1 ได้ผลผลิตในการตรวจสอบที่ต่ำ

3.1.2.2 ใช้เวลาในการตรวจสอบนาน

3.1.2.3 ขาดความพร้อมการใช้งานถ้าบุคคลนั้นไม่สามารถปฏิบัติงานได้

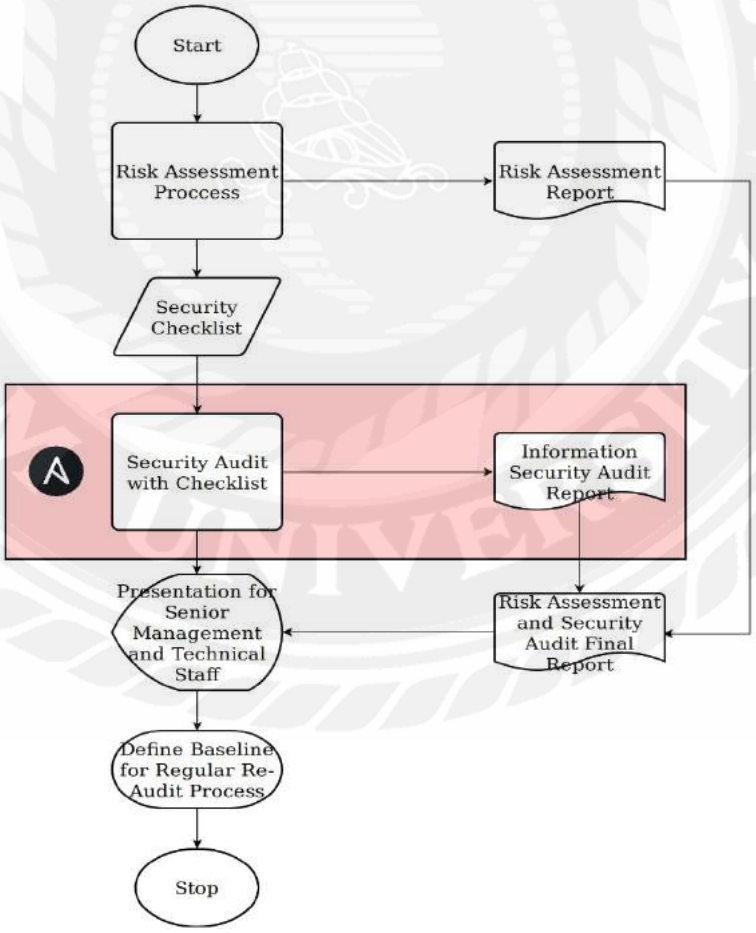
3.1.2.4 เกิดการตรวจสอบผิดพลาดได้เสมอ

3.1.2.5 ใช้ต้นทุนในการตรวจสอบสูงเนื่องจากต้องจ้างบุคลากรที่มีความสามารถเฉพาะทาง

3.2 วิเคราะห์ระบบงานใหม่ (New System Analysis)

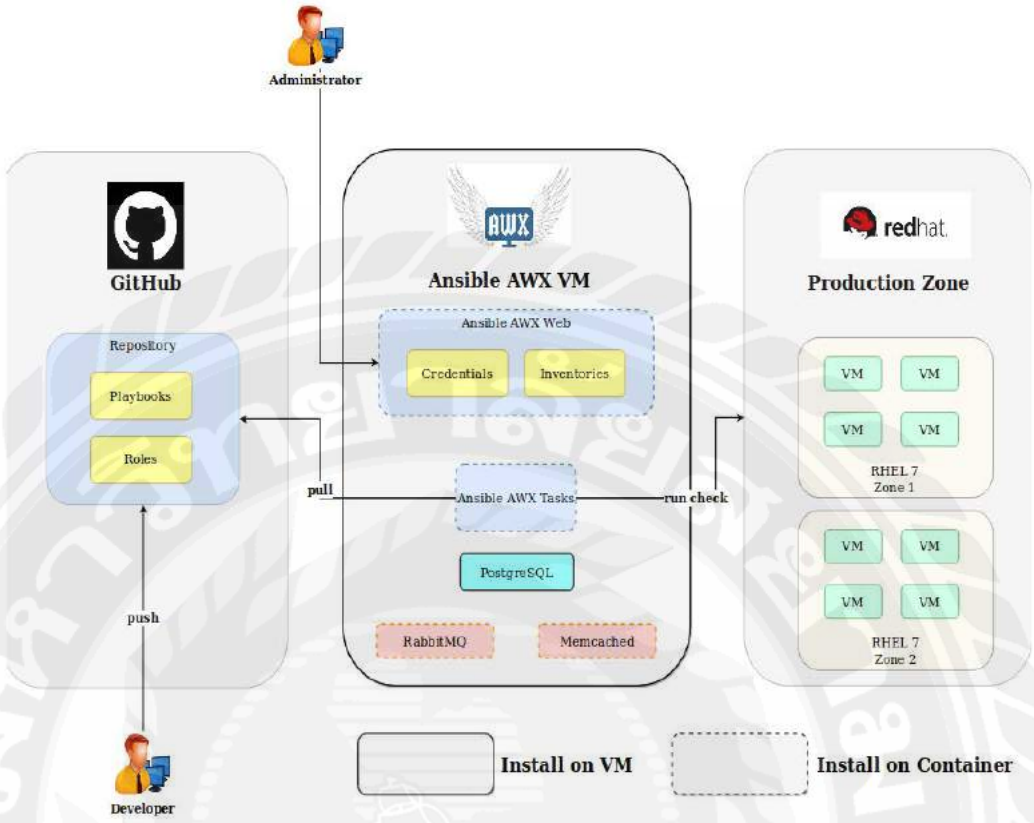
ในระบบงานใหม่นี้ผู้จัดทำได้พัฒนาสริปต์สำหรับตรวจสอบข้อกำหนดความปลอดภัยด้วยภาษ YAMLได้นำโปรแกรม Ansible เข้ามาช่วยในขั้นตอนการตรวจสอบตามมาตรฐานความปลอดภัยโดย Ansible จะทำหน้าที่ตรวจสอบระบบจากสคริปต์ที่พัฒนาขึ้นและออกแบบรายงานแบบอัตโนมัติซึ่งจะช่วยให้ลดเวลาการตรวจสอบและลดข้อผิดพลาดในการตรวจสอบลงได้

3.2.1 แสดงขั้นตอนการตรวจสอบข้อกำหนดความปลอดภัยของระบบงานใหม่



รูปที่ 3.2 ขั้นตอนการตรวจสอบข้อกำหนดความปลอดภัยของระบบงานใหม่

3.2.2 สถาปัตยกรรมของระบบงานใหม่



รูปที่ 3.3 สถาปัตยกรรมของระบบงานใหม่

จากสถาปัตยกรรมองค์ประกอบทั้งหมดของ Ansible AWX ได้แก่ Ansible AWX web, Ansible AWX task, RabbitMQ และ Memcached จะถูกติดตั้งเป็นคอนเทนเนอร์ (Container) อยู่บนเครื่องแม่ข่ายที่เป็น Virtual Machine ส่วนฐานข้อมูล PostgreSQL นั้นจะถูกติดตั้งอยู่บน Virtual Machine โดยตรง ในส่วนเครื่องแม่ข่ายที่จะตรวจสอบจะอยู่ด้านขวาของรูปซึ่งเป็นระบบปฏิบัติการ RHEL 7 (Red Hat Enterprise Linux 7) ทั้งหมด สำหรับส่วนของผู้พัฒนา (Developer) และผู้ดูแลระบบ (Administrator) ผู้พัฒนาจะต้องเขียน Ansible Playbook และ Ansible Role สำหรับสั่งงาน Ansible และทำการส่งโค้ดเข้าไปที่คอนเทนเนอร์ Ansible AWX task เพื่อให้ Ansible นำสคริปต์ประมวลผลต่อไป ส่วนผู้ดูแลระบบนั้นเพียงแค่เข้ามาใช้งานผ่านทางเว็บบราวเซอร์ที่ Ansible AWX เตรียมไว้ให้เพื่อเข้ามาสั่งงานให้ทำการตรวจสอบและดูผลทดสอบจากรายงานที่ออกมา

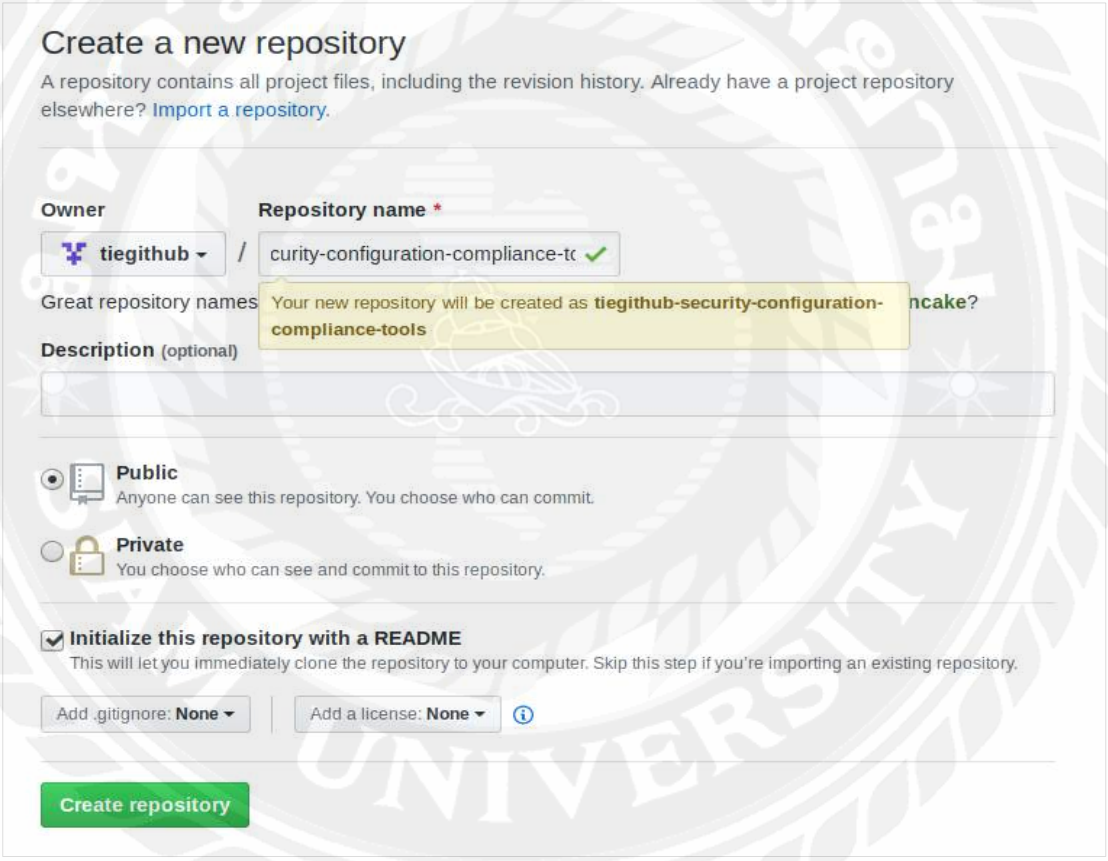
บทที่ 4
การนำไปใช้

4.1 การพัฒนาระบบ (System Development)

ในส่วนการพัฒนาระบบ ผู้จัดทำได้เลือกใช้ซอฟต์แวร์ GitHub ในการทำเวอร์ชั่นคอนโทรล (Version Control System) และใช้ภาษา YAML ในการเขียน Ansible script เพื่อทำการตรวจสอบระบบตามข้อกำหนด โดยมีขั้นตอนดังนี้

4.1.1 สร้าง Repository บน GitHub

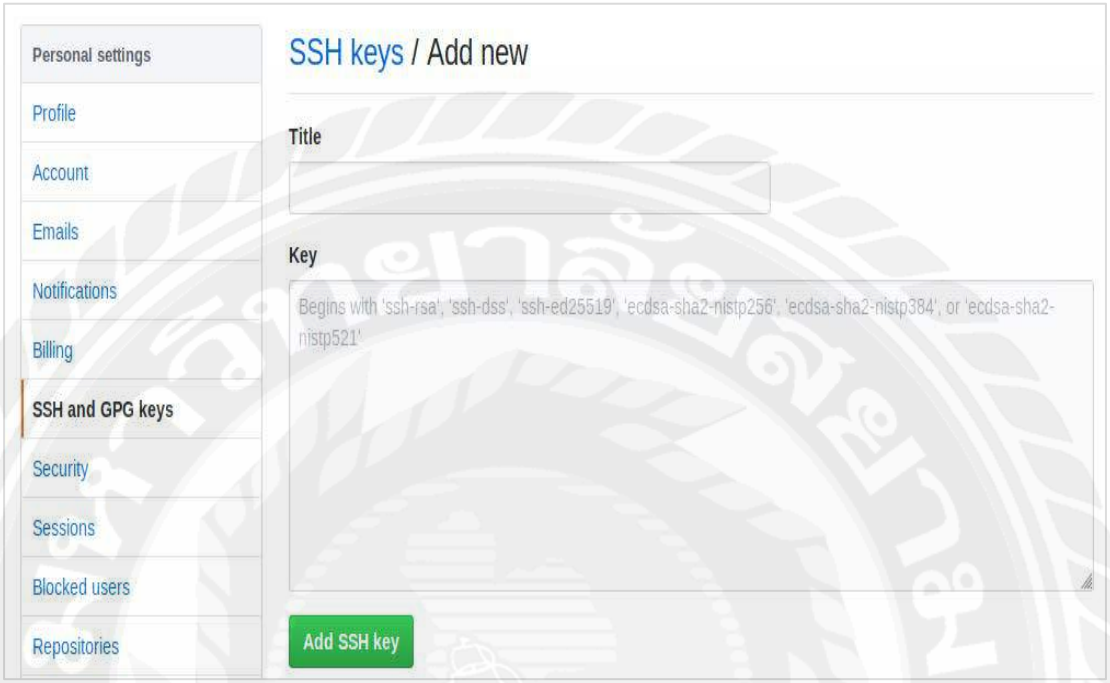
ล็อกอินเข้าเว็บไซต์ https://github.com แล้วเลือก Create a new repository แล้วกรอกข้อมูลที่จำเป็น เมื่อครบแล้วคลิก Create repository



รูปที่ 4.1 หน้าต่างสำหรับสร้าง Repository

4.1.2 สร้างใบรับรอง (Credentials)

การสร้างใบรับรองนี้ทำขึ้นเพื่อเพิ่มความสะดวกในการอ่านสคริปต์หรือส่งสคริปต์ขึ้นไปเก็บบนที่ Repository โดยไม่ต้องป้อนรหัสผ่านทุกครั้ง โดยมีขั้นตอนดังนี้



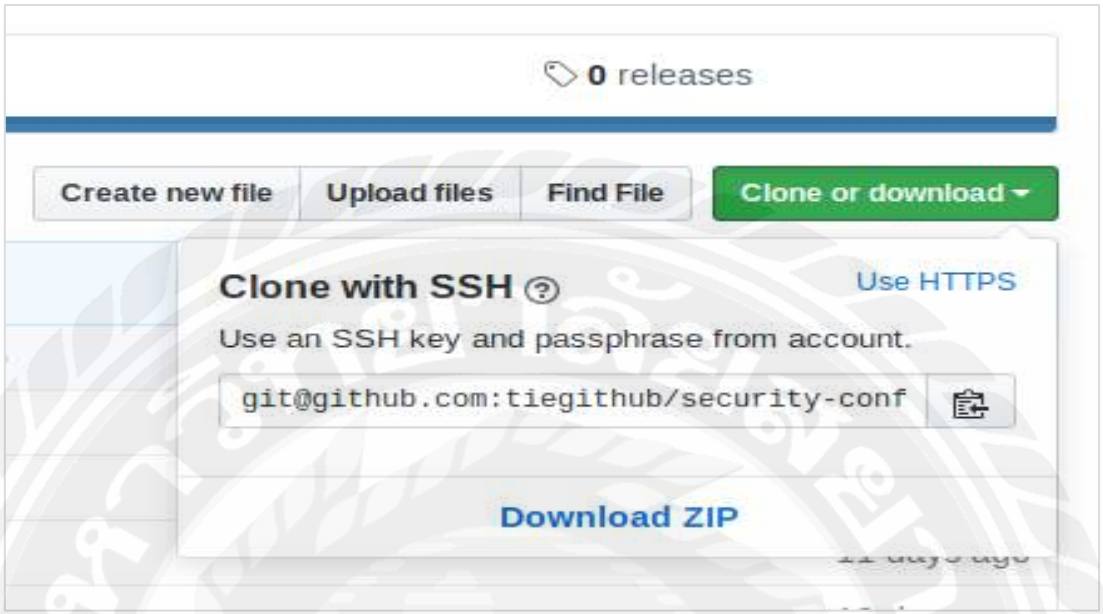
รูปที่ 4.2 หน้าต่างสำหรับเพิ่ม SSH Key

จากรูปที่ 4.2 หลังจากเข้าหน้า Setting ให้เลือกเมนู SSH and GPG keys จากแทบด้านซ้ายและใส่ข้อมูลที่จำเป็นในหน้าต่างด้านขวาได้แก่

- Title : ชื่อคีย์
- Key : ใส่คีย์สาธารณะ (Public Keys) จากเครื่องของผู้ใช้ ตัวอย่างเช่น id_rsa.pub

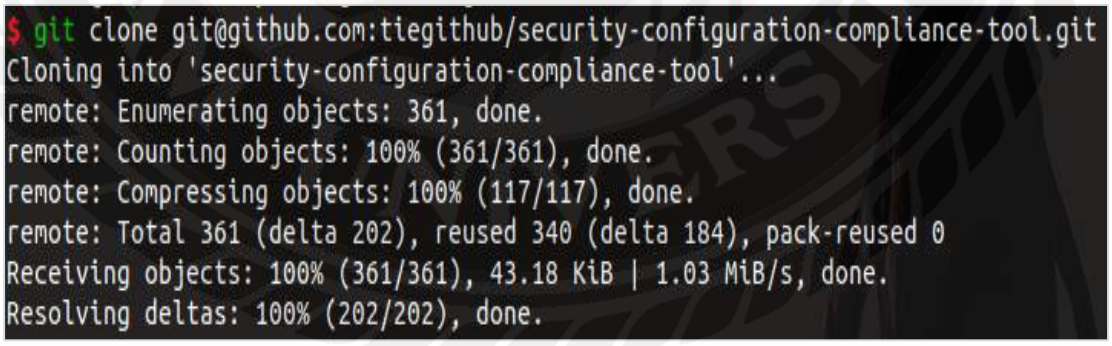
4.1.3 สร้าง Local Repository บนเครื่องผู้ใช้

ในขั้นตอนนี้จะเป็นการสร้าง Local Repository บนเครื่องผู้ใช้ โดยจะเป็นการโคลน Repository จาก GitHub (Remote Repository) ลงมาไว้ที่เครื่องผู้ใช้งาน



รูปที่ 4.3 เลือกโคลน Repository แบบ SSH

จากรูปที่ 4.3 กลับมาที่หน้า Repository และเลือก Clone or download พร้อมกับเลือกประเภทเป็น Clone with SSH แล้วทำการคัดลอก URL เพื่อนำมาใช้ในการโคลน Repository ต่อไป

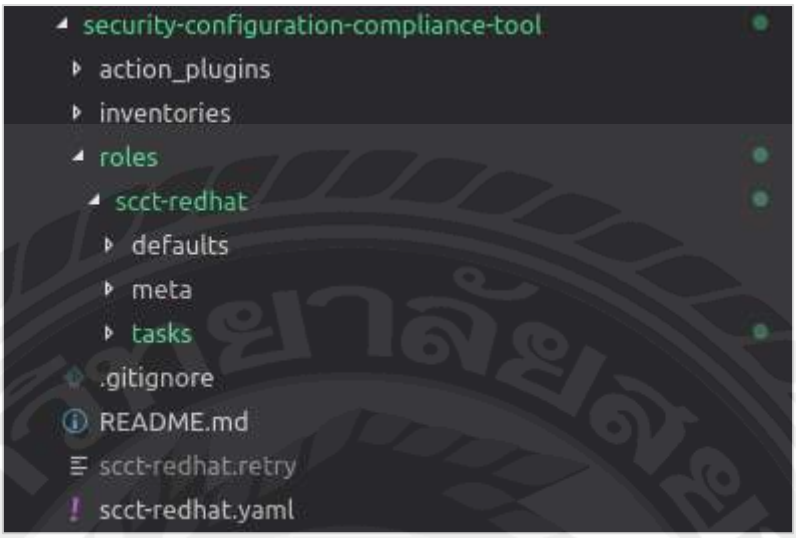


รูปที่ 4.4 คำสั่ง git clone สำหรับการโคลน Repository

จากรูปที่ 4.4 เปิดโปรแกรม Terminal แล้วส่งคำสั่ง git clone ตามด้วย URL ของ Repository เพื่อโคลนมาเป็น Local Repository และใช้งานต่อไป

4.1.4 สร้างโครงสร้างของ Ansible

ทำการออกแบบโครงสร้างตามรูปแบบที่ Ansible แนะนำโดยมีโครงสร้างดังต่อไปนี้

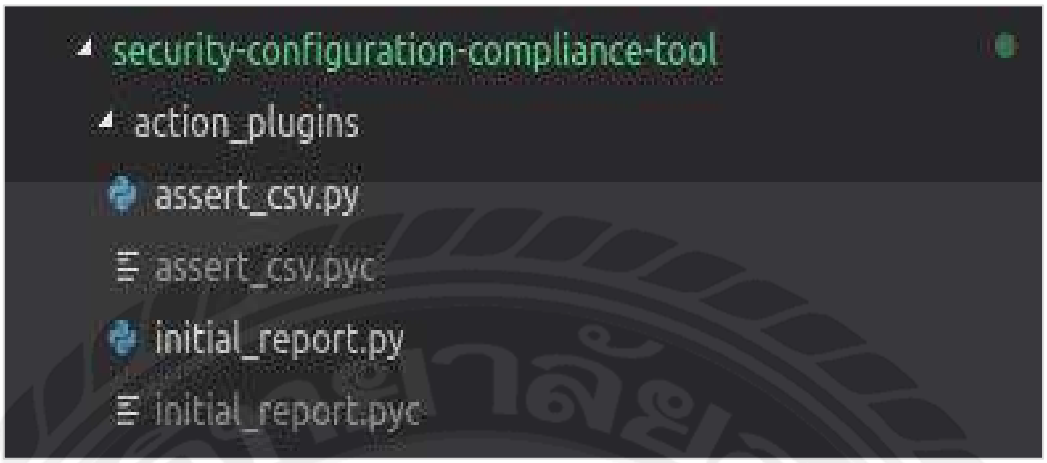


รูปที่ 4.5 โครงสร้างของ Ansible

จากรูปที่ 4.5 จะเห็นได้ว่ารูทของไดเรกทอรี่ (Root Directory) นี้คือ security-configuration-compliance-tool ที่ทำการโคลนมาจาก Remote Repository โดยผู้จัดทำได้ใส่โครงสร้างที่จำเป็นได้แก่

- action_plugins/ : เป็นไดเรกทอรี่สำหรับเก็บโมดูลของ Ansible ที่มีการสร้างขึ้นมาใหม่หรือแก้ไขของเดิมที่มีอยู่
- inventories/ : เป็นไดเรกทอรี่สำหรับเก็บที่อยู่โฮสปลายทางที่จะนำ Ansible Script นี้ไปทำงาน
- roles/ : เป็นไดเรกทอรี่สำหรับเก็บโปรเจคทั้งหมดในที่นี้คือโปรเจค scct-redhat
- roles/scct-redhat/defaults/ : เป็นไดเรกทอรี่สำหรับกำหนด Global Variable ของโปรเจค
- roles/scct-redhat/meta/ : เป็นไดเรกทอรี่สำหรับเก็บเมตะดาต้าที่จำเป็นสำหรับโปรเจคเช่น LICENSE เป็นต้น
- roles/scct-redhat/tasks/ : เป็นไดเรกทอรี่สำหรับเก็บงานทั้งหมดที่ Ansible จะต้องนำไปทำกับเครื่องปลายทาง
- scct-redhat.yaml : เป็นไฟล์ Ansible Playbook ที่ใช้ในการเรียก Role scct-redhat ให้ทำงาน

4.1.5 พัฒนาปลั๊กอิน (Plugin) สำหรับออกรายงานแสดงผลรูปแบบ CSV ไฟล์



รูปที่ 4.6 รายการปลั๊กอินที่จำเป็นต่อการออกรายงานแสดงผล

จากรูปที่ 4.6 จะเป็นไฟล์ที่จำเป็นสำหรับออกรายงานการแสดงผลของโปรเจค โดยจะแบ่งเป็น 2 ไฟล์ได้แก่ ไฟล์ initial_report.py และ ไฟล์ assert_csv.py โดยที่ไฟล์ initial_report.py จะเป็นไฟล์ที่ใช้ในการสร้างส่วนหัวของรายงานแสดงผลและไฟล์ assert_csv.py จะเป็นไฟล์ที่ใช้แสดงผลการทำงานแบบข้อต่อข้อเพื่อนำไปออกรายงาน โดยไฟล์ initial_report.py นั้นจะทำงานเพียงแค่ครั้งต่อการทำงานของโปรแกรมหนึ่งครั้งแต่ไฟล์ assert_csv.py จะทำงานทุกครั้งที่มีงานใน Ansible (Ansibl Task) ถูกทำงาน

```
def initial_report(self, standard_name, os_version, scan_date, host, last_scan):

    report_dir = os.path.join(self.REPORT_DIR, self._task.args.get('report_date'))
    # report_dir = 'reports'
    if not os.path.exists(report_dir):
        os.makedirs(report_dir)
    path = os.path.join(report_dir, host + '.csv')

    msg = to_bytes(self.MSG_FORMAT % dict(standard_name=standard_name, host=host, os_version=os_version,
    with open(path, "ab") as fd:
        fd.write(msg)

def run(self, tmp=None, task_vars=None):
    if task_vars is None:
        task_vars = dict()

    result = super(ActionModule, self).run(tmp, task_vars)
    del tmp  # tmp no longer has any effect

    success_msg = 'Reports are successfully initialized'

    result['_ansible_verbose_always'] = True
    result['changed'] = False
    result['msg'] = success_msg
    last_scan = self.get_lasttimes(task_vars['ansible_hostname'], self._task.args.get('scan_date'))
    self.initial_report(self._task.args.get('standard_name'), task_vars['ansible_distribution_version'],
    self.save_lasttimes(task_vars['ansible_hostname'], self._task.args.get('scan_date'))
    return result
```

รูปที่ 4.7 สคริปต์บางส่วนของไฟล์ initial_report.py

Standard Name:	RHEL7 Compliance		
Server Name:	rh4		
OS Version:	7.4		
Scan date and time:	2019/05/10-12:36:05		
Scan passed:	26		
Scan Failed:	30		
Last Scan:	2019/05/10-12:36:05		
Item No.	Level	Task Requirement	Action or Value

รูปที่ 4.8 ผลลัพธ์ที่ได้จากการทำงานของไฟล์ initial_report.py

```
def report_csv(self, item, level, topic, action, checkcmd, state, remark, hostname):

    report_dir = os.path.join(self.REPORT_DIR, self._task.args.get('report_date'))
    # report_dir = 'reports'
    if not os.path.exists(report_dir):
        os.makedirs(report_dir)
    REPORT_FILE = os.path.join(report_dir, hostname + '.csv')

    msg = to_bytes(self.MSG_FORMAT % dict(item=item, level=level, topic=topic, action=action, checkcmd=checkcmd,
    with open(REPORT_FILE, "ab") as fd:
        fd.write(msg)

def run(self, tmp=None, task_vars=None):
    if task_vars is None:
        task_vars = dict()

    result = super(ActionModule, self).run(tmp, task_vars)
    del tmp  # tmp no longer has any effect

    if 'that' not in self._task.args:
        raise AnsibleError('conditional required in "that" string')

    fail_msg = None
    success_msg = None

    fail_msg = self._task.args.get('fail_msg', self._task.args.get('msg'))
    if fail_msg is None:
        fail_msg = 'Assertion failed'
    elif not isinstance(fail_msg, string_types):
        raise AnsibleError('Incorrect type for fail_msg or msg, expected string and got %s' % type(fail_msg))

    success_msg = self._task.args.get('success_msg')
```

รูปที่ 4.9 สคริปต์บางส่วนของไฟล์ assert_csv.py

1.1	M	Separate partition f, Df or /tmp	For new installations, check
1.5	M	Separate Partition for /var	For new installations, check
1.7	O	Separate Partition for /var/log	For new installations, check
1.8	O	Separate Partition for /var/log/audit	For new installations, check
1.9	M	Separate Partition for /home	For new installations, check
2.4	O	Disable the rhnsd Daemon	# systemctl disable rhnsd

รูปที่ 4.10 ผลลัพธ์ที่ได้จากการทำงานของไฟล์ assert_csv.py

4.1.5 สร้าง inventory ไฟล์สำหรับระบุเครื่องปลายทาง

รูปที่ 4.11 ไฟล์ inventory สำหรับระบุเครื่องปลายทาง

จากรูปที่ 4.11 จะเห็นได้ว่าภายในไดเรกทอรี่ inventories/ จะมีไฟล์ที่ชื่อว่า scct-redhat เพื่อใช้สำหรับระบุเครื่องปลายทางที่จะนำสคริปต์ไปทำงาน

```
[all]
rh1 ansible_user=sysadmin ansible_host=10.88.248.1 ansible_port=22 ansible_sudo_pass=P@ssw0rd
rh2 ansible_user=sysadmin ansible_host=10.88.248.2 ansible_port=22 ansible_sudo_pass=P@ssw0rd
rh3 ansible_user=sysadmin ansible_host=10.88.248.3 ansible_port=22 ansible_sudo_pass=P@ssw0rd
rh4 ansible_user=sysadmin ansible_host=10.88.248.4 ansible_port=22 ansible_sudo_pass=P@ssw0rd
```

รูปที่ 4.12 เนื้อหาของไฟล์ scct-redhat

จากรูปที่ 4.12 แสดงให้เห็นถึงเนื้อหาของไฟล์ scct-redhat โดยมีรายละเอียดดังนี้

- [all] : แสดงถึงกลุ่มของโฮสต์ในที่นี้ไม่มีการระบุกลุ่มจึงใส่ไว้ในกลุ่ม all
- rh1, rh2, rh3, rh4 : แสดงถึงชื่อโฮสต์ปลายทาง
- ansible_user=sysadmin : เป็นการบอก Ansible ให้สั่งสคริปต์ทำงานด้วยชื่อบัญชีผู้ใช้ sysadmin
- ansible_host=10.88.248.x : ระบุที่อยู่ของเครื่องปลายทาง (IP Address)
- ansible_sudo_pass=P@ssw0rd : ระบุรหัสผ่าน sudo เพื่อยกสิทธิ์ขึ้นเป็นรูทในการเรียกใช้การทำงานสคริปต์บางข้อที่ต้องใช้สิทธิ์รูท

4.1.6 กำหนด Global Variable ใน Ansible Role

รูปที่ 4.13 ไฟล์ main.yml ที่ใช้เก็บ global variable

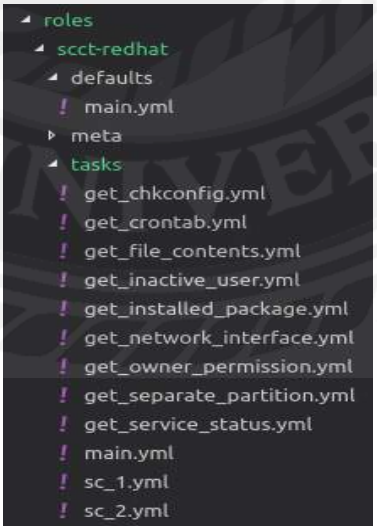
จากรูปที่ 4.13 แสดงถึงโครงสร้างและไฟล์ main.yml ที่อยู่ภายในไดเรกทอรี่ defaults ซึ่งใช้เก็บตัวแปร global variable ที่จะถูกอ่านทุกครั้งเมื่อ role นี้ถูกเรียกใช้งาน

```
current_date: "{{ lookup('pipe', 'date +%Y-%m-%d-%H-%M-%S') }}"
tsal_scan_date: "{{ lookup('pipe', 'date +%Y/%m/%d-%H:%M:%S') }}"
```

รูปที่ 4.14 เนื้อหาของไฟล์ main.yml

จากรูปที่ 4.14 เป็นการประกาศตัวแปร 2 ตัวได้แก่ current_date และ tsal_scan_date เพื่อใช้ในการระบุวันและเวลาในการตรวจสอบและในการออกรายงานการแสดงผล

4.1.7 เขียนสคริปต์สำหรับตรวจสอบระบบ (Main Tasks)



รูปที่ 4.15 โครงสร้างของสคริปต์ (Main Tasks)

จากรูปที่ 4.15 แสดงถึงโครงสร้างของไดเรกทอรี่ tasks/ ซึ่งภายในจะประกอบไปด้วย

- main.yml : เป็นไฟล์แรกที่จะถูกอ่านเมื่อ Ansible อ่านมาถึงไดเรกทอรี่นี้
- get_*.yml : เป็นไฟล์สำหรับเก็บฟังส์ชั่นที่ใช้ในการไปอ่านค่าต่าง ๆ ตามข้อกำหนดที่ระบุ
- sc_*.yml :เป็นไฟล์สำหรับแต่ละหัวข้อโดยแบ่งตามหัวข้อของข้อกำหนดทั้งหมด 24 หัวข้อ

```
- name: set fact
  set_fact:
    tsal_report_dir: "{{ current_date }}"

- name: initial report
  initial_report:
    standard_name: "RHEL7 Compliance"
    scan_date: "{{ tsal_scan_date }}"
    report_date: "{{ tsal_report_dir }}"

- include_tasks: sc_1.yml
- include_tasks: sc_2.yml
- include_tasks: sc_3.yml
- include_tasks: sc_4.yml
- include_tasks: sc_5.yml
- include_tasks: sc_6.yml
```

รูปที่ 4.16 สคริปต์ภายในไฟล์ main.yml

จากรูปที่ 4.16 แสดงถึงสคริปต์ที่อยู่ในไฟล์ main.yml ที่อยู่ในไดเรกทอรี่ tasks/ ซึ่งประกอบไปด้วย 3 ส่วนแบ่งตาม - name ได้แก่

- set fact : เป็นส่วนที่ใช้ในการระบุการสร้างไดเรกทอรี่ภายใต้ไดเรกทอรี่ reports/ โดยแบ่งตามวันและเวลาในการทำงานของสคริปต์แต่ละครั้ง

รูปที่ 4.17 ตัวอย่างโครงสร้างในไดเรกทอรี่ reports/

- initial report : เป็นส่วนที่ใช้ระบุข้อมูลส่วนหัวของรายงานได้แก่ ชื่อมาตรฐานที่ใช้ตรวจสอบ, วันและเวลาที่ตรวจสอบและวันและเวลาสำหรับไดเรกทอรี่จะที่เก็บรายงานครั้งนี้
- include_tasks: เป็นการประกาศให้นำไฟล์ตั้งแต่ไฟล์ sc_1.yml - sc_24.yml เข้ามาทำงานเพื่อทำการตรวจสอบตามข้อกำหนด

```
- include: roles/scct-redhat/tasks/get_separate_partition.yml part=/var
- name: Assert separate partition /var
  ignore_errors: true
  assert_csv:
    item_no: "1.5"
    level: "M"
    topic: "Separate Partition for /var"
    action: "For new installations, check the box to \"Review and modify
    For systems that were previously installed, use the Logical Volume
    check_command: "findmnt --fstab | grep /var"
    remark: "Since the /var directory may contain world-writable files
    report_date: "{{ tsal_report_dir }}"
    that:
      - get_separate_partition.rc == 0
```

รูปที่ 4.18 สคริปต์สำหรับตรวจสอบพาติชั่น /var จากไฟล์ sc_1.yml

จากรูปที่ 4.18 เป็นตัวอย่างสคริปต์สำหรับตรวจสอบพาติชั่น /var ซึ่งนำมาจากไฟล์ sc_1.yml ซึ่งจะเห็นว่ามีการใช้งานโมดูลต่าง ๆ ดังนี้

- โมดูล include เพื่อเรียกใช้งานไฟล์ get_separate_partition.yml เพื่อนำมาอ่านค่าพาติชั่นที่ต้องการสังเกตจาก part=/var ซึ่งเป็นการใส่พารามิเตอร์ให้แก่ฟังส์ชั่น
- โมดูล name ถูกนำมาใช้เพื่อประกาศชื่องานในระหว่างที่ Ansible ทำงานเพื่อให้ง่ายต่อการอ่าน
- โมดูล ignore_errors: true เป็นการบอกว่าหากเจอข้อผิดพลาดให้แสดงข้อผิดพลาดและข้ามไปทำข้อต่อไปซึ่งหากไม่ใส่ค่านี้ไว้แล้ว Ansible จะหยุดทำงานทันทีเมื่อเจอข้อผิดพลาด ซึ่งจะทำให้ไม่สามารถตรวจสอบข้อกำหนดอื่น ๆ ต่อจนเสร็จได้
- โมดูล assert_csv เป็นการเรียกใช้งานโมดูลที่สร้างขึ้นจากไฟล์ assert_csv.yml ที่อยู่ภายในไดเรกทอรี่ action_plugins/ โดยมีรายละเอียดดังนี้
- item_no: แสดงถึงเลขที่ข้อที่ทำการตรวจสอบ ซึ่งอ้างอิงจากเอกสารข้อกำหนด
- level: แสดงถึงความสำคัญของข้อนี้ ซึ่งอ้างอิงจากเอกสารข้อกำหนด

- topic: แสดงชื่อหัวข้อที่ตรวจสอบ ซึ่งอ้างอิงจากเอกสารข้อกำหนด
- action: แสดงถึงการกระทำต่อหัวข้อนี้ ซึ่งอ้างอิงจากเอกสารข้อกำหนด
- check_command: แสดงถึงคำสั่งที่ใช้ตรวจสอบ
- remark: แสดงถึงหมายเหตุ ซึ่งอ้างอิงจากเอกสารข้อกำหนด
- report_date: เป็นการระบุว่าการตรวจสอบข้อนี้ให้ไปอยู่ในเอกสารฉบับใดตอนออกรายงาน
- that: เป็นการตรวจสอบเงื่อนไข ที่จะระบุว่าการตรวจสอบข้อนี้ผ่านหรือไม่ผ่าน โดยผลลัพธ์ที่ได้จะออกมาเป็น True ชาร์จ False

```
- name: Get separate partition
  shell: "findmnt --fstab | grep '{{ part }} '"
  register: get_separate_partition
  changed_when: false
  failed_when: false
# - debug: var=get_separate_partition
```

รูปที่ 4.19 สคริปต์สำหรับอ่านค่าพาติชั่นต่าง ๆ จากไฟล์ get_separate_partition.yml

จากรูปที่ 4.19 ถูกออกแบบให้เป็นฟังส์ชั่นเพื่อนำไปใช้งานในการตรวจสอบพาติชั่นต่างๆ โดยจะถูกนำไปเรียกใช้งานในการตรวจสอบอีกที โดยมีรายละเอียดดังนี้

- name: เป็นการประกาศชื่องานเมื่อ Ansible ทำงาน
- shell: เป็นการเรียกใช้งานคำสั้ง shell ตามที่ระบุเพื่อทำการเก็บข้อมูลจากเครื่องปลายทาง
- register: เป็นการประกาศให้เก็บค่าตัวแปรที่ได้จากบรรทัดด้านบนใส่ตัวแปรตามที่ระบุ
- changed_when: false บอกให้ Ansible ไม่แสดงทำการเปลี่ยนแปลงค่าใด ๆ ก็ตามเมื่อทำงาน
- failed_when: false บอกให้ Ansible ไม่แสดงข้อผิดพลาดถึงแม้ว่าจะเจอข้อผิดพลาดก็ตาม
- debug: เป็นการแสดงเอาท์พุตจากตัวแปร get_separate_partition ใช้ในขั้นตอนการแก้ไขสคริปต์(Debug)

4.2 การติดตั้งระบบ (System Implementation)

เนื่องจากระบบได้ออกแบบไว้เป็นรูปแบบคอนเทนเนอร์ (Container) ดังนั้นในขั้นตอนการติดตั้งระบบจึงต้องใช้ docker-compose และไฟล์ compose เข้ามาช่วยในการเชื่อมต่อคอนเทนเนอร์ต่าง ๆ เข้าด้วยกัน เพื่อทำให้ระบบสามารถทำงานได้

4.2.1 ติดตั้งฐานข้อมูล PostgreSQL [7]

ในการติดตั้งระบบ Ansible AWX นั้นจำเป็นต้องมีการติดตั้งฐานข้อมูลเพื่อไว้ใช้สำหรับสำรองและกู้คืนข้อมูลเมื่อระบบมีปัญหา โดยฐานข้อมูลที่ระบบ Ansible AWX รองรับคือ PostgreSQL เวอร์ชั่น 9.6 ขึ้นไป ในขั้นตอนนี้จะเป็นการติดตั้ง PostgreSQL เวอร์ชั่น 10 บนระบบปฏิบัติการ Ubuntu 16.04 LTS มีขั้นตอนดังนี้

ขั้นตอนที่ 1 : ล็อกอินเป็นผู้ใช้งาน root ด้วยคำสั่ง

```
$ sudo -i
```

ขั้นตอนที่ 2 : เพิ่ม URL ต้นทางสำหรับติดตั้ง PostgreSQL ด้วยคำสั่ง

```
# echo “deb http://apt.postgresql.org/pub/repos/apt/ xenial-pgdg main” >> /etc/apt/sources.list.d/pgdg.list
```

ขั้นตอนที่ 3 : นำเข้า repository signing key ด้วยคำสั่ง

```
# wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
# sudo apt-get update
```

ขั้นตอนที่ 4 : ติดตั้ง PostgreSQL เวอร์ชั่น 10 ด้วยคำสั่ง

```
# apt-get install postgresql-10
```

[7]อ้างอิง https://www.postgresql.org/download/linux/ubuntu link

4.2.2 ตั้งค่าฐานข้อมูล PostgreSQL

ในขั้นตอนนี้จะเป็นการสร้างฐานข้อมูลและตั้งค่าเพื่อให้คอนเทนเนอร์สามารถเชื่อมต่อกับฐานข้อมูลที่ติดตั้งอยู่บนเครื่องโฮสได้ โดยมีขั้นตอนดังนี้

ขั้นตอนที่ 1 : ล็อกอินด้วยผู้ใช้งาน postgres ด้วยคำสั่ง

```
$ sudo -i
# su - postgres
```

ขั้นตอนที่ 2 : ล็อกอินเข้าฐานข้อมูลด้วยคำสั่ง

```
$ psql
```

ขั้นตอนที่ 3 : สร้างฐานข้อมูลและกำหนดผู้ใช้งานด้วยคำสั่ง

```
postgres=# create database awx;
postgres=# create user awx with encrypted password 'awxpass';
postgres=# grant all privileges on database awx to awx;
postgres=# alter user awx with login;
postgres=# \q
```

ขั้นตอนที่ 4 : ออกจากผู้ใช้งาน postgres เพื่อกลับไปเป็น root ด้วยคำสั่ง

```
$ exit
```

ขั้นตอนที่ 5 : ตั้งค่าไฟล์ /etc/postgresql/10/main/postgresql.conf เพื่อให้ postgres รับการเชื่อมต่อทั้งหมด

```
listen_addresses = '*'
```

รูปที่ 4.20 ตั้งค่าการเชื่อมต่อในไฟล์ postgresql.conf

ขั้นตอนที่ 6 : ตั้งค่าไฟล์ /etc/postgresql/10/main/pg_hba.conf เพื่อให้ postgres อนุญาติให้ docker เข้าถึงฐานข้อมูล

```
# IPv4 docker connections:
host    awx             awx             172.0.0.0/8         md5
```

รูปที่ 4.21 ตั้งค่าการเชื่อมต่อในไฟล์ pg_hba.conf

ขั้นตอนที่ 7 : รีสตาร์ทฐานข้อมูลด้วยคำสั่ง

```
# systemctl restart postgresql-10
```

4.2.3 ติดตั้ง Docker CE และ Docker Compose[8]

ในขั้นตอนนี้จะเป็นการติดตั้งโปรแกรม Docker และ Docker Compose เพื่อใช้สำหรับสร้างและการจัดการคอนเทนเนอร์ต่างๆ ในระบบ โดยมีขั้นตอนดังนี้

ขั้นตอนที่ 1 : อัพเดท apt แพ็คเก็จ

```
$ sudo apt-get update
```

ขั้นตอนที่ 2 : ติดตั้งแพ็คเก็จที่จำเป็น

```
$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
```

[8]อ้างอิง https://docs.docker.com/install/linux/docker-ce/ubuntu/

[8]อ้างอิง https://docs.docker.com/compose/install/

ขั้นตอนที่ 3 : เพิ่ม Docker GPG คีย์

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

ขั้นตอนที่ 4 : ติดตั้ง Docker stable repository

```
$ sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
```

ขั้นตอนที่ 5 : อัพเดท apt แพคเกจ

```
$ sudo apt-get update
```

ขั้นตอนที่ 6 : ติดตั้ง Docker

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

ขั้นตอนที่ 7 : ตั้งค่าให้สามารถใช้ Docker ได้โดยไม่ผ่านคำสั่ง sudo

```
$ sudo usermod -aG docker $USER
```

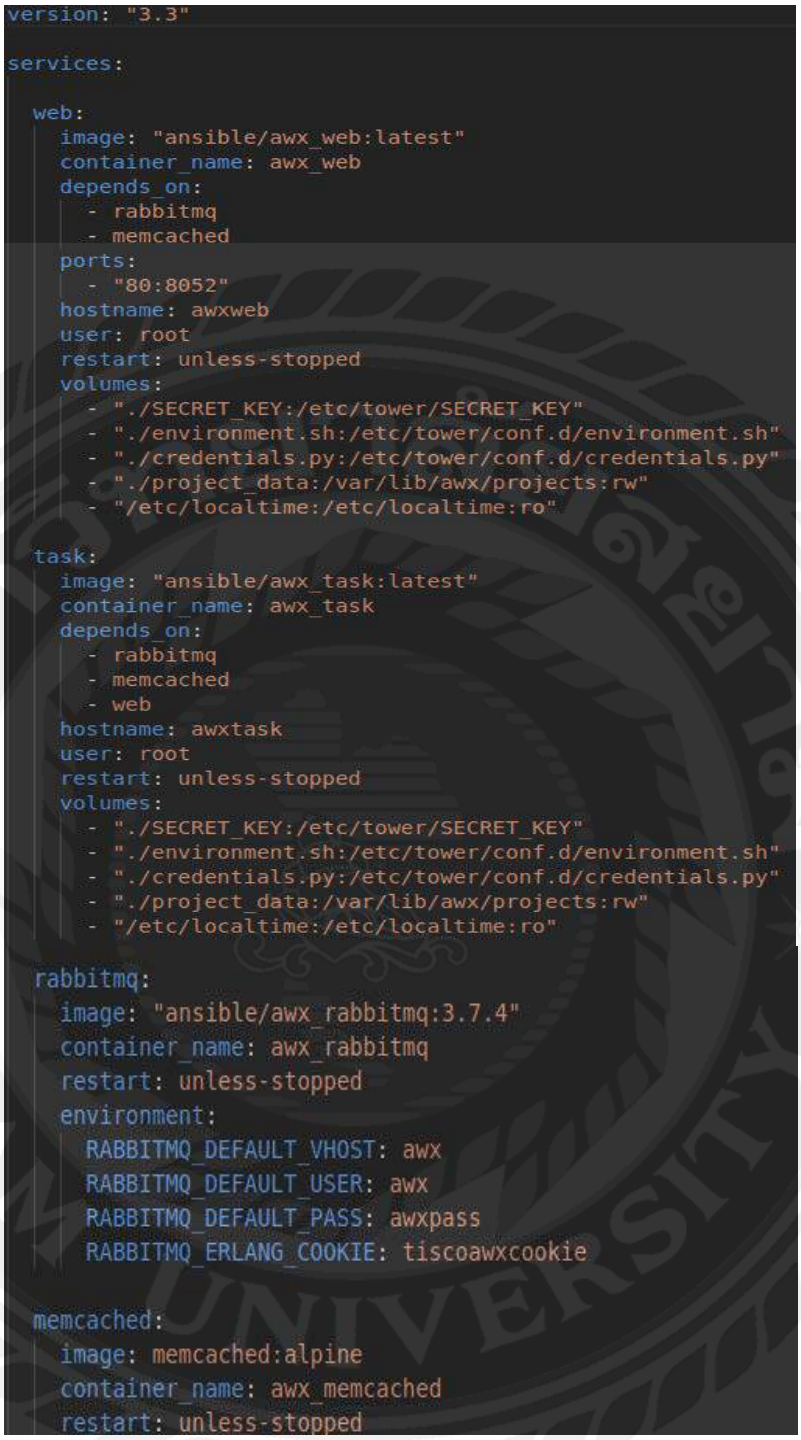
ขั้นตอนที่ 8 : ดาว์นโหลด Docker compose

```
$ sudo curl -L "https://github.com/docker/compose/releases/download/1.24.0/docker-compose-$(uname -s)-
$(uname -m)" -o /usr/local/bin/docker-compose
```

ขั้นตอนที่ 9 : เพิ่มสิทธิ์การ execute ให้แก่ Docker compose

```
$ sudo chmod +x /usr/local/bin/docker-compose
```

4.2.4 สร้างไฟล์ docker-compose.yml[9]



```yaml
version: "3.3"

services:

  web:
    image: "ansible/awx_web:latest"
    container_name: awx_web
    depends_on:
      - rabbitmq
      - memcached
    ports:
      - "80:8052"
    hostname: awxweb
    user: root
    restart: unless-stopped
    volumes:
      - "./SECRET_KEY:/etc/tower/SECRET_KEY"
      - "./environment.sh:/etc/tower/conf.d/environment.sh"
      - "./credentials.py:/etc/tower/conf.d/credentials.py"
      - "./project_data:/var/lib/awx/projects:rw"
      - "/etc/localtime:/etc/localtime:ro"

  task:
    image: "ansible/awx_task:latest"
    container_name: awx_task
    depends_on:
      - rabbitmq
      - memcached
      - web
    hostname: awxtask
    user: root
    restart: unless-stopped
    volumes:
      - "./SECRET_KEY:/etc/tower/SECRET_KEY"
      - "./environment.sh:/etc/tower/conf.d/environment.sh"
      - "./credentials.py:/etc/tower/conf.d/credentials.py"
      - "./project_data:/var/lib/awx/projects:rw"
      - "/etc/localtime:/etc/localtime:ro"

  rabbitmq:
    image: "ansible/awx_rabbitmq:3.7.4"
    container_name: awx_rabbitmq
    restart: unless-stopped
    environment:
      RABBITMQ_DEFAULT_VHOST: awx
      RABBITMQ_DEFAULT_USER: awx
      RABBITMQ_DEFAULT_PASS: awxpass
      RABBITMQ_ERLANG_COOKIE: tiscoawxcookie

  memcached:
    image: memcached:alpine
    container_name: awx_memcached
    restart: unless-stopped
```

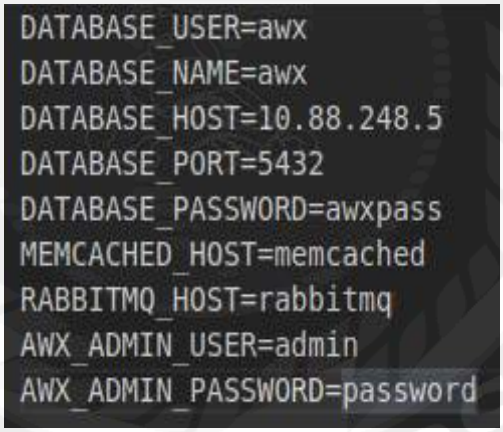
รูปที่ 4.22 เนื้อหาภายในไฟล์ docker-compose.yml

[9]อ้างอิง https://github.com/ansible/awx/blob/devel/installer/roles/local_docker/templates/docker-compose.yml.j2 link

จากรูปที่ 4.22 เป็นชุดคำสั่งภายในไฟล์ docker-compose.yml โดยมีรายละเอียดดังนี้

- version: ระบุ compose เวอร์ชั่น
- service: ระบุว่ามีเซอร์วิสใดบ้างในไฟล์ compose ได้แก่ web, task, rabbitmq และ memcached
- image: ระบุ docker image ที่จะนำมาใช้งาน
- container_name: ตั้งชื่อคอนเทนเนอร์เมื่อคอนเทนเนอร์ทำงาน
- depens_on: ระบุว่าเซอร์วิสใดบ้างที่ต้องเริ่มทำงานก่อนเซอร์วิสนี้
- hostname: ตั้งชื่อ hostname ของคอนเทนเนอร์
- user: ระบุผู้ใช้งานเมื่อคอนเทนเนอร์ทำงาน
- restart: กำหนดให้คอนเทนเนอร์ทำงานแบบอัตโนมัติ
- volumes: กำหนดว่าไฟล์ใดบ้างที่จะต้องถูกเม้าเมื่อคอนเทนเนอร์ทำงาน
- environment: กำหนดค่า environment variable เมื่อคอนเทนเนอร์ทำงาน

4.2.5 สร้างไฟล์ enviroment.sh[10]



รูปที่ 4.23 เนื้อหาภายในไฟล์ enviroment.sh

จากรูปที่ 4.23 เป็นการกำหนดตัวแปรที่จำเป็นไว้ในไฟล์ enviroment.sh เพื่อทำการคัดลอกเมื่อ Ansible AWX เริ่มทำงาน

[10]อ้างอิง https://github.com/ansible/awx/blob/devel/installer/roles/local_docker/templates/environment.sh.j2 link

4.2.6 สร้างไฟล์ credentials.py[11]

```
DATABASES = {
    'default': {
        'ATOMIC_REQUESTS': True,
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': "awx",
        'USER': "awx",
        'PASSWORD': "awxpass",
        'HOST': "10.88.248.5",
        'PORT': "5432",
    }
}
BROKER_URL = 'amqp://{}:{}@{}:{}/{}'.format(
    "awx",
    "awxpass",
    "rabbitmq",
    "5672",
    "awx")

CHANNEL_LAYERS = {
    'default': {'BACKEND': 'asgi_amqp.AMQPChannelLayer',
                'ROUTING': 'awx.main.routing.channel_routing',
                'CONFIG': {'url': BROKER_URL}}
}
CLUSTER_HOST_ID = "awxtask"
```

รูปที่ 4.24 เนื้อหาภายในไฟล์ credentials.py

จากรูปที่ 4.24 ไฟล์ credentials.py จะถูกเม้าเข้าไปที่คอนเทนเนอร์ awx_web และ awx_task เมื่อเริ่มทำงาน เพื่อให้คอนเทนเนอร์เหล่านี้รู้ว่าต้องต่อไปยังฐานข้อมูลและคอนเทนเนอร์ rabbitmq ได้อย่างไร

4.2.7 สร้างไฟล์ SECRET_KEY

```
vX6gQ8HuaMLECudd87peSW5E
```

รูปที่ 4.25 Secret_key สำหรับ Ansible AWX

จากรูปที่ 4.25 เป็นการกำหนด Secret key สำหรับระบบ Ansible AWX ซึ่งไฟล์จะถูกคัดลอกเมื่อคอนเทนเนอร์ awx_web และ awx_task เริ่มทำงาน

[11]อ้างอิง https://github.com/ansible/awx/blob/devel/installer/roles/local_docker/templates/credentials.py.j2

4.2.8 สั่งไฟล์ docker-compose.yml ให้เริ่มทำงาน

```
ubuntu@tie-fp-awx-01:~/awx$ docker-compose up -d
Creating network "awx_default" with the default driver
Creating awx_rabbitmq  ... done
Creating awx_memcached ... done
Creating awx_web       ... done
Creating awx_task      ... done
```

รูปที่ 4.26 สั่งให้ docker-compose เริ่มทำงาน

จากรูปที่ 4.26 หลังจากเตรียมไฟล์ทั้งหมดครบแล้วก็สั่งเริ่มทำงานด้วยคำสั่ง docker-compose up โดยคำสั่ง docker-compose up จะหาไฟล์ที่ชื่อ docker-compose.yml และสั่งทำงานโดยอัตโนมัติ สำหรับแท็ก -d หมายถึง ให้คำสั้งนี้ทำงานอยู่เบื้องหลัง

รูปที่ 4.27 หน้าล็อกอินของ Ansible AWX

จากรูปที่ 4.27 หลังจากสั่ง docker-compose แล้ว ทดสอบเข้าหน้าเว็บบราว์จะพบกับหน้าต่างล็อกอินของ Ansible AWX ซึ่งแสดงว่าระบบทำงานถูกต้อง

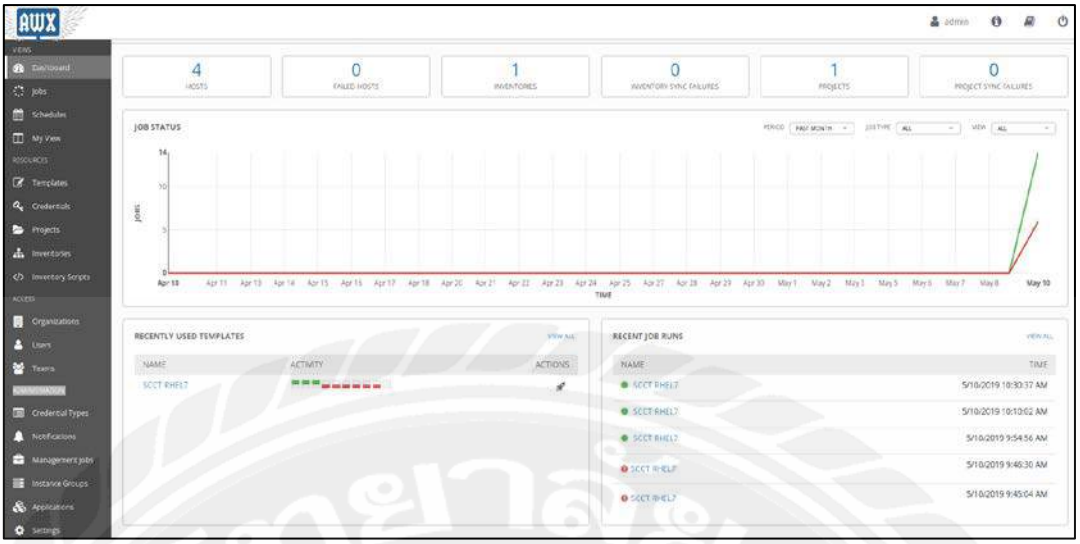
4.3 ส่วนติดต่อผู้ใช้ (User Interface)

ในส่วนติดต่อผู้ใช้ของ Ansible AWX ทำออกมาได้เรียบง่าย มีเมนูการใช้งานอยู่ที่แถบด้านซ้ายประกอบด้วย 4 ส่วนหลักๆ คือ

- Views ใช้สำหรับดูภาพรวมของระบบทั้งหมด เช่น ประสิทธิภาพของระบบ, จำนวนงานทั้งหมด, การตั้งตารางงาน เป็นต้น
- Resources เป็นส่วนที่ใช้ในการกำหนดทรัพยากรณ์ต่างๆที่จะนำระบบไปใช้งาน ได้แก่ Source code ที่จะใช้ทำงาน, กำหนดเครื่องปลายทางที่จะนำ Source code ไปใช้งาน เป็นต้น
- Access เป็นส่วนที่ใช้สำหรับกำหนดผู้เข้าถึงระบบต่างๆ
- Administration เป็นส่วนที่ใช้สำหรับกำหนดค่าคอนฟิกกูเรชันต่างๆของระบบ

รูปที่ 4.28 หน้าล็อกอินเข้าใช้งานระบบ

จากรูปที่ 4.28 เป็นหน้าสำหรับการยืนยันตัวตนเพื่อเข้าใช้งานระบบโดยผู้ใช้ต้องระบุ USERNAME และ PASSWORD ที่ถูกต้อง แล้วทำการกดปุ่ม SIGN IN เพื่อเข้าใช้งานระบบ

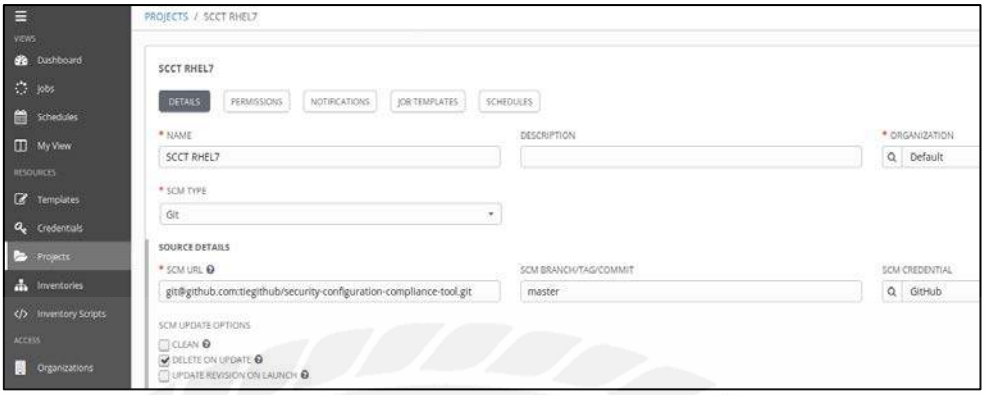


รูปที่ 4.29 หน้า Dashboard ของระบบ

จากรูปที่ 4.29 เป็นหน้าจอหลักของระบบโดยจะมีเมนูที่จำเป็นต่อการใช้งานอยู่ด้านซ้าย โดยจะเน้นเป็นสีเทาเข้าเพื่อให้เห็นได้ชัดเจนสำหรับในส่วนของหน้าจอหลักจะมีกราฟแสดงประสิทธิภาพโดยรวมของระบบพร้อมกับแสดงถึงจำนวนการทำงานที่ผิดพลาดและถูกต้อง

รูปที่ 4.30 หน้า Credentials ของระบบ

จากรูปที่ 4.30 เมื่อผู้ใช้งานเลือกแท็บ Credentials จากเมนูด้านซ้าย จะเข้าสู่หน้าจอที่ใช้สำหรับสร้างหรือแก้ไขใบรับรองภายในระบบทั้งหมด ใบรับรองนี้จะนำไปใช้สำหรับการเชื่อมระบบ Ansible AWX เข้ากับระบบต่างๆ เช่น GitHub และ โอสต์ปลายทาง



รูปที่ 4.31 หน้า Project ของระบบ

จากรูปที่ 4.31 ในหน้าจอนี้จะใช้สำหรับสร้างโครงการ (Project) บน Ansible AWX ซึ่งจะต้องนำไปผูกกับ Repository บน GitHub เพื่อที่จะนำ Source code (Ansible Playbook และ Ansible Role) มาใช้งาน โดยผู้ใช้งานจะต้องกรอกข้อมูลที่จำเป็นได้แก่

- NAME : ชื่อของโปรเจค
- SCM TYPE : ประเภทของ Source code ในที่นี้เลือกเป็น Git
- SCM URL : ถ้าเลือก SCM TYPE เป็น Git ในส่วนที่ให้ใส่ GitHub Repository URL
- SCM CREDENTIAL : เลือก GitHub ซึ่งเป็นใบรับรองที่ได้สร้างไว้ก่อนหน้านี้
- SCM BRANCH : ใส่ branch ที่ต้องการให้ระบบไปดึง Source code ในที่นี้เป็น master
- DELETE ON UPDATE : เลือกไว้ในกรณีอัพเกรดโครงการให้ทำการลบ Source code ออกก่อนแล้วค่อยสร้างใหม่

รูปที่ 4.32 หน้า Inventories ของระบบ

จากรูปที่ 4.32 ในหน้าจอนี้จะใช้สำหรับการกำหนดรายชื่อโฮสต์ปลายทางที่ต้องการให้ระบบทำการตรวจสอบ โดยต้องตั้งชื่อและระบุที่อยู่ (IP Address) ของโฮสปลายทางให้ถูกต้อง

รูปที่ 4.33 หน้า Template ของระบบ

จากรูปที่ 4.33 เป็นหน้าจอที่ใช้กำหนดเทมเพลท (Template) ของระบบ โดยหลักการจะเป็นการจำคู่โครงการ (Project) เข้ากับ รายการโฮสต์ที่จะถูกตรวจสอบ (Inventories) โดยมีข้อมูลที่ต้องกรอกได้แก่

- NAME : ชื่อของเทมเพลท
- JOB TYPE : เป็นเภทของงานในที่นี้เลือก Run
- INVENTORY : เลือกจาก Inventory ที่จะใช้งาน
- PROJECT : เลือกจาก Project ที่จะใช้งาน
- PLAYBOOK : เลือก Ansible Playbook ที่จะใช้งาน
- CREDENTIAL : เลือกใบรับรองสำหรับโอสปลายทาง

รูปที่ 4.34 สั่งเรียกการทำงานจากเทมเพลท

จากรูปที่ 4.34 เมื่อสร้างเทมเพลทเสร็จเรียบร้อยแล้วก็จะมาขึ้นขั้นตอนการเรียกใช้งานเทมเพลทโดยคลิกที่รูปจรวดเพื่อเริ่มทำงาน

รูปที่ 4.35 รายละเอียดการทำงานของเทมเพลท

จากรูปที่ 4.35 จะเป็นการแสดงรายละเอียดการทำงานของเทมเพลทเมื่อเทมเพลทถูกเรียกใช้งานโดยจะบอกสถานะว่ากำลังทำงานอยู่ (Running) ที่มุมบนซ้ายแล้วแสดง Logs ที่ออกมาจากระบบในส่วนด้านขวาของหน้าจอเพื่อให้ผู้ใช้งานได้ติดตามดูได้อย่างใกล้ชิด

	A	B	C	D	
1					
2		Standard Name:	Test module standard		
3		Server Name:	rh4		
4		OS Version:	7.4		
5		Scan date and time:	2019/05/10-04:58:40		
6		Scan passed:	26		
7		Scan Failed:	31		
8		Last Scan:	2019/05/10-03.27.27		
9		Exception memo			
10					
11		Item No.	Level	Task Requirement	Action or Value
12		1.1	M	Separate partition f, Df or /tmp	For new installations, check the box to "Review and modify partitioning" and create a
13		1.5	M	Separate Partition for /var	For new installations, check the box to "Review and modify partitioning" and create a
14		1.7	O	Separate Partition for /var/log	For new installations, check the box to "Review and modify partitioning" and create a
15		1.8	O	Separate Partition for /var/log/audit	For new installations, check the box to "Review and modify partitioning" and create a
16		1.9	M	Separate Partition for /home	For new installations, check the box to "Review and modify partitioning" and create a
17		2.4	O	Disable the rhnsd Daemon	# systemctl disable rhnsd
18		3.1	M	Install AIDE	# yum install aide <Output messages from Yum install> aide. <hardware platform> <
19		3.2	M	Implement Periodic Execution of File Integrity	# crontab -u root -e0 5 * * * /usr/sbin/aide --check
20		4.1	M	Ensure SELinux is not disabled in /boot/grub2/grub.cfg	Remove all instances of selinux=0 and enforcing=0 from /etc/grub2.cfg
21		4.2	M	Set the SELinux State	Edit the /etc/selinux/config file to set the SELINUX parameter SELINUX=enforcing
22		5.1	M	Set User/Group Owner on /boot/grub2/grub.cfg	# chown root:root /etc/grub2.cfg
23		5.2	M	Set Permissions on /boot/grub2/grub.cfg	# chmod og-rwx /etc/grub2.cfg
24		5.3	M	Set Boot Loder Password	# grub2-mkpasswd-pbkdf2 Enter password: <password> Reenter password: <pass
25	6.1.1		M	Restrict Core Dumps	Add the following line to the /etc/security/limits.conf file. * hard core 0
26	6.1.2		M	Restrict Core Dumps	Add the following line to the /etc/sysctl.conf file. fs.suid_dumpable = 0
27		6.2	M	Enable Randomized Virtual Memory Region Placement	Add the following line to the /etc/sysctl.conf file. kernel.randomize_va_space = 2
28		7.1	M	Use the Lates OS Release	# cat /etc/redhat-release

รูปที่ 4.36 รูปแบบรายงานผลการทดสอบ

จากรูปที่ 4.36 เป็นรูปแบบของรายงานผลการทดสอบหลังจากเป็นโดยโปรแกรม LibreOffice Calc โดยจะแสดง ชื่อมาตรฐานที่ใช้ในการตรวจสอบ ระบบปฏิบัติการของโอสต์ที่ตรวจสอบวันและเวลาในการตรวจสอบและวันเวลาล่าสุดที่ตรวจสอบ แสดงจำนวนข้อที่ผ่านและไม่ผ่าน และแสดงรายละเอียดที่ตรวจสอบตามหัวข้อต่างๆ

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 สรุปผลปริญญานิพนธ์

การพัฒนาเครื่องมือสำหรับตรวจสอบการตั้งค่าตามข้อกำหนดความปลอดภัย ได้พัฒนาเสร็จสิ้นลงตามวัตถุประสงค์ที่ตั้งไว้อย่างสมบูรณ์ โดยในส่วนของผู้งานสามารถใช้เครื่องมือในการตรวจสอบการตั้งค่าตามข้อกำหนดความปลอดภัยได้และได้ผลลัพธ์ถูกต้อง

ผู้จัดทำได้ทำการพัฒนาเครื่องมือสำหรับตรวจสอบการตั้งค่าตามข้อกำหนดความปลอดภัยด้วยการนำเครื่องมือและเทคโนโลยีในปัจจุบันได้แก่เครื่องมือการจัดการการคอนฟิกกูเรชั่นแบบศูนย์กลาง (Configuration Management Tools) ที่ชื่อว่า Ansible มาใช้ในการเขียนรายการตรวจเช็คตามข้อกำหนดต่างๆ และนำเทคโนโลยีเวอร์ชวล แมชชีน (Virtual Machine) รวมกับเทคโนโลยีคอนเทนเนอร์ (Container) ในการจำลองเครื่องเซิร์ฟเวอร์และติดตั้งระบบ Ansible AWX เพื่อเครื่องแม่ข่ายสำหรับทำการตรวจสอบการตั้งค่าตามข้อกำหนดความปลอดภัย

5.2 ข้อดีของระบบ

5.2.1 สามารถเพิ่มผลผลิตในการตรวจสอบระบบให้มากขึ้นในเวลาที่เท่ากัน

5.2.2 สามารถเพิ่มความพร้อมใช้งานในการตรวจสอบระบบสามารถสั่งให้ทำงานได้ทันที่หรือตั้งเวลาในการตรวจสอบล่วงหน้าได้

5.2.3 สามารถลดความผิดพลาดในการตรวจสอบระบบจากการตรวจสอบแบบที่ใช้บุคลากรตรวจสอบลงได้

5.2.4 สามารถลดต้นทุนการจ้างบุคลากรที่มีความสามารถเฉพาะด้านในการตรวจสอบระบบลงได้

5.3 ข้อจำกัดของระบบ

ระบบยังไม่สามารถใส่ข้อยกเว้นสำหรับการตรวจสอบได้

5.4 ข้อเสนอแนะ

เพื่อเพิ่มประสิทธิภาพในการทำงานและให้ระบบมีความสมบูรณ์มากยิ่งขึ้นควรพัฒนาระบบให้สามารถใส่ข้อยกเว้นสำหรับการตรวจสอบได้

บรรณานุกรม

Docker's official. (2019). *Get Docker CE for Ubuntu*. Retrieved from https://docs.docker.com/install/linux/docker-ce/ubuntu

Docker's official. (2019). *Install Docker Compose*. Retrieved from https://docs.docker.com/compose/install

Margaret Rouse. (2005, September). Security Audit [Blog post]. Retrieved from https://searchcio.techtarget.com/definition/security-audit

Shane McDonald. (2019). *credentials.py.j2*. Retrieved from https://github.com/ansible/awx/blob/devel/installer/roles/local_docker/templates/credentials.py.j2

Shane McDonald. (2019). *docker-compose.yml.j2.* Retrieved from https://github.com/ansible/awx/blob/devel/installer/roles/local_docker/templates/docker-compose.yml.j2

Shane McDonald. (2019). *environment.sh.j2.* Retrieved from https://github.com/ansible/awx/blob/devel/installer/roles/local_docker/templates/environment.sh.j2

The PostgreSQL Global Development Group. (2019). *Linux downloads (Ubuntu)*. Retrieved from https://www.postgresql.org/download/linux/ubuntu

Wikipedia. (n.d). Ansible (software). Retrieved November 11, 2560, from https://en.wikipedia.org/wiki/Ansible_(software)

Wikipedia. (n.d). Docker (software). Retrieved December 13, 2560, from https://en.wikipedia.org/wiki/Docker_(software)

Wikipedia. (n.d). PostgreSQL. Retrieved December 10, 2560, from https://en.wikipedia.org/wiki/PostgreSQL

Wikipedia. (n.d). Red Hat. Retrieved December 13, 2560, from https://en.wikipedia.org/wiki/Red_Hat

Wikipedia. (n.d). Virtual machine. Retrieved November 10, 2560, from https://en.wikipedia.org/wiki/Virtual_machine